Msander Reference Manual

Principal contributors to the current codes:

David A. Case (Rutgers)
Carlos Simmerling (Stony Brook)
Adrian Roitberg (Florida)
Kenneth M. Merz (Michigan State)
Tom Darden (OpenEye)
Celeste Sagui (NCSU)
Feng Pan (FSU)
Jason Swails (Entos)
Andreas W. Götz (UC San Diego)
David Cerutti (Rutgers)
Tyler Luchko (CSU Northridge)
Vinícius Wilian D. Cruzeiro (UC San Diego)
Madushanka Manathunga (Michigan State)
Nikolai R. Skrynnikov (Purdue, SPbU)
Oleg Mikhailovskii (Purdue, SPbU)
Yi Xue (Tsinghua)
Sergei A. Izmailov (SPbU)
Alexey Onufriev (Virginia Tech)
Saeed Izadi (Virginia Tech, Genentech)
Xiongwu Wu (NIH)
Holger Gohlke (Düsseldorf/FZ Jülich)
George Giambasu (Rutgers)
Jian Liu (Peking Univ.)
Andriy Kovalenko (NINT)
Peter A. Kollman (UC San Francisco)

Contents

Part I.

Introduction and Installation

1. Introduction

msander ("modern sander") is the basic energy minimizer and molecular dynamics program. This program relaxes the structure by iteratively moving the atoms down the energy gradient until a sufficiently low average gradient is obtained. The molecular dynamics portion generates configurations of the system by integrating Newtonian equations of motion. MD will sample more configurational space than minimization, and will allow the structure to cross over small potential energy barriers. Configurations may be saved at regular intervals during the simulation for later analysis, and basic free energy calculations using thermodynamic integration may be performed. More elaborate conformational searching and modeling MD studies can also be carried out using the *sander* module. This allows a variety of constraints to be added to the basic force field, and has been designed especially for the types of calculations involved in NMR, Xray or cryo-EM structure refinement.

2. The Generalized Born/Surface Area Model

Implicit solvent methods can speed up atomistic simulations by approximating the discrete solvent as a continuum, thus drastically reducing the number of particles in the system. An additional effective speedup often comes from much faster sampling of the conformational space afforded by these methods.[1–5] The generalized Born (GB) solvation model is the most commonly used implicit solvent model for atomistic MD simulation; it has been most widely tested on ff99SB and ff14SBonlysc, but in principle could be used with other non-polarizable force fields, such as ff03. A recent (2019) review gives a good overview.[6] To estimate the total solvation free energy of a molecule, ΔG_{solv}, one typically assumes that it can be decomposed into the "electrostatic" and "non-electrostatic" parts:

$$\Delta G_{solv} = \Delta G_{el} + \Delta G_{nonel} \tag{2.1}$$

where ΔG_{nonel} is the free energy of solvating a molecule from which all charges have been removed (i.e. partial charges of every atom are set to zero), and ΔG_{el} is the free energy of first removing all charges in the vacuum, and then adding them back in the presence of a continuum solvent environment. Generally speaking, ΔG_{nonel} comes from the combined effect of two types of interaction: the favorable van der Waals attraction between the solute and solvent molecules, and the unfavorable cost of breaking the structure of the solvent (water) around the solute. In the current Amber codes, this is taken to be proportional to the total solvent accessible surface area (SA) of the molecule, with a proportionality constant derived from experimental solvation energies of small non-polar molecules, and uses a fast LCPO algorithm [7] to compute an analytical approximation to the solvent accessible area of the molecule.

The Poisson-Boltzmann approach described in the next section has traditionally been used in calculating ΔG_{el}. However, in molecular dynamics applications, the associated computational costs are often very high, as the Poisson-Boltzmann equation needs to be solved every time the conformation of the molecule changes. Amber developers have pursued an alternative approach, the analytic generalized Born (GB) method, to obtain a reasonable, computationally efficient estimate to be used in molecular dynamics simulations. The methodology has become popular,[8–15] especially in molecular dynamics applications,[16–19] due to its relative simplicity and computational efficiency, compared to the more standard numerical solution of the Poisson-Boltzmann equation. Within Amber GB models, each atom in a molecule is represented as a sphere of radius R_i with a charge q_i at its center; the interior of the atom is assumed to be filled uniformly with a material of dielectric constant 1. The molecule is surrounded by a solvent of a high dielectric ε (80 for water at 300 K). The GB model approximates ΔG_{el} by an analytical formula,[8, 20]

$$\Delta G_{el} \approx -\frac{1}{2}\sum_{ij} \frac{q_i q_j}{f_{GB}(r_{ij}, R_i, R_j)} \left(1 - \frac{\exp[-\kappa f_{GB}]}{\varepsilon}\right) \tag{2.2}$$

where r_{ij} is the distance between atoms i and j, the R_i are the so-called *effective Born radii,* and $f_{GB}()$ is a certain smooth function of its arguments. The electrostatic screening effects of (monovalent) salt are incorporated [20] via the Debye-Huckel screening parameter κ.

A common choice [8] of f_{GB} is

$$f_{GB} = \left[r_{ij}^2 + R_i R_j \exp(-r_{ij}^2/4R_i R_j)\right]^{1/2} \tag{2.3}$$

although other expressions have been tried.[11, 21] The effective Born radius of an atom reflects the degree of its burial inside the molecule: for an isolated ion, it is equal to its van der Waals (VDW) radius ρ_i. Then one obtains the particularly simple form:

$$\Delta G_{el} = -\frac{q_i^2}{2\rho_i}\left(1-\frac{1}{\varepsilon}\right) \tag{2.4}$$

where we assumed $\kappa = 0$ (pure water). This is the famous expression due to Born for the solvation energy of a single ion. The function $f_{GB}()$ is designed to interpolate, in a clever manner, between the limit $r_{ij} \to 0$, when atomic spheres merge into one, and the opposite extreme $r_{ij} \to \infty$, when the ions can be treated as point charges obeying the Coulomb's law.[14] For deeply buried atoms, the effective radii are large, $R_i \gg \rho_i$, and for such atoms one can use a rough estimate $R_i \approx L_i$, where L_i is the distance from the atom to the molecular surface. Closer to the surface, the effective radii become smaller, and for a completely solvent exposed side-chain one can expect R_i to approach ρ_i.

The effective radii depend on the molecule's conformation, and so have to be re-computed every time the conformation changes. This makes the computational efficiency a critical issue, and various approximations are normally made that facilitate an effective estimate of R_i. With the exception of GBNSR6 (see Section **??**), the so-called *Coulomb field approximation*, or *CFA*, is used for Amber GB models, which replaces the true electric displacement around the atom by the Coulomb field. Within this assumption, the following expression can be derived:[14]

$$R_i^{-1} = \rho_i^{-1} - \frac{1}{4\pi}\int \theta(|\mathbf{r}|-\rho_i) r^{-4} d^3\mathbf{r} \tag{2.5}$$

where the integral is over the solute volume surrounding atom i. For a realistic molecule, the solute boundary (molecular surface) is anything but trivial, and so further approximations are made to obtain a closed-form analytical expression for the above equation, *e.g.* the so-called pairwise de-screening approach of Hawkins, Cramer and Truhlar,[22] which leads to a GB model implemented in Amber with *igb=1*. The 3D integral used in the estimation of the effective radii is performed over the van der Waals (VDW) spheres of solute atoms, which implies a definition of the solute volume in terms of a set of spheres, rather than the complex molecular surface,[23] commonly used in the PB calculations. For macromolecules, this approach tends to underestimate the effective radii for buried atoms,[14] arguably because the standard integration procedure treats the small vacuum–filled crevices between the van der Waals (VDW) spheres of protein atoms as being filled with water, even for structures with large interior.[21] This error is expected to be greatest for deeply buried atoms characterized by large effective radii, while for the surface atoms it is largely canceled by the opposing error arising from the Coulomb approximation, which tends [9, 13, 24] to overestimate R_i.

The deficiency of the model described above can, to some extent, be corrected by noticing that even the optimal packing of hard spheres, which is a reasonable assumption for biomolecules, still occupies only about three quarters of the space, and so "scaling-up" of the integral by a factor of four thirds should effectively increase the underestimated radii by about the right amount, without any loss of computational efficiency. This idea was developed and applied in the context of pH titration,[14] where it was shown to improve the performance of the GB approximation in calculating pKa values of protein sidechains. However, the one-parameter correction introduced in Ref. [14] was not optimal in keeping the model's established performance on small molecules. It was therefore proposed [19] to re-scale the effective radii with the re-scaling parameters being proportional to the degree of the atom's burial, as quantified by the value I_i of the 3D integral. The latter is large for the deeply buried atoms and small for exposed ones. Consequently, one seeks a well-behaved re-scaling function, such that $R_i \approx (\rho_i^{-1} - I_i)^{-1}$ for small I_i, and $R_i > (\rho_i^{-1} - I_i)^{-1}$ when I_i becomes large. The following simple, infinitely differentiable re-scaling function was chosen to replace the model's original expression for the effective radii:

$$R_i^{-1} = \tilde{\rho}_i^{-1} - \rho_i^{-1}\tanh(\alpha\Psi - \beta\Psi^2 + \gamma\Psi^3) \tag{2.6}$$

where $\Psi = I_i\tilde{\rho}_i$, and α, β, γ are treated as adjustable dimensionless parameters which were optimized using the guidelines mentioned earlier (primarily agreement with the PB). Currently, Amber supports two GB models (termed OBC) based on this idea. These differ by the values of α, β, γ, and are invoked by setting igb to either igb=2 or igb=5. The details of the optimization procedure and the performance of the OBC model relative to the PB treatment and in MD simulations on proteins is described in Ref. [19]; an independent comparison to the PB in calculating the electrostatic part of solvation free energy on a large data set of proteins can be found in Ref. [25].

Our experience with generalized Born simulations is mainly with *ff99SB, ff14SBonlysc* or *ff03*; the current GB models are not compatible with polarizable force fields. Replacing explicit water with a GB model is equivalent to

1	2	5	7	8
mbondi	*mbondi2*	*mbondi2*	*bondi*	*mbondi3*

Table 2.1.: *Recommended radii sets for various GB models. For values of igb given in the top row, the string in the second row should be entered in LEaP as "set default PBRadii xxx".*

specifying a different force field, and users should be aware that none of the GB options (in Amber or elsewhere) is as mature as simulations with explicit solvent; user discretion is advised. For example, it was shown that salt bridges are too strong in some of these models [26, 27] and some of them provide secondary structure distributions that differ significantly from those obtained using the same protein parameters in explicit solvent, with GB having too much α-helix present.[28, 29] The combination of the *ff14SBonlysc* force field with igb=8 gives the best results for proteins [30][31], nucleic acids and protein-nucleic acid complexes. [32]

Despite these limitations, implicit treatment of solvent is widely used in molecular simulations for two main reasons: algorithmic/computational speed and conformational sampling. [5, 33] Implicit solvent methods can be algorithmically/computationally faster, as measured by simulation time steps per processor (CPU) time, because the vast number of individual interactions between the atoms of individual solvent molecules do not need to be explicitly computed. Implicit-solvent simulations can also sample conformational space faster in the low viscosity regime afforded by the implicit solvent model.[1–5] To some extent, the interest in implicit-solvent-based simulations is motivated by the need to sample very large conformational spaces for problems such as protein folding, binding-affinity calculations, or large-scale fluctuations of nucleosomal DNA fragments. The speedup of conformational change can vary considerably, depending on the details of the transition, and can range from no speedup at all to almost a 100-fold speedup. [5] In general, the larger the conformational change, the higher the speedup one may expect, but this tendency is not universal or uniform. These speedup values are also expected to vary by the specific flavour of GB model used, a detailed analysis for igb5 can be found in Ref. [5].

The generalized Born models used here are based on the "pairwise" model introduced by Hawkins, Cramer and Truhlar,[22, 34] which in turn is based on earlier ideas by Still and others.[8, 13, 24, 35] The so-called overlap parameters for most models are taken from the Tinker molecular modeling package (http://tinker.wustl.edu). The effects of added monovalent salt are included at a level that approximates the solutions of the linearized Poisson-Boltzmann equation.[20] The original implementation was by David Case, who thanks Charlie Brooks for inspiration. Details of our implementation of generalized Born models can be found in Refs. [36, 37].

2.1. GB/SA input parameters

As outlined above, there are several "flavors" of GB available, depending upon the value of *igb*. The version that has been most extensively tested corresponds to *igb*=1; the "OBC" models (*igb*=2 and 5) are newer, but appear to give significant improvements and are recommended for most projects (certainly for peptides or proteins). The newest, most advanced, and least extensively tested model, *GBn* (*igb*=7), yields results in considerably better agreement with molecular surface Poisson-Boltzmann and explicit solvent results than the "OBC" models under many circumstances.[29] The *GBn* model was parameterized for peptide and protein systems and is not recommended for use with nucleic acids. A modification on the GBn model (igb=8) further improves agreement between Poisson-Boltzmann and explicit solvent data compared to the original formulation (igb=7).[30] Users should understand that all (current) GB models have limitations and should proceed with caution. Generalized Born simulations can only be run for non-periodic systems, *i.e.* where *ntb*=0. Unlike its use in explicit solvent PME simulations, short nonbonded cutoff values have much stronger impact on accuracy of the GB calculations. Essentially, any cutoff values other than *cut* > *structure size* can lead to artifacts. Current GPU implementation of the GB can not use cutoffs. An alternative that retains most of the speed of the GB with a cutoff, but without most of its artifacts, is GB-HCP described in Section **??**. If the nonbonded cutoff is used in GB calculations, it should be greater than that for PME calculations, perhaps cut=16. The slowly-varying forces generally do not have to be evaluated at every step for GB, either nrespa=2 or 4, although that option may lead to some artifacts as well.

igb **= 0** No generalized Born term is used. (Default)

= 1 The Hawkins, Cramer, Truhlar[22, 34] pairwise generalized Born model is used, with parameters described by Tsui and Case.[36] This model uses the default radii set up by LEaP. It is slightly different from the GB model that was included in Amber6. If you want to compare to Amber 6, or need to continue an ongoing simulation, you should use the command "set default PBradii amber6" in LEaP, and set *igb*=1 in *sander*. For reference, the Amber6 values are those used by an earlier Tsui and Case paper.[17] Note that most nucleic acid simulations have used this model, so you take care when using other values. Also note that Tsui and Case used an offset (see below) of 0.13 Å, which is different from its default value.

= 2 Use a modified GB model developed by A. Onufriev, D. Bashford and D.A. Case; the main idea was published earlier,[14] but the actual implementation here[19] is an elaboration of this initial idea. Within this model, the effective Born radii are re-scaled to account for the interstitial spaces between atom spheres missed by the GB^{HCT} approximation. In that sense, GB^{OBC} is intended to be a closer approximation to true molecular volume, albeit in an average sense. With *igb*=2, the inverse of the effective Born radius is given

by:cedure

$$R_i^{-1} = \overline{\rho}_i^{-1} - \tanh\left(\alpha\Psi - \beta\Psi^2 + \gamma\Psi^3\right)/\rho_i$$

where $\overline{\rho}_i = \rho_i - offset$, and $\Psi = I\rho_i$, with I given in our earlier paper. The parameters α, β, and γ were determined by empirical fits, and have the values 0.8, 0.0, and 2.909125. This corresponds to model I in Ref [19]. With this option, you should use the LEaP command "set default PBradii mbondi2" to prepare the *prmtop* file.

= 3 or 4 These values are unused; they were used in Amber 7 for parameter sets that are no longer supported.

= 5 Same as *igb=2*, except that now α, β, γ are 1.0, 0.8, and 4.85. This corresponds to model II in Ref [19]. With this option, you should use the command "set default PBradii mbondi2" in setting up the *prmtop* file, although "set default PBradii bondi" is also OK. When tested in MD simulations of several proteins,[19] both of the above parameterizations of the "OBC" model showed equal performance, although further tests [25] on an extensive set of protein structures revealed that the *igb=5* variant agrees better with the Poisson-Boltzmann treatment in calculating the electrostatic part of the solvation free energy.

= 6 With this option, there is no continuum solvent model used at all; this corresponds to a non-periodic, "vacuum", model where the non-bonded interactions are just Lennard-Jones and Coulomb interactions.

= 7 The *GBn* model described by Mongan, Simmerling, McCammon, Case and Onufriev[38] is employed. This model uses a pairwise correction term to GB^{HCT} to approximate a molecular surface dielectric boundary; that is to eliminate interstitial regions of high dielectric smaller than a solvent molecule. This correction affects all atoms and is geometry-specific, going beyond the geometry-free, "average" re-scaling approach of GB^{OBC}, which mostly affects buried atoms. With this method, you should use the bondi radii set. The overlap or screening parameters in the *prmtop* file are ignored, and the model-specific *GBn* optimized values are substituted. The model carries little additional computational overhead relative to the other GB models described above.[38] This method is not recommended fcedureor systems involving nucleic acids.

= 8 Same GB functional form as the GBn model (igb=7), but with different parameters. The offset, overlap screening parameters, and gbneckscale are changed. In addition, individual α , β , and γ parameters can be specified for each of the elements H, C, N, O, S, P. Parameters for other elements have not been optimized, and the default values used are the ones from igb=5, which were not element-dependent. Default values were optimized for H, C, N, O and S atoms in protein systems.[30] Although the parameters for P in proteins can be specified, the default values were not optimized and are the igb=5 values. Nucleic acids have separate

parameters from those used for proteins, and default values were optimized for H, C, N, O and P atoms in nucleic acid systems.[32]

The following are the default parameters sander uses with *igb=8*:

```
Sh=1.425952, Sc=1.058554, Sn=0.733599,
So=1.061039, Ss=-0.703469, Sp=0.5,
offset=0.195141, gbneckscale=0.826836,
gbalphaH=0.788440, gbbetaH=0.798699, gbgammaH=0.437334,
gbalphaC=0.733756, gbbetaC=0.506378, gbgammaC=0.205844,
gbalphaN=0.503364, gbbetaN=0.316828, gbgammaN=0.192915,
gbalphaOS=0.867814, gbbetaOS=0.876635, gbgammaOS=0.387882,
gbalphaP=1.0, gbbetaP=0.8, gbgammaP=4.85
screen_hnu=1.69654, screen_cnu=1.26890,
screen_nnu=1.425974, screen_onu=0.18401, screen_pnu=1.54506,
gb_alpha_hnu=0.53705, gb_beta_hnu=0.36286, gb_gamma_hnu=0.11670,
gb_alpha_cnu=0.33167, gb_beta_cnu=0.19684, gb_gamma_cnu=0.09342,
gb_alpha_nnu=0.68631, gb_beta_nnu=0.46319, gb_gamma_nnu=0.13872,
gb_alpha_onu=0.60634, gb_beta_onu=0.46301, gb_gamma_onu=0.14226,
gb_alpha_pnu=0.41836, gb_beta_pnu=0.29005, gb_gamma_pnu=0.10642
```

Parameters for proteins and for nucleic acids were optimized separately and can be independently specified. Protein parameters: Sh, Sc, Sn, So, Ss and Sp are scaling parameters, gbalphaX, gbbetaX, gbgammaX are the α , β , γ set for element X. gbalphaOS, gbbetaOS, gbgammaOS is the α , β , γ set applied to both O and S. The phosphorus parameters (in proteins) were not optimized and are simply taken as the parameters used in the OBC-2 model (igb=5). Nucleic acid parameters (end with "nu"): screen_Xnu (X=h, c, n, o, p) are scaling parameters, gb_alpha_Xnu (X=h, c, n, o, p) are the α , β , γ set for element X.

Since parameters are assigned for each atom based on its residue name (hard-coded in "sander/egb.F90" (subroutine isnucat)), users need to update the residue table in the sander source code if nucleic acids with different names are simulated using this GB model.

The default values for offset=0.195141, gbneckscale=0.826836 are recommended for both proteins and nucleic acids.

mbondi3 radii are recommended with igb=8 and can be employed with the LEaP command "set default PBradii mbondi3". The mbondi3 radii were adjusted based on protein simulations, and optimization of these radii for nucleic acids is currently underway.

=10 Calculate the reaction field and nonbonded interactions using a numerical Poisson-Boltzmann solver. This option is described in the Chapter **??**. Note that this is *not* a generalized Born simulation, in spite of its use of *igb*; it is rather an alternative continuum solvent model.

`intdiel` Sets the interior dielectric constant of the molecule of interest. Default is 1.0. Other values have not been extensively tested.

`extdiel` Sets the exterior or solvent dielectric constant. Default is 78.5.

`saltcon` Sets the concentration (M) of 1-1 mobile counterions in solution, using a modified generalized Born theory based on the Debye-Hückel limiting law for ion screening of interactions.[20] Default is 0.0 M (*i.e.* no Debye-Hückel screening.) Setting *saltcon* to a nonzero value does result in some increase in computation time.

`rgbmax` This parameter controls the maximum distance between atom pairs that will be considered in carrying out the pairwise summation involved in calculating the effective Born radii. Atoms whose associated spheres are farther way than *rgbmax* from given atom will not contribute to that atom's effective Born radius. This is implemented in a "smooth" fashion (thanks mainly to W.A. Svrcek-Seiler), so that when part of an atom's atomic sphere lies inside *rgbmax* cutoff, that part contributes

to the low-dielectric region that determines the effective Born radius. The default is 25 Å, which is usually plenty for single-domain proteins of a few hundred residues. Even smaller values (of 10-15 Å) are reasonable, changing the functional form of the generalized Born theory a little bit, in exchange for a considerable speed-up in efficiency, and without introducing the usual cut-off artifacts such as drifts in the total energy.
The *rgbmax* parameter affects only the effective Born radii (and the derivatives of these values with respect to atomic coordinates). The *cut* parameter, on the other hand, determines the maximum distance for the electrostatic, van der Waals and "off-diagonal" terms of the generalized Born interaction. The value of *rgbmax* might be either greater or smaller than that of *cut*: these two parameters are independent of each other. However, values of *cut* that are too small are more likely to lead to artifacts than are small values of *rgbmax*; therefore one typically sets *rgbmax* <= *cut*.

`rbornstat` If *rbornstat = 1*, the statistics of the effective Born radii for each atom of the molecule throughout the molecular dynamics simulation are reported in the output file. Default is 0.

`offset` The dielectric radii for generalized Born calculations are decreased by a uniform value "offset" to give the "intrinsic radii" used to obtain effective Born radii. Default is 0.09 Å.

`gbsa` Option to carry out GB/SA (generalized Born/surface area) simulations. For the default value of 0, surface area will not be computed and will not be included in the solvation term. If *gbsa = 1*, surface area will be computed using the LCPO model.[7] If *gbsa = 2*, surface area will be computed by recursively approximating a sphere around an atom, starting from an icosahedra. Note that no forces are generated in this case, hence, *gbsa = 2* only works for a single point energy calculation and is mainly intended for energy decomposition in the realm of MM-GBSA. If *gbsa = 3,* surface area will be computed using a fast pairwise approximation [39] suitable for GPU computing in pmemd.cuda program; the acceleration in pmemd.cuda compared with *gbsa = 2* is ~30 times faster [39]. Note that *gbsa = 3* is currently not supported in sander, MM-GBSA, QM/MM or libsff. Although *gbsa = 3* is supported in pmemd, the general usage is not recommended as the speed gain is trivial, given that the algorithm was particularly designed for fast approximation of surface area in GPU-accelerated GB simulations. Therefore, we recommend users to use gbsa=3 with pmemd.cuda.

`surften` Surface tension used to calculate the nonpolar contribution to the free energy of solvation (when *gbsa = 1*), as Enp = surften*SA. The default is 0.005 kcal/mol/A^2.[40] For *gbsa = 3*, suften works comparably with *gbsa = 1* given the same value. [39]

`rdt` This parameter is only used for GB simulations with LES (Locally Enhanced Sampling). In GB+LES simulations, non-LES atoms require multiple effective Born radii due to alternate de-screening effects of different LES copies. When the multiple radii for a non-LES atom differ by less than RDT, only a single radius will be used for that atom. See Chapter 14 for more details. Default is 0.0 Å.

3. Reference Interaction Site Model

In addition to explicit and continuum implicit solvation models, Amber also has a third type of solvation model for molecular mechanics simulations, the reference interaction site model (RISM) of molecular solvation[41–54]. In AmberTools, 1D-RISM is available as rism1d. 3D-RISM is available as an option in NAB, MMPBSA.py and sander. rism3d.snglpnt is a simplified, standalone interface, ideal for calculating solvation thermodynamics on individual structures and trajectories. Details specific to using sander and sander.MPI can be found in Chapter 8.

3.1. Introduction

RISM is an inherently microscopic approach, calculating the equilibrium distribution of the solvent, from which all thermodynamic properties are then determined. Specifically, RISM is an approximate solution to the Ornstein-Zernike (OZ) equation[42, 51, 52, 55, 56]

$$h(r_{12},\Omega_1,\Omega_2) = c(r_{12},\Omega_1,\Omega_2) + \rho \int d\mathbf{r}_3\, d\Omega_3\, c(r_{13},\Omega_1,\Omega_3)\, h(r_{32},\Omega_3,\Omega_2), \tag{3.1}$$

where r_{12} is the separation between particles 1 and 2 while Ω_1 and Ω_2 are their orientations relative to the vector $\mathbf{r}_{12}$. The two functions in this relation are h, the total correlation function, and c, the direct correlation function. The total correlation function is defined as

$$h_{ab}(r_{ab},\Omega_a,\Omega_b) \equiv g_{ab}(r_{ab},\Omega_a,\Omega_b) - 1,$$

where g_{ab} is the pair-distribution function, which gives the conditional density distribution of species b about a. In cases where only radial separation is considered, for example by orientational averaging over site α of species a and site γ of species b, gives the familiar one dimensional site-site radial distribution function, $g_{\alpha\gamma}(r_{\alpha\gamma})$.

For real mixtures, it is often convenient to speak in terms of a solvent, V, of high concentration and a solute, U, of low concentration. A generic case of solvation is infinite dilution of the solute, i.e., $\rho^{\mathrm{U}} \to 0$. We can rewrite Equation (3.1), in the limit of infinite dilution, as a set of three equations:

$$h^{\mathrm{VV}}(r_{12},\Omega_1,\Omega_2) = c^{\mathrm{VV}}(r_{12},\Omega_1,\Omega_2) + \rho^{\mathrm{V}} \int d\mathbf{r}_3\, d\Omega_3\, c^{\mathrm{VV}}(r_{13},\Omega_1,\Omega_3)\, h^{\mathrm{VV}}(r_{32},\Omega_3,\Omega_2), \tag{3.2}$$

$$h^{\mathrm{UV}}(r_{12},\Omega_1,\Omega_2) = c^{\mathrm{UV}}(r_{12},\Omega_1,\Omega_2) + \rho^{\mathrm{V}} \int d\mathbf{r}_3\, d\Omega_3\, c^{\mathrm{UV}}(r_{13},\Omega_1,\Omega_3)\, h^{\mathrm{VV}}(r_{32},\Omega_3,\Omega_2), \tag{3.3}$$

$$h^{\mathrm{UU}}(r_{12},\Omega_1,\Omega_2) = c^{\mathrm{UU}}(r_{12},\Omega_1,\Omega_2) + \rho^{\mathrm{V}} \int d\mathbf{r}_3\, d\Omega_3\, c^{\mathrm{UV}}(r_{13},\Omega_1,\Omega_3)\, h^{\mathrm{VU}}(r_{32},\Omega_3,\Omega_2). \tag{3.4}$$

Equation (3.3) is directly relevant for biomolecular simulations where we are often interested in the properties of a single, arbitrarily complex solute in the solution phase. Solutions to Equation (3.3) can be obtained using 3D-RISM. However, a solution to Equation (3.2) for pure solvent is a necessary prerequisite and is readily obtained from 1D-RISM.

To obtain a solution to the OZ equations it is necessary to have a second equation that relates h and c or uniquely defines one of these functions. The general closure relation is[55]

$$g(r_{12},\Omega_1,\Omega_2) = \exp\left[-\beta u(r_{12},\Omega_1,\Omega_2) + h(r_{12},\Omega_1,\Omega_2) - c(r_{12},\Omega_1,\Omega_2) + b(r_{12},\Omega_1,\Omega_2)\right] \tag{3.5}$$

u is the potential energy function for the two particles and b is known as the bridge function (a non-local functional, representable as infinite diagrammatic series in terms of h [55]). It should be noted that u is the only point at which the interaction potential enters the equations. Depending on the method used to solve the OZ equations, u is

generally an explicit potential. In principle, it should now be possible to solve our two equations. For example, we may wish to use SPC/E as a water model. Inputting the relevant aspects of the SPC/E model into u, 1D-RISM can be used to calculate the equilibrium properties of the SPC/E model. A different explicit water model will yield different properties.

A fundamental problem for all OZ-like integral equation theories is the bridge function, which contains multiple integrals that are readily solved only in special circumstances. In practice, an approximate closure relation must be used. While many closures have been developed, at this time only three are implemented in 3D-RISM: hypernetted-chain approximation (HNC), Kovalenko-Hirata (KH) and the partial series expansion of order-n (PSE-n).

For HNC, we set $b = 0$, giving[55]

$$\begin{aligned} g^{\mathrm{HNC}}(r_{12},\Omega_1,\Omega_2) &= \exp\left(-\beta u(r_{12},\Omega_1,\Omega_2) + h(r_{12},\Omega_1,\Omega_2) - c(r_{12},\Omega_1,\Omega_2)\right) \\ &= \exp\left(t^*(r_{12},\Omega_1,\Omega_2)\right) \end{aligned} \tag{3.6}$$

where t^* is the renormalize-indirect correlation function. HNC works well in many situations, including charged particles, but has difficulties when the size ratios of particles in the system are highly varied and may not always converge on a solution when one should exist. Also, as the bridge term is generally repulsive, HNC allows particles to approach too closely, overestimating non-Coulombic interactions[52].

KH is a combination of HNC and the mean spherical approximation (MSA), the former being applied to the spatial regions of solvent density depletion ($g < 1$), including the repulsive core, and the latter to those of solvent density enrichment ($g > 1$), such as association peaks[51, 52]

$$g^{\mathrm{KH}}(r_{12},\Omega_1,\Omega_2) = \begin{cases} \exp\left(t^*(r_{12},\Omega_1,\Omega_2)\right) & \text{for } g(r_{12},\Omega_1,\Omega_2) \leq 1 \\ 1 + t^*(r_{12},\Omega_1,\Omega_2) & \text{for } g(r_{12},\Omega_1,\Omega_2) > 1 \end{cases}. \tag{3.7}$$

Like HNC, KH handles Coulombic systems well but overestimates non-Coulombic interactions. Unlike HNC, it does not have difficulties with highly asymmetric particle sizes and readily converges to stable solutions for almost all systems of practical interest. The reliability of the KH closure makes it particularly suitable for molecular mechanics calculations.

PSE-n offers the ability to interpolate between KH and HNC. Here, the exponential regions of solvent density enrichment are treated as a Taylor expansion,

$$g^{\text{PSE-n}}(r_{12},\Omega_1,\Omega_2) = \begin{cases} \exp\left(t^*(r_{12},\Omega_1,\Omega_2)\right) & \text{for } g(r_{12},\Omega_1,\Omega_2) \leq 1 \\ \sum_{i=0}^{n} {(t^*(r_{12},\Omega_1,\Omega_2))^i}/{i!} & \text{for } g(r_{12},\Omega_1,\Omega_2) > 1 \end{cases}. \tag{3.8}$$

In the case of $n = 1$, the KH closure is obtained, while in the limit of $n \to \infty$ HNC is recovered. This allows a balance between the numerical stability of KH and the often better accuracy of HNC.

3.1.1. 1D-RISM

1D-RISM is used to calculate bulk properties of the solvent and is a prerequisite for 3D-RISM, for which the primary result is the bulk solvent site-site susceptibility in reciprocal space, $\chi^{\mathrm{VV}}(k)$. As its name would suggest, 1D-RISM is a one-dimensional calculation. The six-dimensional OZ equations are reduced to one dimension (radial separation) via the fundamental RISM approximation[42–45, 55, 56], which produces the intramolecular pair correlation matrix,

$$\omega_{\alpha\gamma}(k) = \sin(kr_{\alpha\gamma})/(kr_{\alpha\gamma}) \tag{3.9}$$

where α and γ label the different atom types in the model. Note that atoms of the same type in RISM theory have the same Lennard-Jones and Coulomb parameters. For example, most three site water models have two RISM types, oxygen and hydrogen. Depending on the model, propane, C_3H_8, may have two carbon types and two

hydrogen types. Equation (3.2) then becomes

$$\begin{aligned} h_{\alpha\gamma}(r) &= \sum_{\mu\nu}\int d\mathbf{r}'d\mathbf{r}''\,\omega_{\alpha\mu}(|r-r'|)c_{\mu\nu}(|r'-r''|)\left[\omega_{\nu\gamma}(r'')+\rho_\nu h_{\nu\gamma}(r'')\right] \\ &= \frac{1}{(2\pi)^3}\int e^{i\mathbf{k}\cdot\mathbf{r}}d\mathbf{k}\left[\omega\mathbf{c}\left[\mathbf{1}-\rho\omega\mathbf{c}\right]^{-1}\omega\right]_{\alpha\gamma} \\ &= \sum_0^\infty \omega(k)\mathbf{c}(k)\omega(k)\left[\rho\mathbf{c}(k)\omega(k)\right]^n. \end{aligned} \tag{3.10}$$

Equation (3.10) must be complemented with one of the five closures currently supported by rism1d (see Subsection **??**). In 1d, these are site-site closures and there is no orientational dependence. For example, the HNC closure (Eq. (3.6)) becomes,

$$g^{\mathrm{HNC}}_{\alpha\gamma}(r) = \exp\left[-\beta u_{\alpha\gamma}(r)+h_{\alpha\gamma}(r)-c_{\alpha\gamma}(r)\right]. \tag{3.11}$$

Equation (3.10), with KH, HNC or PSE-*n* closures, is readily applicable to liquid mixtures, with site indices of the site-site correlation functions enumerating interaction sites on all (different) species in the solution and the intramolecular matrix (3.9) set equal to zero for sites α,γ belonging to different species.

A dielectrically consistent version of 1D-RISM theory (DRISM) enforces the proper dielectric asymptotics of the site-site correlation functions, and so provides the self-consistent dielectric properties of electrolyte solution with polar solvent and salt in a range of concentrations, including the given dielectric constant of the solution [57].

The 1D-RISM integral equations are then solved for the site-site direct correlation function in an iterative manner, accelerated by the modified direct inversion of the iterative subspace (MDIIS) [52, 58]. All correlation functions are represented as one-dimensional grids and the convolution integrals in Equation (3.10) are performed in reciprocal space by making use of a fast Fourier transform applied to the short-range parts of all the correlations, while the electrostatic asymptotics are separated out and Fourier transformed analytically [52–54].

1D-RISM is a general method and not restricted to water or pure solvents. For example, 1D-RISM may be used to treat solutions of aqueous alkali and halide ions at various concentrations [59]. The output from 1D-RISM can then be used for complex solutes, such as DNA [60], in 3D-RISM.

3.1.2. 3D-RISM

With the results from 1D-RISM, a 3D-RISM calculation for a specific solute can be carried out. For 3D-RISM calculations, only the solvent orientational degrees of freedom are averaged over and Equation (3.3) becomes[50, 51]

$$h^{\mathrm{UV}}_\gamma(\mathbf{r}) = \sum_\alpha \int d\mathbf{r}'c^{\mathrm{UV}}_\alpha(\mathbf{r}-\mathbf{r}')\,\chi^{\mathrm{VV}}_{\alpha\gamma}(r'), \tag{3.12}$$

where $\chi^{\mathrm{VV}}_{\alpha\gamma}(r)$ is the site-site susceptibility of the solvent, obtained from 1D-RISM and given by

$$\chi^{\mathrm{VV}}_{\alpha\gamma}(r) = \omega^{\mathrm{VV}}_{\alpha\gamma}(r)+\rho_\alpha h^{\mathrm{VV}}_{\alpha\gamma}(r).$$

3D-RISM supports HNC, KH and PSE-*n* closures (see Sections **??**, **??** and **??**). As with the 1D-RISM closures, these are constructed by analogy from Eqs. 3.6-3.8. For example, HNC becomes

$$g^{\mathrm{HNC,UV}}_\gamma(\mathbf{r}) = \exp\left(-\beta u^{\mathrm{UV}}_\gamma(\mathbf{r})+h^{\mathrm{UV}}_\gamma(\mathbf{r})-c^{\mathrm{UV}}_\gamma(\mathbf{r})\right). \tag{3.13}$$

As with 1D-RISM, correlation functions are represented on (3D) grids, convolution integrals are performed in reciprocal space and a self-consistent solution is iteratively converged upon using the MDIIS accelerated solver. There is one 3D grid for each solvent type for each correlation function. For example, for a solute in SPC/E water there will be both $g^{\mathrm{UV}}_{\mathrm{H}}(\mathbf{r})$ and $g^{\mathrm{UV}}_{\mathrm{O}}(\mathbf{r})$ grids. Each point on the $g^{\mathrm{UV}}_{\mathrm{H}}(\mathbf{r})$ will give the fractional density of water hydrogen a that location of real-space.

To properly treat electrostatic forces in electrolyte solution with polar molecular solvent and ionic species, the electrostatic asymptotics of all the correlation functions (both the 3D and radial ones) are treated analytically [52,

53, 61]. The non-periodic electrostatic asymptotics are separated out in the direct and reciprocal space and the remaining short-range terms of the correlation functions are discretized on a 3D grid in a non-periodic box large enough to ensure decay of the short-range terms at the box boundaries [61]. The convolution of the short-range terms in the integral equation (3.12) is calculated using 3D fast Fourier transform [62, 63]. Accordingly, the electrostatic asymptotics terms in the thermodynamics integral (3.15) below are handled analytically and reduced to one-dimensional integrals easy to compute [61].

With a converged 3D-RISM solution for h^{UV} and c^{UV}, it is straightforward to calculate solvation thermodynamics. From the perspective of molecular simulations, the most important thermodynamic values are the excess chemical potential of solvation (solvation free energy), μ^{ex} and the mean solvation force, $\mathbf{f}_i^{\mathrm{UV}}(\mathbf{R}_i)$, on each solute atom, i. μ^{ex} can be obtained through analytical thermodynamic integration for HNC,

$$\mu^{\mathrm{ex,HNC}} = k_{\mathrm{B}}T\sum_{\alpha}\rho_{\alpha}^{\mathrm{V}}\int d\mathbf{r}\left[\frac{1}{2}\left(h_{\alpha}^{\mathrm{UV}}(\mathbf{r})\right)^2 - c_{\alpha}^{\mathrm{UV}}(\mathbf{r}) - \frac{1}{2}h_{\alpha}^{\mathrm{UV}}(\mathbf{r})c_{\alpha}^{\mathrm{UV}}(\mathbf{r})\right], \tag{3.14}$$

KH ,

$$\mu^{\mathrm{ex,KH}} = k_{\mathrm{B}}T\sum_{\alpha}\rho_{\alpha}^{\mathrm{V}}\int d\mathbf{r}\left[\frac{1}{2}\left(h_{\alpha}^{\mathrm{UV}}(\mathbf{r})\right)^2\Theta\left(-h_{\alpha}^{\mathrm{UV}}(\mathbf{r})\right) - c_{\alpha}^{\mathrm{UV}}(\mathbf{r}) - \frac{1}{2}h_{\alpha}^{\mathrm{UV}}(\mathbf{r})c_{\alpha}^{\mathrm{UV}}(\mathbf{r})\right], \tag{3.15}$$

and PSE-n,

$$\mu^{\mathrm{ex,PSE}\text{-}n} = k_{\mathrm{B}}T\sum_{\alpha}\rho_{\alpha}^{\mathrm{V}}\int d\mathbf{r}\left[\frac{1}{2}\left(h_{\alpha}^{\mathrm{UV}}(\mathbf{r})\right)^2 - c_{\alpha}^{\mathrm{UV}}(\mathbf{r}) - \frac{1}{2}h_{\alpha}^{\mathrm{UV}}(\mathbf{r})c_{\alpha}^{\mathrm{UV}}(\mathbf{r}) - \frac{\left(t^*(\mathbf{r})\right)^{n+1}}{(n+1)!}\Theta\left(h_{\alpha}^{\mathrm{UV}}(\mathbf{r})\right)\right], \tag{3.16}$$

where Θ is the Heaviside function.

Analogous versions of Eqns. 3.6, 3.15 and 3.16 are used in 1D-RISM. While these are used for DRISM they are have been derived for XRISM. Furthermore, these equations have been derived a number of different ways with slightly different functional forms of the $-\frac{1}{2}hc$ term [51, 64–67]. These different functional forms are equivalent in XRISM but not in DRISM. The form introduced by Pettitt and Rossky [65] is the most popular in the literature and the default selection in `rism1d`. It is possible to have `rism1d` evaluate and output all three functional forms (see **??**) but, for DRISM, none of these expressions are strictly correct.

The force equation

$$\mathbf{f}_i^{\mathrm{UV}}(\mathbf{R}_i) = -\frac{\partial\mu^{\mathrm{ex}}}{\partial\mathbf{R}_i} = -\sum_{\alpha}\rho_{\alpha}\int d\mathbf{r}\, g_{\alpha}^{\mathrm{UV}}(\mathbf{r})\frac{\partial u_{\alpha}^{\mathrm{UV}}(\mathbf{r}-\mathbf{R}_i)}{\partial\mathbf{R}_i}$$

is valid for all closures with a path independent expression for the excess chemical potential, such as HNC, KH and PSE-n closures implemented in 3D-RISM [41, 68–70].

In addition to closure specific expressions for the solvation free energy, other approximations also exist. The Gaussian fluctuation (GF) approximation[71, 72] is given as

$$\mu^{\mathrm{ex,GF}} = k_{\mathrm{B}}T\sum_{\alpha}\rho_{\alpha}^{\mathrm{V}}\int d\mathbf{r}\left[-c_{\alpha}^{\mathrm{UV}}(\mathbf{r}) - \frac{1}{2}h_{\alpha}^{\mathrm{UV}}(\mathbf{r})\,c_{\alpha}^{\mathrm{UV}}(\mathbf{r})\right] \tag{3.17}$$

and has been shown to yield improved absolute solvation free energies for both polar and non-polar solutes[72, 73] but not necessarily for relative free energies[74]. It is not associated with a particular closure but is typically used in place of the expression for a given closure.

Eqs. (3.14)-(3.16) give the total solvation free energy, ΔG_{sol}, but it is often useful to decompose this into electrostatic (solvent polarization), ΔG_{pol}, and non-electrostatic (dispersion and cavity formation), $(\Delta G_{\mathrm{dis}} + \Delta G_{\mathrm{cav}})$, terms. Conceptually, we can divide the path of the thermodynamic integration into two steps: first the solute without partial charges is inserted into the solvent (dispersion and cavity formation) and then partial charges are introduced, which polarize the solvent,

$$\mu^{\mathrm{ex}} = \Delta G_{\mathrm{sol}} = \Delta G_{\mathrm{pol}} + \Delta G_{\mathrm{dis}} + \Delta G_{\mathrm{cav}}.$$

ΔG_{sol} is produced by a 3D-RISM calculation on the charged solute. ΔG_{pol} is then the difference of the two calculations. As a point of reference, generalized-Born and Poisson-Boltzmann methods calculate only ΔG_{pol} and, typically, use a calculation involving solvent accessible surface area to predict $\Delta G_{\text{dis}} + \Delta G_{\text{cav}}$.

3.2. Practical Considerations

3.2.1. Computational Requirements and Parallel Scaling

Calculating a 3D-RISM solution for a single solute conformation typically requires about 100 times more computer time than the same calculation with explicit solvent or PB. While there are other factors to consider, such as sampling confined solvent or overall efficiency of sampling in the whole statistical ensemble at once, this can be prohibitive for many applications. Memory is also an issue as the 3D correlation grids require anywhere from a few megabytes for the smallest solutes to gigabytes for large complexes. A lower bound and very good estimate for the total memory required is

$$\text{Total memory} \geq 8\,\text{bytes} \times \left[N_{\text{box}}\, N^{\text{V}} \left(\underbrace{2N_{\text{MDIIS}}}_{c,\text{residual}} + \underbrace{1}_{u} + \underbrace{N_{\text{decomp}}}_{\text{polar decomp}} \underbrace{N_{\text{propagate}}}_{\text{past solutions}} \right) \right.$$
$$\left. (N_{\text{box}} + 2N_y N_z) \left\{ \underbrace{4}_{\text{asymptotics}} + \underbrace{1}_{\text{FFT scratch}} + \underbrace{2}_{g,h} N^{\text{V}} \right\} \right]$$

where $N_{\text{box}} = N_x \times N_y \times N_z$ is the total number of grid points, N^{V} is the number of solvent atom species and N_{MDIIS} is the number of MDIIS vectors used to accelerate convergence. u^{UV}, c^{UV} and the residual of c^{UV} are stored in real-space only and require a full grid for each solvent. c^{UV} and its residual also require N_{MDIIS} grids for the MDIIS routine (see the mdiis_nvec keyword) and $N_{\text{propagate}}$ grids to make use of solutions from previous solute configurations to improve the initial guess (see the npropagate keyword). If a polar/non-polar decomposition is requested (see the polardecomp keyword) an additional set of grids for past solutions with no solute charges is kept ($N_{\text{decomp}} = 2$); by default this is turned off ($N_{\text{decomp}} = 1$). The full real space grid plus an additional $2N_yN_x$ grid points are needed (due to the FFT) for g and h for each solvent species and for the four grids required to compute the long range asymptotics. Memory, therefore, scales linearly with N_{box} while computation time scales as $O(N_{\text{box}} \log(N_{\text{box}}))$ due to the requirements of calculating the 3D fast Fourier transform (3D-FFT). To overcome these requirements, two options are available beyond optimizations already in place, multiple time steps and parallelization. Multiple time step methods are available only in sander (Chapter 8) and are applicable to molecular dynamics calculations only. Parallelization is available for all calculations but is limited by system size and computational resources.

Both `sander` and `NAB` have MPI implementations of 3D-RISM (see Section **??** for NAB compiling instructions) that distribute both memory requirements and computational load. As memory is distributed, the aggregate memory of many computers can be used to perform calculations on very large systems. Memory distribution is handled by the FFTW 3.3 library so decomposition is done along the z-axis. If a variable solvation box size is used, the only consideration is to avoid specifying a large, prime number of processes (≥ 7). For fixed box sizes, the number of grids points in each dimension must be divisible by two (a general requirement) and the number of grid points in the z-axis must be divisible by the number of processes. `sander.MPI` also has the additional consideration that the number of processes cannot be larger than the number of solute residues; `NAB` does not suffer from this limitation.

3.2.2. Output

g^{UV}, h^{UV} and c^{UV} files can be output for 3D-RISM calculations and are useful for visualization and calculation of thermodynamic quantities. As all file formats save only one density per file (see https://ambermd.org/FileFormats.php), there is one file for each solvent atom type for each requested frame. For the default MRC format, each file is

$(256 + N_{\text{box}} \times 4)$ bytes, which can quickly fill disk space. Note that these file format use single precision floating point numbers.

3.2.3. Numerical Accuracy

Numerical accuracy depends on the residual tolerance specified for the numerical solution at runtime and the solvation box physical size and grid spacing. In most cases, you will need to test these parameters to ensure you have the accuracy required. As a rough guide, the numerical error in the solvation free energy is related to the tolerance by

$$\varepsilon_{\Delta G_{\text{solv}}} \approx 10 \times \text{tolerance}. \tag{3.18}$$

Molecular dynamics [41], minimization and trajectory post-processing [74] have different requirements for the maximum residual tolerance. Molecular dynamics does well with a tolerance of 10^{-5} and `npropagate=5`. Minimization requires tolerances of 10^{-11} or lower and is typically limited to `drms` $\geq 10^{-4}$. Trajectory post-processing for MM/RISM should use enough digits to obtain the necessary accuracy when differences in solvation free energy are computed. For example, if a error $< 0.2\,\text{kcal/mol}$ is required for $\Delta\Delta G_{\text{solv}}$, then ΔG_{solv} should be computed with an absolute error of $0.1\,\text{kcal/mol}$. The relative error required to achieve this depends on the magnitude of ΔG_{solv}.

Almost all applications should use a grid spacing of 0.3 to 0.5 Å or smaller. A larger grid spacing quickly leads to severe errors in thermodynamic quantities. Smaller grid spacing may be necessary for some applications (e.g., mapping potentials of mean force).

The size of the solvation box can be set in a number of ways; e.g., setting the box size directly, setting a buffer distance between the solute and the edges of the solvent box or should typically be at least 14 Å for water or larger for ionic solutions. The solvation box size should be increased until the thermodynamic properties converge (see Section **??**). Systems with a neutral solute or non-ionic solvent are the simplest case, as solvent box size associated errors are primarily due to the truncation of the Lennard-Jones potential. Fortunately, this error can be corrected for if a cutoff is applied and the cutoff does not extend beyond the solvent box. In general, when using this correction, a cutoff where

$$u_{\alpha}^{\text{LJ}}(r_{\text{cut}}) \leq \text{tolerance}/10 \tag{3.19}$$

does not affect numerical precision of the calculation. Since long range Coulomb interactions are handled analytically by the long range asymptotics functions [53, 74], the solvent box size can be determined by the cutoff distance in many cases, which is calculated from the maximum error in the Lennard-Jones calculation and is determined at run time by the combination of `ljTolerance`, `tolerance`, `buffer`, and `solvbox` values used. The behavior is summarized in Table 3.1 on page 21.

For calculations with charged solutes in ionic solvent, the absolute size of the box required for sufficient numerical accuracy will depend on the absolute charge of the concentration of ions. Generally, lower ion concentrations require larger solvent boxes. Here, we recommend experimenting with different buffer sizes and setting the Lennard-Jones tolerance according to Eq. (3.19).

Independent of solvent-box size and grid spacing, time can be saved by truncating the reciprocal space expressions for the long range asymptotics. In general, a cutoff where

$$\hat{c}_{\alpha}^{(as)}(k_{\text{cut}}) \leq \text{tolerance}/10 \tag{3.20}$$

does not affect numerical precision of the calculation. The cutoff in reciprocal space is determined by `asympKSpaceTolerance`.

For solutes with more than 1000 atoms, it becomes beneficial to replace the direct sum, real-space calculations of the Coulomb and long-range asymptotic interactions with treecode fast summation. Table 3.2 contains suggested parameter choices for treecode summation based off experience. Some calculated values are more sensitive than others, so we recommend experimenting with these settings for your system.

3.2.4. Solvation Free Energy Corrections

3D-RISM with HNC-like closures is known to overestimate the non-polar component of the solvation free energy. Several alternate expressions for the solvation free energy have been developed to correct this and are

buffer	ljTolerance < 0	ljTolerance 0	ljTolerance > 0
3*[-5em] < 0	Fixed box size with dimensions of `solvbox`. LJ cutoff fit to box size and correction applied.	Fixed box size with dimensions of `solvbox`. No LJ cutoff or correction applied.	Fixed box size with dimensions of `solvbox`. LJ cutoff with `ljTolerance` applied. Correction applied if the box size is large enough.
0	`ljTolerance=tolerance/10` and the box size is selected to fit the cutoff. Correction applied.	Error.	Box size is selected to fit the cutoff. Correction applied if the box size is large enough.
> 0	Box size determined by `buffer`. LJ cutoff fit to box size and correction applied.	Box size determined by `buffer`. No LJ cutoff or correction applied.	Box size determined by `buffer`. Correction applied if the box size is large enough.

Table 3.1.: *The relationship between `ljTolerance`, `tolerance`, `buffer`, and `solvbox` in determining 3D-RISM solvent box and Lennard-Jones cutoff values.*

	treecodeMAC	treecodeOrder	treecodeN0
Total Correlation Function	0.3	$\max\left(2, \frac{\log_{10}(\text{tolerance})+5.7}{-0.7}\right)$	500
Direct Correlation Function	0.3	$\max\left(2, \frac{\log_{10}(\text{tolerance})+1.9}{-0.8}\right)$	500
Coulomb	0.3	$\max\left(2, \frac{\log_{10}(\text{tolerance})+1.4}{-0.8}\right)$	500

Table 3.2.: *Suggested 3D-RISM treecode parameters.*

based all, or in part, on the partial molar volume (PMV) of the solute. These include the Universal Correction (UC) [75], Ng Bridge Correction (NgB) [76] and the Pressure Correction Plus (PC+/3D-RISM) correction [77]. 3D-RISM currently implements UC and PV+/3D-RISM as runtime options. NgB results can be calculated from the standard thermodynamic output if the polarDecomp option is used but is not implemented directly. UC and NgB are both parameterized corrections. So, parameters for these corrections must be used only with the .xvv file used to create them. Our implementation of UC uses the excess chemical potential of the closure rather than the GF functional, as we have found this provides better results in general [78]. All of these corrections have been almost exclusively used with pure water under ambient conditions, though there are promising results for UC with non-polar liquids.[79] Using these methods with different solvents and co-solvents is a subject of on-going research.

3.3. Work Flow

Using 3D-RISM with SANDER for molecular dynamics, minimization or snapshot analysis is very similar to using implicit solvent models like GBSA. However, some additional preliminary setup is required, the extent of which depends on the solvent to be used.

3D-RISM requires detailed information of the bulk solvent in the form of the site-site susceptibility, χ^{VV}, and properties such as the temperature and partial charges. This is read in as an `.xvv` file, which is produced by a 1D-RISM calculation. If another 3D-RISM calculation is to be preformed with any details of the bulk solvent changed (e.g., temperature or pressure) a new `.xvv` file must be produced. Examples of precomputed `.xvv` files for SPC/E and TIP3P water can be found in `$AMBERHOME/AmberTools/test/rism1d`.

Special care must be taken when producing .xvv files for use with 3D-RISM, particularly with respect to grid parameters. It is important that the spatial extent of the grid be large enough to capture the essential long range features of the solvent while the spacing must be fine enough to sample the short-range structure. A grid spacing

of 0.025 Å is sufficient for most applications. The number of grid points required, which will determine the physical length of the grid in Å, generally depends on the properties of the solvent. Low concentration aqueous salt solutions typically require much larger grids than pure bulk water. A good indicator that the grid is large enough is convergence of delhv0 in the .xvv file. When converged, delhv0 should retain four to five digits of precision when the number of grid points is doubled.

The ability of 3D-RISM to perform temperature derivatives and calculate solvation energy and entropy requires `.xvv` files with with temperature dependence information. `rism1d` must be run with entropicDecomp option turned on (Section **??**). The version number in the `.xvv` file header indicates the maximum information available. Version 1.001 (current) allows temperature derivatives and solvation entropies and energies for all reported quantities. Version 1.000 (since Amber12) does not allow temperature derivatives of the PMV or solventation energies and entropies of PMV-based corrections. Version 0.001 does not have information for any temperature derivatives.

1D-RISM calculations require details of the some bulk properties of the solvent, such as temperature and dielectric constant, and an explicit model of the molecular components. These are read in from one or more `.mdl` files, depending on the composition of the solvent. Several `.mdl` files are included in the Amber11 distribution and can be found in `$AMBERHOME/dat/rism1d/model`. These include many of the explicit models for solvent and ions used with the Amber force fields. Other solvents models may be used by creating appropriate MDL files. See Section **??** for format details.

3.3.1. Solution Convergence

The default parameters for 3D-RISM are selected to provide the best performance for the majority of systems. In cases where a convergence is not achieved, the strategies below may be useful.

3.3.1.1. Closure Bootstrap

When a PSE-*n* or HNC closure is desired, the most effective method to overcome convergence issues is to use a low order closure solution as a starting guess. The KH closure should be the starting point as it is numerically robust and, typically, converges easily in the vast majority of case. After this, higher orders of PSE-*n* can be used until the desired closure is reached. The procedure for 1D-RISM and 3D-RISM differs slightly in practice.

1D-RISM `rism1d` can use restart files to implement this approach (see Section Subsection **??**). First, run `rism1d` with the KH closure to convergence. Then use the .sav file as input for the next highest closure. The root name of the .sav file must be the same as your .inp file. To avoid overwriting lower order solutions, name the files by closure or use separate directories. You will have to rename the .sav files as you go.

3D-RISM All 3D-RISM interfaces have closure bootstrapping builtin via the *closure* and *tolerance* keywords. Closures should be specified as an ordered list with last closure being the highest order closure. The solutions of the intermediate closures can have a high tolerance. The default tolerance for intermediate closures is 1 and there is no observed benefit to tolerances less than 1e-2. See details in Subsection **??**, Subsection 3.4.1.1 and Section **??**.

3.3.1.2. MDIIS Settings

MDIIS default setting are appropriate for most cases. Should your residual diverge or the solver get stuck on a particular value, you can try modest adjustments.

Decrease *mdiis_del* *mdiis_del* controls the step size of MDIIS. A smaller step size can help convergence but if this is set too small it can cause convergence problems. For `rism1d`, this should be no lower than 0.1 or 0.2. For 3D-RISM, it should be 0.5 at the lowest.

Increase *mdiis_nvec* This is the number of trial solutions that are saved for predicting a new solution. The optimal number for rapid convergence is typically 10 for 3D-RISM and 20 for 1D-RISM. However, for 3D-RISM, the default choice of 5 requires much less memory and is computationally faster even though more iterations are required. Increasing the *mdiis_nvec* may help for 3D-RISM but is unlikely to help for 1D-RISM.

Increase *mdiis_restart* Occasionally, the MDIIS routine goes in the wrong direction and the residual increases significantly. If it increases more than *mdiis_restart* then the MDIIS routine selects the solution with the lowest residual and purges the other trial solutions. The default value of 10 can be too aggressive and cause the solver to cycle. Increasing the value to 100 or 1000 sometimes allows the solver to recover from a misstep.

3.3.1.3. Parameter Annealing

Chargeless, hot gases are the easiest systems to converge. For 1D-RISM, this can be used to bootstrap a solution in a similar manner to closure bootstrapping. By slowly turning on charges, lowering the temperature or increasing the density, a converged solution may be reached. This only works for 1D-RISM because it requires restarting from a previous solution. As with closure bootstrapping, files should be carefully renamed during the procedure. There is no general protocol but the parameter increment should be reduced as the target value is approached. E.g., turning on charges in a linear fashion usually isn't helpful.

3.3.1.4. Forcefield selection

The forcefield may affect convergence due to the number of solvent sites involved or the particular parameters of the forcefield.

Number of Sites Molecules with more sites are more difficult to converge. Six or more sites is already difficult to converge and more that 10 may not be possible under any circumstances. One solution is to use a united atom or coarse grained forcefields to reduce the number of sites.

Alternate Parameterization Some parameter sets simply yield a stiffer set of equations to solve. Choosing an alternate parameter set may allow convergence with only small differences in the numerical results. For example, the cSPC/E water model with SPC/E Joung/Cheatham ions is easier to converge at higher ion concentrations in 1D-RISM than cTIP3P water with TIP3P Joung/Cheatham ions. Both models give nearly identical results in RISM at lower concentrations but NaCl in cTIP3P water will not converge above 0.5 M for the PSE-3 closure despite using all of the above methods.

3.4. 3D-RISM in sander

3D-RISM functionality is available in *sander* and is built as part of the standard install procedure. MPI functionality for 3D-RISM in sander requires some additional information at compile time, described in Section **??**. Some features specific to *sander* are discussed here.

3.4.1. 3D-RISM in `sander`

Full 3D-RISM functionality is available in sander as part of the standard install procedure. However, some methods available in `sander` are not compatible with 3D-RISM, such as QM/MM simulations. At this time, only standard molecular dynamics, minimization and trajectory post-processing with non-polarizable force fields are supported. With the exception of multiple time step features, 3D-RISM keywords in sander are identical to those in NAB, rism3d.snglpnt and MMPBSA.py.

3D-RISM specific command line options for sander are

```
sander [standard options] -xvv xvvfile -guv guvroot -huv huvroot
    -cuv cuvroot -uuv uuvroot -asymp asympfile
    -quv quvroot -chgdist chgdistroot
    -exchem exchemroot -solvene solveneroot -entropy entropyroot -potUV potUVroot
```

xvvfile *input* description of bulk solvent properties, required for 3D-RISM calculations. Produced by `rism1d`.

guvroot *output* root name for solute-solvent 3D pair distribution function, $G^{\mathrm{UV}}(\mathbf{R})$. This will produce one file for each solvent atom type for each frame requested.

huvroot *output* root name for solute-solvent 3D total correlation function, $H^{\mathrm{UV}}(\mathbf{R})$. This will produce one file for each solvent atom type for each frame requested.

cuvroot *output* root name for solute-solvent 3D total correlation function, $C^{\mathrm{UV}}(\mathbf{R})$. This will produce one file for each solvent atom type for each frame requested.

uuvroot *output* root name for solute-solvent 3D potential energy function, $U^{\mathrm{UV}}(\mathbf{R})$, in units of kT. This will produce one file for each solvent atom type for each frame requested.

asympfile *output* root name for solute-solvent 3D long-range real-space asymptotics for C and H. This will produce one file for each of C and H for each frame requested and does not include the solvent site charge. Multiply the distribution by the solvent site charge to obtain the long-range asymptotics for that site.

quvroot *output* root name for solute-solvent 3D charge density distribution [e/Å]. This will produce one file that combines contributions from all solvent atom types for each frame requested.

chgdistroot *output* root name for solute-solvent 3D charge distribution [e]. This will produce one file that combines contributions from all solvent atom types for each frame requested.

exchemroot *output* root name for 3D excess chemical potential distribution files.

solveneroot *output* root name for 3D solvation energy distribution files.

entropyroot *output* root name for 3D solvation entropy distribution files.

potUVroot *output* root name for 3D solute-solvent potential energy distribution files.

Generated output files can be large and numerous. For each type of correlation, a separate file is produced for each solvent atom type. The frequency that files are produced is controlled by the `ntwrism` parameter. Every time step that output is produced, a new set of files is written with the time step number in the file name. For example, a molecular dynamics calculation using an SPC/E water model with `ntwrism=2` and `-guv guv` on the command line will produce two files on time step ten: `guv.O.10.mrc` and `guv.H1.10.mrc`.

3.4.1.1. Keywords

With the exception of irism, which is found in the &cntrl name list, all 3D-RISM options are specified in the &rism name list.

`irism` [0] Use 3D-RISM. Found in `&cntrl` name list.

= 0 Off.

= 1 On.

Closure Approximation

`closure` [KH] Comma separate list of closure approximations. If more than one closure is provided, the 3D-RISM solver will use the closures in order to obtain a solution for the last closure in the list when no previous solutions are available. The solution for the last closure in the list is used for all output.

= KH Kovalenko-Hirata (KH).

= HNC Hyper-netted chain equation (HNC).

=PSE*n* Partial series expansion of order-n (PSE-n), where "n" is a positive integer.

Solvation Free Energy Corrections

`gfCorrection` [0] Compute the Gaussian fluctuation excess chemical potential functional (see §3.1.2).

= 0 Off.

= 1 On.

`pcpluscorrection` [0] Compute the PC+/3D-RISM excess chemical potential functional (see §3.2.4).

= 0 Off.

= 1 On.

`uccoeff` [0,0,0,0] Compute the UC excess chemical potential functional with the provided coefficients (see §3.2.4). *a* and *b* are the coefficients for the original UC functional, though using the closure excess chemical potential functional. *a1* and *b1* are optional and provide temperature dependence to the correction (UCT in [78]).

Long-range asymptotics Long-range asymptotics are used to analytically account for solvent distribution beyond the solvent box. Long-range asymptotics are always used when calculating a solution but can be omitted for the subsequent thermodynamic calculations, though it is not recommended.

`asympcorr` [T] Use long-range asymptotic corrections for thermodynamic calculations.

= T Use the long-range corrections.

= F Do not use long-range corrections.

`treeDCF` [1] Use direct sum or the treecode approximation to calculate the direct correlation function long-range asymptotic correction.

0 Use direct sum.

1 Use treecode approximation.

`treeTCF` [1] Use direct sum or the treecode approximation to calculate the total correlation function long-range asymptotic correction.

0 Use direct sum.

1 Use treecode approximation.

`treeCoulomb` [0] Use direct sum or the treecode approximation to calculate the Coulomb potential energy.

0 Use direct sum.

1 Use treecode approximation.

`treeDCFMAC` [0.1] Treecode multipole acceptance criterion for the direct correlation function long-range asymptotic correction.

`treeTCFMAC` [0.1] Treecode multipole acceptance criterion for the total correlation function long-range asymptotic correction.

`treeCoulombMAC` [0.1] Treecode multipole acceptance criterion for the Coulomb potential energy.

`treeDCFOrder` [2] Treecode Taylor series order for the direct correlation function long-range asymptotic correction.

`treeTCFOrder` [2] Treecode Taylor series order for the total correlation function long-range asymptotic correction. Note that the Taylor expansion used does not converge exactly to the TCF long-range asymptotic correction, so a very high order will not necessarily increase accuracy.

`treeCoulombOrder` [2] Treecode Taylor series order for the Coulomb potential energy.

`treeDCFN0` [500] Maximum number of grid points contained within the treecode leaf clusters for the direct correlation function long-range asymptotic correction. This sets the depth of the hierarchical octtree.

`treeTCFN0` [500] Maximum number of grid points contained within the treecode leaf clusters for the total correlation function long-range asymptotic correction. This sets the depth of the hierarchical octtree.

`treeCoulombN0` [500] Maximum number of grid points contained within the treecode leaf clusters for the Coulomb potential energy. This sets the depth of the hierarchical octtree.

Solvation Box The non-periodic solvation box super-cell can be defined as variable or fixed in size. When a variable box size is used, the box size will be adjusted to maintain a minimum buffer distance between the atoms of the solute and the box boundary. This has the advantage of maintaining the smallest possible box size while adapting to changes of solute shape and orientation. Alternatively, the box size can be specified at run-time. This box size will be used for the duration of the sander calculation.

Solvent box dimensions have a strong effect on the numerical precision of 3D-RISM. See Subsection 3.2.3 for recommendation on selecting an appropriate box size and resolution.

`solvcut` [`buffer`] Sets Lennard-Jones cutoff distance for periodic calculations. If '-1' or no value is specified then the buffer distance is used.

Variable Box Size

`buffer` [14] Minimum distance in Å between the solute and the edge of the solvent box. See §3.2.3 for details on how this affects numerical accuracy and how this interacts with `ljTolerance`, and `tolerance`.

< 0 Use fixed box size (`ng3` and `solvbox`).

>= 0 Buffer distance.

`grdspc` [0.5,0.5,0.5] Linear grid spacing in Å.

Fixed Box Size

`ng3` [] Sets the number of grid points for a fixed size solvation box. This is only used if buffer< 0.

`nx,ny,nz` Points for x, y and z dimensions.

`solvbox` [] Sets the size in Å of the fixed size solvation box. This is only used if buffer< 0. See §3.2.3 for details on how this affects numerical accuracy and how this interacts with `ljTolerance`, and `tolerance`.

`lx,ly,lz` Box length in x, y and z dimensions.

Solution Convergence

`tolerance` [1e-5] A list of maximum residual values for solution convergence. When used in combination with a list of closures it is possible to define different tolerances for each of the closures. This can be useful for difficult to converge calculations (see Subsection **??** for details). For the sake of efficiency, it is best to use as high a tolerance as possible for all but the last closure. For minimization a tolerance of 1e-11 or lower is recommended. See §3.2.3 for details on how this affects numerical accuracy and how this interacts with `ljTolerance`, `buffer`, and `solvbox`. Three formats of list are possible.

`one tolerance` All closures but the last use a tolerance of 1. The last tolerance in the list is used by the last closure. In practice this, is the most efficient.

`two tolerances` All closures but the last use the first tolerance in the list. The last tolerance in the list is used by the last closure.

n `tolerances` Tolerances from the list are assigned to the closure list in order.

`ljTolerance` [-1] Determines the Lennard-Jones cutoff distance based on the desired accuracy of the calculation. See §3.2.3 for details on how this affects numerical accuracy and how this interacts with `tolerance`, `buffer`, and `solvbox`.

`asympKSpaceTolerance` [-1] Determines the reciprocal space long range asymptotics cutoff distance based on the desired accuracy of the calculation. See §3.2.3 for details on how this affects numerical accuracy. Possible values are

< 0 `asympKSpaceTolerance=tolerance/10`,

0 no cutoff, and

> 0 given value determines the maximum error in the reciprocal-space long range asymptotics calculations.

`mdiis_del` [0.7] "Step size" in MDIIS.

`mdiis_nvec` [5] Number of vectors used by the MDIIS method. Higher values for this parameter can greatly increase memory requirements but may also accelerate convergence.

`mdiis_restart` [10] If the current residual is `mdiis_restart` times larger than the smallest residual in memory, then the MDIIS procedure is restarted using the lowest residual solution stored in memory. Increasing this number can sometimes help convergence.

`mdiis_method` [2] Specify implementation of the MDIIS routine.

= 0 Original. For small systems (e.g. $< 64^3$ grid points) this implementation may be faster than the BLAS optimized version.

= 1 BLAS optimized.

= 2 BLAS and memory optimized.

`maxstep` [10000] Maximum number of iterations allowed to converge on a solution.`nrespa`

`npropagate` [5] Number of previous solutions propagated forward to create an initial guess for this solute atom configuration.

= 0 Do not use any previous solutions

= 1..5 Values greater than 0 but less than 4 or 5 will use less system memory but may introduce artifacts to the solution (e.g., energy drift).

Minimization and Molecular Dynamics

`centering` [1] Controls how the solute is centered/re-centered in the solvent box.

= -4 Center-of-geometry with grid-point rounding. Center on first step only.

= -3 Center-of-mass with grid-point rounding. Center on first step only.

= -2 Center-of-geometry. Center on first step only.

= -1 Center-of-mass. Center on first step only.

= 0 No centering. Dangerous.

= 1 Center-of-mass. Center on every step. Recommended for molecular dynamics.

= 2 Center-of-geometry. Center on every step. Recommended for minimization.

= 3 Center-of-mass with grid-point rounding.

= 4 Center-of-geometry with grid-point rounding.

zerofrc [1] Redistribute solvent forces across the solute such that the net solvation force on the solute is zero.

= 0 Unmodified forces.

= 1 Zero net force.

Trajectory Post-Processing

apply_rism_force [1] Calculate and use solvation forces from 3D-RISM. Not calculating these forces can save computation time and is useful for trajectory post-processing.

= 0 Do not calculate forces.

= 1 Calculate forces.

Output

ntwrism [0] Indicates that solvent density grid should be written to file every `ntwrism` iterations.

= 0 No files written.

>= 1 Output every `ntwrism` time steps.

`molReconstruction` [0] For any thermodynamic distributions requested, also out the molecular reconstruction (see section **??**).

`volfmt` ['mrc'] Format of volumetric data files. May be mrc, ccp4, dx or xyzv (see section **??**).

`verbose` [0] Indicates level of diagnostic detail about the calculation written to the log file.

= 0 No output.

= 1 Print the number of iterations used to converge.

= 2 Print details for each iteration and information about what FCE is doing every `progress` iterations.

write_thermo [1] Print solvation thermodynamics in addition to standard sander output. The format is the same as that found in NAB and rism3d.snglpnt.

polarDecomp [0] Decomposes solvation free energy into polar and non-polar components. Note that this typically requires 80% more computation time.

= 0 No polar/non-polar decomposition.

= 1 Polar/non-polar decomposition.

entropicDecomp [0] Decomposes solvation free energy into energy and entropy components. Also performs temperature derivatives of other calculated quantities. Note that this typically requires 80% more computation time and requires a .xvv file version 1.000 or higher (see §**??** and 3.3).

= 0 No entropic decomposition.

= 1 Entropic decomposition.

`progress` [1] Display progress of the 3D-RISM solution every `kshow` iterations. 0 indicates this information will not be displayed. Must be used with `verbose` > 1.

3.4.1.2. Example

Molecular Dynamics (`imin=0`)

```
molecular dynamics with 3D-RISM and impulse MTS
&cntrl
    ntx=1, ntpr=100, ntwx=1000,ntwr=10000,
    nstlim=10000,dt=0.001,                !No shake or r-RESPA
    ntt=3, temp0=300, gamma_ln=20,        !Langevin dynamics
    ntb=0,                                !Non-periodic
    cut=999.,                             !Calculate all
                                          !solute-solute
                                          !interactions
    irism=1,
/
&rism
    rismnrespa=5,                         !r-RESPA MTS
    fcenbasis=10,fcestride=2,fcecrd=2     !FCE MTS
/
```

Minimization (`imin=1`)

```
Default XMIN minimization with 3D-RISM
&cntrl
    imin=1, maxcyc=200,
    drms=1e-3,          !RMS force. Can be as low as 1e-4
    ntmin=3,            !XMIN
    ntpr=5,
    ntb=0,              !Non-periodic
    cut=999.,           !Calculate all
                        !solute-solute interactions
    irism=1
/
&rism
    tolerance=1e-11,    !Low tolerance
    solvcut=9999,       !No cut-off for
                        !solute-solvent interactions
    centering=2         !Solvation box centering
                        !using center-of-geometry
/
```

Trajectory Post-Processing (`imin=5`)

```
Trajectory post-processing with 3D-RISM
&cntrl
    ntx=1, ntpr=1, ntwx=1,
    imin=5,maxcyc=1,       !Single-point energy calculation
                           !on each frame
    ntb=0,                 !Non-periodic
    cut=9999.,             !Calculate all
                           !solute-solute interactions
    irism=1
/
&rism
```

```
        tolerance=1e-4,          !Saves some time compared to 1e-5
        apply_rism_force=0,      !Saves some time. Forces are not used.
        npropagate=1             !Saves some time and 4*8*Nbox bytes
                                 !of memory compared to npropagate=5.
    /
```

4. sqm: Semi-empirical quantum chemistry

AmberTools contains its own quantum chemistry program, called *sqm*. This is code extracted from the QM/MM portions of *sander*, but is limited to "pure QM" calculations. A principal current use is as a replacement for MOPAC for deriving AM1-bcc charges, but the code is much more general than that. Presently, it is limited to single point calculations and energy minimizations (geometry optimizations) for closed-shell systems. It supports a wide variety of semi-empirical Hamiltonians, including many recent ones. An external electric field generated by a set of point charges can be included for single point calculations. Our plan is to add capabilities to subsequent versions. The major contributors are as follows:

- The original semi-empirical support was written by Ross Walker, Mike Crowley, and Dave Case,[80] based on public-domain MOPAC codes of J.J.P. Stewart.
- DFTB2 (SCC-DFTB) support was written by Gustavo Seabra, Ross Walker and Adrian Roitberg,[81] and is based on earlier work of Marcus Elstner.[82, 83]
- Support for diagonal third-order corrections to SCC-DFTB was written by Gustavo Seabra and Josh Mcclellan.
- DFTB3 was added by Andreas Goetz.
- Various SCF convergence schemes were added by Tim Giese and Darrin York.
- The PM6 Hamiltonian was added by Andreas Goetz and dispersion and hydrogen bond corrections were added by Andreas Goetz and Kyoyeon Park.
- The extension for MNDO type Hamiltonians to support d orbitals was written by Tai-Sung Lee, Darrin York and Andreas Goetz.
- The charge-dependent exchange-dispersion corrections of vdW interactions[84] was contributed by Tai-Sung Lee, Tim Giese, and Darrin York.
- Support for reading user-defined parameters for NDDO methods was added by Tai-Sung Lee and Darrin York.

The DFTB/DFTB2 code was originally based on the DFT/DYLAX code by Marcus Elstner *et al.*, but has since been extensively re-written and optimized. The DFTB3 implementation is an extension of this code.

4.1. Available Hamiltonians

Available MNDO-type semi-empirical Hamiltonians are PM3,[85] AM1,[86] RM1,[87] MNDO,[88] PDDG/PM3,[89] PDDG/MNDO,[89] PM3CARB1,[90], PM3-MAIS[91, 92], MNDO/d[93–95], AM1/d (Mg from AM1/d[96] and H, O, and P from AM1/d-PhoT[97]) and PM6[98].

Also available is the density functional theory-based tight-binding (DFTB) Hamiltonian[81, 99, 100] and its self-consistent-charge version with Taylor expansion up to second order (SCC-DFTB or DFTB2)[82] and third-order (DFTB3)[101]. If you use the mio-1-1 parameters for DFTB2, you can add an empirical correction for dispersion effects[102] and calculate CM3 charges[103] (both only for elements H, C, N, O, S, P). Diagonal third-order corrections are available for DFTB2[104] with mio-1-1 parameters but it is recommended to perform full DFTB3 simulations instead. Neither dispersion corrections nor halogen corrections are implemented for DFTB3.

The elements supported by each QM method are:

- MNDO: H, Li, Be, B, C, N, O, F, Al, Si, P, S, Cl, Zn, Ge, Br, Cd, Sn, I, Hg, Pb
- MNDO/d: H, Li, Be, B, C, N, O, F, Na, Mg, Al, Si, P, S, Cl, Zn, Ge, Br, Sn, I, Hg, Pb
- AM1: H, C, N, O, F, Al, Si, P, S, Cl, Zn, Ge, Br, I, Hg
- AM1/d: H, C, N, O, F, Mg, Al, Si, P, S, Cl, Zn, Ge, Br, I, Hg
- PM3: H, Be, C, N, O, F, Mg, Al, Si, P, S, Cl, Zn, Ga, Ge, As, Se, Br, Cd, In, Sn, Sb, Te, I, Hg, Tl, Pb, Bi
- PDDG/PM3: H, C, N, O, F, Si, P, S, Cl, Br, I
- PDDG/MNDO: H, C, N, O, F, Cl, Br, I
- RM1: H, C, N, O, P, S, F, Cl, Br, I
- PM3CARB1: H, C, O
- PM3-MAIS: H, O, Cl
- PM6: H, He, Li, Be, B, C, N, O, F, Ne, Na, Mg, Al, Si, P, S, Cl, Ar, K, Ca, Sc, Ti, V, Cr, Mn, Fe, Co, Ni, Cu, Zn, Ga, Ge, As, Se, Br, Kr, Rb, Sr, Y, Zr, Nb, Mo, Tc, Ru, Rh, Pd, Ag, Cd, In, Sn, Sb, Te, I, Xe, Cs, Ba, La, Lu, Hf, Ta, W, Re, Os, Ir, Pt, Au, Hg, Tl, Pb, Bi
- DFTB/DFTB2/DFTB3: (Any atoms for which parameters are available from *www.dftb.org*)

The PM6 implementation has not been extensively tested for all available elements. Please check your results carefully, possibly by comparison to other codes that implement PM6, if transition metal elements are present. SCF convergence may be more difficult to achieve for transition metal elements with partially filled valence shells.

If the PM6 Hamiltonian is used in a QM/MM simulation with *sander* using electrostatic embedding (see Section 6) or if an electric field of external point charges is used, then the electrostatic interactions between QM and MM atoms are modeled using the MNDO type core repulsion function for interactions between QM and MM atoms. Parameters for the exponents α of the QM atoms are taken from PM3 (a default value of five is used for the exponents α of the MM atoms as is the case for MNDO, AM1 and PM3). Since PM3 does not have parameters for all elements that are supported by PM6, the missing exponents were defined in an ad hoc manner (see the source code in $AMBERHOME/AmberToosl/src/sqm/qm2_parameters.F90, variable alp_pm6). The magnitude of the coefficients α is probably not critical for the accuracy of QM/MM calculations but this should be tested on a case by case basis. This does not affect QM calculations with *sqm*.

4.1.1. DFTB parameter files

In order to use DFTB2 or DFTB3 (*qm_theory=DFTB2* or *DFTB3*) a set of integral parameter files is required. The mio-1-1 parameter files for DFTB2 and 3ob-3-1 parameter files are distributed with Amber under a Creative Commons Attribution-ShareAlike 4.0 International License, see http://creativecommons.org/licenses/by-sa/4.0/. The parameters were obtained from the website *www.dftb.*org on February 22, 2017. You may want to check if there are any updates to the parameters. If you perform DFTB simulations, in addition to Amber please cite the publications describing the QM/MM and DFTB implementations as well as following references for the DFTB parameters:

When using DFTB2 with mio-1-1 and following elements:

- O, N, C, H: M. Elstner, D. Porezag, G. Jungnickel, J. Elsner, M. Haugk, Th. Frauenheim, S. Suhai, G. Seifert, *Phys. Rev. B* **58** (1998) 7260.
- S: T. A. Niehaus, M. Elstner, Th. Fruanheim, S. Suhai, *J. Molec. Struct. (THEOCHEM)* **541** (2001) 185.
- P: M. Gaus, Q. Cui, M. Elstner, *J. Chem. Theory Comput.* **7** (2011) 931-948.

When using DFTB3 with 3ob-3-1 and following elements:

- O, N, C, H: M. Gaus, A. Goez, M. Elstner, *J. Chem. Theory Comput.* **9** (2013) 338-354.
- P, S: M. Gaus, X. Lu, M. Elstner, Q. Cui, *J. Chem. Theory Comput.* **10** (2014) 1518-1537.
- Mg, Zn: X. Lu, M. Gaus, M. Elstner, Q. Cui, *J. Phys. Chem. B* **119** (2015) 1062-1082.
- Na, F, K, Ca, Cl, Br, I: M. Kubillus, T. Kubar, M. Gaus, J. Rezac, M. Esltner, *J. Chem. Theory Comput.* **11** (2015) 332-342.

Additional parameter files can be obtained from the website *www.dftb.org*. By default it is assumed that DFTB2 uses the mio-1-1 parameter set and DFTB3 the 3ob-3-1 parameter set and that the corresponding files with extension *.skf* reside in the directories *$AMBERHOME/dat/slko/mio-1-1* and *$AMBERHOME/dat/slko/3ob-3-1*. If you want to use other parameter sets and/or put the parameter files in other directories then you have to specify the location in the input file (keyword *dftb_slko_path,* see section 4.3 for details).

Following parameter files for use with DFTB2 and the mio-1-1 parameter set are also distributed with AmberTools: Dispersion parameters for H, C, N, O, P and S are available in the file *$AMBERHOME/dat/slko/mio-1-1/DISPERSION.INP_ONCHSP*, CM3 parameters for the same atoms are in the file *$AMBERHOME/dat/slko/mio-1-1/CM3_PARAMETERS.DAT* file, and two parametrizations for diagonal third-order SCC-DFTB terms (SCC-DFTB-PA and SCC-DFTB-PR) are in the files *DFTB_3RD_ORDER_PA.DAT* and *DFTB_3RD_ORDER_PR.DAT*, both located in the same directory.

4.2. Dispersion and hydrogen bond correction

An empirical dispersion and hydrogen bonding correction is implemented for the MNDO type Hamiltonians AM1 and PM6[105]. The empirical dispersion correction follows the formalism for DFT-D[106] and consists of a physically sound r^{-6} term that is damped at short distances to avoid the short-range repulsion which can be written as

$$E_{\mathrm{dis}} = -s_6 \sum_{ij} f_{\mathrm{d}amp}(r_{ij}, R^0_{ij}) C_{6,ij} r_{ij}^{-6}, \tag{4.1}$$

where r_{ij} is the distance between two atoms i and j, R^0_{ij} is the equilibrium van der Waals (vdW) separation derived from the atomic vdW radii, $C_{6,ij}$ the dispersion coefficient, and s_6 a general scaling factor. The damping function is given as

$$f_{\mathrm{d}amp}(r_{ij}, R^0_{ij}) = \left[1 + \exp\left(-\alpha \frac{r_{ij}}{s_R R^0_{ij}} - 1 \right) \right]^{-1}. \tag{4.2}$$

Bondi vdW radii[107] are used and for a pair of unlike atoms we have

$$R^0_{ij} = \frac{{R^0_{ii}}^3 + {R^0_{jj}}^3}{{R^0_{ii}}^2 + {R^0_{jj}}^2}. \tag{4.3}$$

For the C_6 coefficients the following equation is used,

$$C_{6,ij} = 2 \frac{(C^2_{6,ii} C^2_{6,jj} N_{\mathrm{e}ff,i} N_{\mathrm{e}ff,j})^{1/3}}{(C_{6,ii} N^2_{\mathrm{e}ff,j})^{1/3} + (C_{6,jj} N^2_{\mathrm{e}ff,i})^{1/3}}, \tag{4.4}$$

where the Slater-Kirkwood effective number of electrons $N_{\mathrm{e}ff,i}$ and the C_6 coefficients can easily be found in the literature[106].

An empirical hydrogen bonding correction[105] that is transferable among different semiempirical Hamiltonians and has been parametrized for use with the dispersion correction described above is also available. This correction does not make the assumption of a specific acceptor/hydrogen/donor binding situation. Instead it considers the hydrogen bond as a charge-independent atom- atom term between two atoms capable of serving as an acceptor or donor (for example, O, N) and weights this by a function that accounts for the steric arrangement of the two

atoms and the favorable positioning of a hydrogen atom inbetween. A damping function corrects for long- and short-range behavior,

$$E_{\mathrm{H}-bond} = \frac{C_{AB}}{r_{AB}^2} f_{geom} f_{\mathrm{d}amp}, \tag{4.5}$$

$$f_{geom} = \cos(\theta_A)^2 \cos(\phi_A)^2 \cos(\psi_A)^2 \cos(\phi_B)^2 \cos(\phi_B)^2 \cos(\psi_B)^2 f_{\mathrm{bond}}, \tag{4.6}$$

$$f_{\mathrm{bond}} = 1 - \frac{1}{1+\exp[-60(r_{\mathrm{XH}}/1.2-1)]}, \tag{4.7}$$

$$f_{\mathrm{d}amp} = \left(\frac{1}{1+\exp[-100(r_{AB}/2.4-1)]}\right)\left(1 - \frac{1}{1+\exp[-10(r_{\mathrm{AB}}/7.0-1)]}\right), \tag{4.8}$$

$$C_{AB} = \frac{C_{\mathrm{A}} + C_{\mathrm{B}}}{2}. \tag{4.9}$$

Here, C_{A} and C_{B} are the atomic hydrogen bonding correction parameters and the (torsion) angles in the function f_{geom} are defined similarly to an earlier hydrogen bond correction[108].

The hydrogen bond correction can be used both for single point energy calculations or geometry optimizations with SQM and for molecular dynamics simulations with SANDER. However, we do not recommend the use for molecular dynamics at present since cutoffs needed to be implemented for the calculation of f_{geom} of equation (4.6). This and some other conditional evaluations give rise to discontinuities in the potential energy surface and thus make this method unattractive for MD simulations.

4.3. Usage

The *sqm* program uses the following simple command line:

```
sqm [-O] -i <input-file> -o <output-file>
```

mdin is the default input-file name, and *mdout* is the default output-file name.As in other Amber programs, the "-O" flag allows the program to over-write the output file.

An example input file for running a simple minimization is shown here:

```
Run semi-empirical minimization
 &qmmm
  qm_theory='AM1',   qmcharge=0,
 /
    6    CG       -1.9590         0.1020         0.7950
    6   CD1       -1.2490         0.6020        -0.3030
    6   CD2       -2.0710         0.8650         1.9630
    6   CE1       -0.6460         1.8630        -0.2340
    6    C6       -1.4720         2.1290         2.0310
    6    CZ       -0.7590         2.6270         0.9340
    1   HE2       -1.5580         2.7190         2.9310
   16   S15       -2.7820         0.3650         3.0600
    1   H19       -3.5410         0.9790         3.2740
    1   H29       -0.7870        -0.0430        -0.9380
    1   H30        0.3730         2.0450        -0.7840
    1   H31       -0.0920         3.5780         0.7810
    1   H32       -2.3790        -0.9160         0.9010
```

The *&qmmm* namelist contains variables that allow you to control the options used. Following that is one line per atom, giving the atomic number, atom name, and Cartesian coordinates (free format). The variables in the *&qmmm* namelist are these:

`qm_theory` Level of theory to use for the QM region of the simulation (Hamiltonian). Default is to use the semi-empirical Hamiltonian PM3. Options are AM1, RM1, MNDO, PM3-PDDG, MNDO-PDDG,

PM3-CARB1, MNDO/d (same as MNDOD), AM1/d (same as AM1D), PM6, DFTB2 (same as DFTB), and DFTB3. The dispersion correction can be switched on for AM1 and PM6 by choosing AM1-D* and PM6-D, respectively. The dispersion and hydrogen bond correction will be applied for AM1-DH+ and PM6-DH+.

`dftb_slko_path` Path to the DFTB Slater-Koster parameter files. Defaults to '$AMBERHOME/dat/slko/mio-1-1/' for DFTB2 and '$AMBERHOME/dat/slko/3ob-3-1/' for DFTB3. You can specify a different directory here, which is assumed to be a subdirectory of '$AMBERHOME/dat/slko/' unless you specify an absolute path.

`dftb_disper` Flag turning on (1) or off (0) the use of a dispersion correction to the DFTB2 energy (only for mio-1-1 parameters). Requires *qm_theory=DFTB2*. It is assumed that you have the file DISPERSION.INP_ONCHSP in your $AMBERHOME/dat/slko/mio-1-1 directory. This file must be downloaded from the website www.dftb.org, as described in the beginning of this chapter. Only available for elements H, C, O, N, P, S. (Default = 0)

`dftb_3rd_order` Third order diagonal corrections to DFTB2 with mio-1-1 parameters. Default=" (the empty string which means no third order correction).

= 'PA' Use the SCC-DFTB-PA parametrization, which was developed for proton affinities. The parameters will be read from the *$AMBERHOME/dat/slko/DFTB_3RD_ORDER_PA.DAT* file.

= 'PR' Use the SCC-DFTB-PR parametrization, which was developed for phosphate hydrolysis reactions. The parameters will be read from the *$AMBERHOME/dat/slko/DFTB_3RD_ORDER_PR.DAT* file.

= 'READ' Parameters will be read from the *mdin* file, in a separate "dftb_3rd_order" namelist, which must have the same format as the files above.

= '*filename*' Parameters will be read from the file specified by *filename*, in the "dftb_3rd_order" namelist, which must have the same format as the files above.

`dftb_chg` Flag to choose the type of charges to report when doing a DFTB calculation.

= 0 (default) - Print Mulliken charges.

= 2 Print CM3 charges. Only available for DFTB2 with mio-1-1 parameters for elements H, C, N, O, S and P.

`dftb_telec` Electronic temperature, in K, used to accelerate SCC convergence in DFTB calculations. The electronic temperature affects the Fermi distribution promoting some HOMO/LUMO mixing, which can accelerate the convergence in difficult cases. In most cases, a low *telec* (around 100K) is enough. Should be used only when necessary, and the results checked carefully. Default: 0.0K

`dftb_maxiter` Maximum number of SCC iterations before resetting Broyden in DFTB calculations. (default: 70)

`qmcharge` Charge on the QM system in electron units (must be an integer). (Default = 0)

`spin` Multiplicity of the QM system. Currently only singlet calculations are possible and so the default value of 1 is the only available option. Note that this option is ignored by DFTB/SCC-DFTB, which allows only ground state calculations. In this case, the spin state will be calculated from the number of electrons and orbital occupancy.

`qmqmdx` Flag for whether to use analytical or numerical derivatives of the semiempirical electron repulsion integrals. The default (and recommended) option is to use ANALYTICAL QM-QM derivatives.

= 1 (default) - Use analytical derivatives for QM-QM forces.

= 2 Use numerical derivatives for QM-QM forces. Note: the numerical derivative code has not been optimised as aggressively as the analytical code and as such is significantly slower. Numerical derivatives are intended mainly for testing purposes.

`verbosity` Controls the verbosity of QM/MM related output. *Warning:* Values of 2 or higher will produce a *lot* of output.

= 0 (default) - only minimal information is printed - Initial QM geometry and link atom positions as well as the SCF energy at every ntpr steps.

= 1 Print SCF energy at every step to many more significant figures than usual. Also print the number of SCF cycles needed on each step.

= 2 As 1 and also print info about memory reallocations, number of pairs per QM atom, QM core - QM core energy, QM core - MM atom energy, and total energy.

= 3 As 2 and also print SCF convergence information at every step.

= 4 As 3 and also print forces on the QM atoms due to the SCF calculation and the coordinates of the link atoms at every step.

= 5 As 4 and also print all of the info in kJ/mol as well as kcal/mol.

`tight_p_conv` Controls the tightness of the convergence criteria on the density matrix in the SCF.

=0 (default) - loose convergence on the density matrix (or Mulliken charges, in case of a SCC-DFTB calculation). SCF will converge if the energy is converged to within scfconv and the largest change in the density matrix is within 0.05*sqrt(scfconv).

= 1 Tight convergence on density(or Mulliken charges, in case of a SCC-DFTB calculation). Use same convergence (scfconv) for both energy and density (charges) in SCF. Note: in the SCC-DFTB case, this option can lead to instabilities.

`scfconv` Controls the convergence criteria for the SCF calculation, in kcal/mol. In order to conserve energy in a dynamics simulation with no thermostat it is often necessary to use a convergence criterion of 1.0d-9 or tighter. Note, the tighter the convergence the longer the calculation will take. Values tighter than 1.0d-11 are not recommended as these can lead to oscillations in the SCF, due to limitations in machine precision, that can lead to convergence failures. Default is 1.0d-8 kcal/mol. Minimum usable value is 1.0d-14.

`pseudo_diag` Controls the use of 'fast' pseudo diagonalisations in the SCF routine. By default the code will attempt to do pseudo diagonalisations whenever possible. However, if you experience convergence problems then turning this option off may help. Not available for DFTB/SCC-DFTB.

= 0 Always do full diagonalisation.

= 1 Do pseudo diagonalisations when possible (default).

`pseudo_diag_criteria` Float controlling criteria used to determine if a pseudo diagonalisation can be done. If the difference in the largest density matrix element between two SCF iterations is less than this criteria then a pseudo diagonalisation can be done. This is really a tuning parameter designed for expert use only. Most users should have no cause to adjust this parameter. (Not applicable to DFTB/SCC-DFTB calculations.) Default = 0.05

`diag_routine` Controls which diagonalization routine will be used during the SCF procedure. This is an advanced option to fine-tune performance which has negligible effect on energies (and generally little effect on geometries in the case of SQM energy minimizations). The speed of each diagonalizer is a function of the number and type of QM atoms as well as the LAPACK library that the program was linked to. As such there is not always an obvious choice to obtain the best performance. The simplest option is to set diag_routine = 0 in which case the program will test each diagonalizer in turn, including the pseudo diagonalizer, and select the one that gives optimum performance. As of AmberTools 15 diag_routine = 0 is the default for both SQM and QMMM in Sander. Not available for DFTB/SCC-DFTB.

= 0 Automatically select the fastest routine (default).

= 1 Use internal diagonalization routine.

= 2 Use lapack dspev.

= 3 Use lapack dspevd.

= 4 Use lapack dspevx.

= 5 Use lapack dsyev.

= 6 Use lapack dsyevd.

= 7 Use lapack dsyevr.

`printcharges` **= 0** Don't print any info about QM atom charges to the output file (default)

= 1 Print Mulliken QM atom charges to output file every *ntpr* steps.

`print_eigenvalues` Controls printing of MO eigenvalues.

= 0 Do not print MO eigenvalues

= 1 Print MO eigenvalues at the end of a single point calculation or geometry optimization (default)

= 2 Print MO eigenvalues at the end of every SCF cycle (only NDDO methods, not DFTB)

= 3 Print MO eigenvalues during each step of the SCF cycle (only NDDO methods, not DFTB)

`qxd` Flag to turn on (=.true.) or off (=.false., default) the charge-dependent exchange-dispersion corrections of vdW interactions[84].

`parameter_file` **=** 'PARAM.FILE' Read user-defined parameters from the file 'PARAM.FILE'. The first three space-separated entries (case insensitive) of each line will be interpreted as a user-modified parameter in the sequence of *parameter name, element name,* and *value*. For example, a line contains "USS Cl -111.6139480D0 " will cause the USS parameter of the Cl element changed to -111.6139480. A line beginning with "END" will stop the reading. This function currently only works for MNDO, AM1, PM3, MNDO/d, and AM1/d. Also, when new nuclear core-core parameters (FN, in PM3, AM1, and AM1/d) are re-defined, the number of FNN parameter sets (NUM_FN) also needs to be defined. For example, if FNn3 (n = 1, 2, or 3) is defined, then NUM_FN needs to be set to 3 or 4.

`peptide_corr` **= 0** Don't apply MM correction to peptide linkages. (default)

= 1 Apply a MM correction to peptide linkages. This correction is of the form $E_{scf} = E_{scf} + h_{type}(i_{type}) \sin^2 \phi$, where ϕ is the dihedral angle of the H-N-C-O linkage and h_{type} is a constant dependent on the Hamiltonian used. (Recommended, except for DFTB/SCC-DFTB.)

`itrmax` Integer specifying the maximum number of SCF iterations to perform before assuming that convergence has failed. Default is 1000. Typically higher values will not do much good since if the SCF hasn't converged after 1000 steps it is unlikely to. If the convergence criteria have not been met after itrmax steps the SCF will stop and the minimisation will proceed with the gradient at itrmax. Hence if you have a system which does not converge well you can set itrmax smaller so less time is wasted before assuming the system won't converge. In this way you may be able to get out of a bad geometry quite quickly. Once in a better geometry SCF convergence should improve.

`maxcyc` Maximum number of minimization cycles to allow, using the *xmin* minimizer (see Section **??**) with the TNCG method. Default is 9999. Single point calculations can be done with *maxcyc* = 0.

`ntpr` Print the progress of the minimization every *ntpr* steps; default is 10.

`grms_tol` Terminate minimization when the gradient falls below this value; default is 0.02

`ndiis_attempts` Controls the number of iterations that DIIS (direct inversion of the iterative subspace) extrapolations will be attempted. Not available for DFTB/SCC-DFTB. The SCF does not even begin to exhaust its attempts at using DIIS extrapolations until the end of iteration 100. Therefore, for example, if ndiis_attempts=50, then DIIS extrapolations would be performed at end of iterations 100 to 150. The purpose of not performing DIIS extrapolations before iteration 100 is because the existing code base performs quite well for most molecules; however, if convergence is not met after 100 iterations, then it is presumed that further iterations will not yield SCF convergence without doing something different, i.e., DIIS. Thus, the implementation of DIIS in SQM is a mechanism to try and force SCF convergence for molecules that are otherwise difficult to converge. Default 0. Maximum 1000. Minimum 0. Note that DIIS will automatically turn itself on for 100 attempts at the end of iteration 800 even if you did not explicitly set ndiis_attempts to a nonzero value. This is done as a final effort to achieve convergence.

`ndiis_matrices` Controls the number of matrices used in the DIIS extrapolation. Including only one matrix is the same as not performing an extrapolation. Including an excessive number of matrices may require a large amount of memory. Not available for DFTB/SCC-DFTB. Default 6. Minimum 1. Maximum 20.

`vshift` Controls level shifting (only NDDO methods, not DFTB). Virtual orbitals can be shifted up by vshift (in eV) to improve SCF convergence in cases with small HOMO/LUMO gap. Default 0.0 (no level shift).

`errconv` SCF tolerance on the maximum absolute value of the error matrix, i.e., the commutator of the Fock matrix with the density matrix. The value has units of hartree. The default value of errconv is sufficiently large to effectively remove this tolerance from the SCF convergence criteria. Not available for DFTB/SCC-DFTB. Default 1.d-1. Minimum 1.d-16. Maximum 1.d0.

`qmmm_int` When running QM calculations in the sqm program, an electric field of external point charges can be added. In this way, the electrostatic effect outside of the QM region can be modeled, making the calculation a simplified QM/MM calculation without QM/MM vdW's contribution. Like QM/MM calculations (see Section 6), the method to couple QM and MM electrostatic interactions for external charges and semiempirical Hamiltonians can be specified via the *qmmm_int* namelist variable.

The current implementation limits use of external charges to only single point energy calculations. To run such a calculation, an additional field, which begins with #EXCHARGES and ends with #END, is required to specify the external point charges in the input. Each external point charge must include atomic number, atom name, X, Y, Z coordinates and the charge in units of the electron charge. An example input looks like:

```
single point energy calculation (adenine), with external charges (thymine)
&qmmm
 qm_theory = 'PM3',
 qmcharge = 0,
 maxcyc = 0,
 qmmm_int = 1,
/
7  N   1.0716177  -0.0765366   1.9391390
1  H   0.0586915  -0.0423765   2.0039181
1  H   1.6443796  -0.0347395   2.7619159
6  C   1.6739638  -0.0357766   0.7424316
7  N   0.9350155  -0.0279801  -0.3788916
6  C   1.5490760   0.0012569  -1.5808009
1  H   0.8794435   0.0050260  -2.4315709
```

```
7  N   2.8531510   0.0258031  -1.8409596
6  C   3.5646109   0.0195446  -0.7059872
6  C   3.0747955  -0.0094480   0.5994562
7  N   4.0885824  -0.0054429   1.5289786
6  C   5.1829921   0.0253971   0.7872176
1  H   6.1882591   0.0375542   1.1738824
7  N   4.9294871   0.0412404  -0.5567274
1  H   5.6035368   0.0648755  -1.3036811
#EXCHARGESwill be
6  C  -4.7106131   0.0413373   2.1738637  -0.03140
1  H  -4.4267056   0.9186178   2.7530256   0.06002
1  H  -4.4439282  -0.8302573   2.7695655   0.05964
1  H  -5.7883971   0.0505530   2.0247280   0.03694
6  C  -3.9917387   0.0219348   0.8663338  -0.25383
6  C  -4.6136833   0.0169051  -0.3336520   0.03789
1  H  -5.6909220   0.0269347  -0.4227183   0.16330
7  N  -3.9211729  -0.0009646  -1.5163659  -0.47122
1  H  -4.4017172  -0.0036078  -2.4004924   0.35466
6  C  -2.5395897  -0.0149474  -1.5962357   0.80253
8  O  -1.9416783  -0.0291878  -2.6573783  -0.63850
7  N  -1.9256484  -0.0110593  -0.3638948  -0.58423
1  H  -0.8838255  -0.0216168  -0.3784269   0.35404
6  C  -2.5361367   0.0074651   0.8766724   0.71625
8  O  -1.8674730   0.0112093   1.9120833  -0.60609
#END
```

5. QUICK: *ab initio* quantum chemistry

AmberTools now distributes the *QUICK* (QUantum Interaction Computational Kernel) *ab initio* quantum chemistry program.[109–113] *QUICK* is a GPU enabled *ab initio* and density functional theory software capable of performing electronic structure calculations on general organic/biomolecular systems. *QUICK* is capable of performing efficient Hartree-Fock (HF) and density functional theory (DFT) energy and gradient calculations. The standalone version of *QUICK* is available in four different types: serial, MPI parallel, CUDA serial, and CUDA MPI parallel; giving rise to four respective executables: *quick*, *quick.MPI*, *quick.cuda*, and *quick.cuda.MPI*.

QUICK is also available as the QM engine for QM/MM simulations with SANDER. Furthermore, the functionalities of the serial GPU-accelerated and multi-GPU-accelerated versions of *QUICK* can be accessed directly from SANDER executables called *sander.quick.cuda* and *sander.quick.cuda.MPI*; these executables are identical to *sander* and *sander.MPI* in all SANDER functionalities, except they are capable of performing efficient QM/MM calculations with the *QUICK* library through an API. The serial and parallel functionalities of QUICK can be accessed from the *sander* and *sander.MPI* executables. More information about the QM/MM functionalities is provided in section 6.3.

If you use QUICK in your work, please cite the following reference:

- Manathunga, M.; Jin, C.; Cruzeiro, V. W. D.; Smith, J.; Keipert, K.; Pekurovsky, D.; Mu, D.; Miao, Y.; He, X.; Ayers, K; Brothers, E.; Götz, A. W.; Merz, K. M. QUICK-21.03. University of California San Diego, CA and Michigan State University, East Lansing, MI, 2021

If you perform DFT calculations please also cite:

- Manathunga, M.; Miao, Y.; Mu, D.; Götz, A. W.; Merz, K. M. Parallel Implementation of Density Functional Theory Methods in the Quantum Interaction Computational Kernel Program. *J. Chem. Theory Comput.* **16**, 4315–4326 (2020).

If you use the GPU accelerated version of QUICK please also cite:

- Manathunga, M.; Jin, C,; Cruzeiro, V. W. D.; Miao, Y.; Mu, D.; Arumugam, K.; Keipert, K.; Aktulga, H. M.; Merz, K. M., Jr.; Götz, A. W. (2021): Harnessing the Power of Multi-GPU Acceleration into the Quantum Interaction Computational Kernel Program. *ChemRxiv*. Preprint. https://doi.org/10.26434/chemrxiv.13769209.v1

- Mia, Y.; Merz, K. M., Jr. Acceleration of High Angular Momentum Electron Repulsion Integrals and Integral Derivatives on Graphics Processing Units. *J. Chem. Theory Comput.* **11**, 1449-1462 (2015).

More details about *QUICK* can be found in *the QUICK* user manual, at the following link: https://quick-docs.readthedocs.io/en/21.3.0 .

5.1. Features and limitations

The current version of *QUICK*, 21.03, shipped with AmberTools contains the following features and limitations:
Features:

- Hartree-Fock energy calculations
- Density functional theory calculations (LDA, GGA and Hybrid-GGA functionals available)
- Gradient and geometry optimization calculations

- Mulliken charge analysis
- Supports QM/MM calculations with Amber
- MPI parallelization for CPU platforms
- GPU implementation via CUDA for NVIDIA GPUs
- Multi-GPU support via MPI + CUDA, also across multiple compute nodes

Limitations:

- Supports only closed shell systems
- Supports energy/gradient calculations with basis functions up to d
- Supports only Cartesian basis functions (no spherical harmonics)
- Effective core potentials (ECPs) are not supported
- DFT calculations are performed exclusively using SG1 grid system

5.2. Installation

Currently, the installation of *QUICK*, both the standalone executables and the corresponding features for QM/MM simulations, are optional. This means users must choose to compile Amber with *QUICK* support in order to use it. Please note that the two-electron repulsion integral (ERI) code is very complex. As a consequence the compilation of the CUDA code for GPUs can take a long time. It will take several minutes for a single GPU architecture. By default, the Amber build system generates executables that work for all Nvidia GPU architectures that are supported by the available CUDA Toolkit. As a consequence the build can take very long, upwards of half an hour - be patient, the compiler is working hard to generate lightning fast code for HF and DFT calculations!

If compiling Amber with the default installation procedure using CMake (see section **??**), the following steps need to be executed:

1. Add `-DBUILD_QUICK=TRUE` into your `cmake` command at `amber20_src/build/run_cmake`
2. (Re)compile Amber

If installing Amber using the old (legacy) build system (see section **??**), the following must be done instead:

1. Configure Amber with Quick support. Example: `./configure -quick [additional options] gnu`
2. (Re)compile Amber

5.3. Usage

Examples of *QUICK* input files can be found at: *$AMBERHOME/AmberTools/src/quick/test*.

Here is an example of a gradient calculation for a single water molecule at the B3LYP/cc-pVDZ level of theory:

```
B3LYP BASIS=cc-pVDZ CHARGE=0 MULT=1 GRADIENT

O                -0.06756756   -0.31531531    0.00000000
H                 0.89243244   -0.31531531    0.00000000
H                -0.38802215    0.58962052    0.00000000
```

Before running QUICK, users must source *$AMBERHOME/amber.sh* (or *$AMBERHOME/amber.csh*, depending on your enviroment).

Assuming the input file above is called *water.in*, in order to run the calculation with the serial version of *QUICK*:

```
quick water.in
```

Or to run with the CUDA parallel version of *QUICK* with 2 GPUs:

```
mpirun -np 2 quick.cuda.MPI water.in
```

In all cases, an output file called *water.out* will be generated. If you use multiple GPUs, please make sure to use one MPI process per GPU.

6. QM/MM calculations

Sander supports the option of describing part of the system quantum mechanically in an approach known as a hybrid (or coupled potential) QM/MM simulation. There are three basic ways in which QM/MM simulations are enabled in *sander*:

1. Semi-empirical neglect of diatomic overlap (NDDO)-type and density functional tight binding (DFTB) Hamiltonians are supported natively by *sander* via the sqm software library. The basic documentation (e.g. what Hamiltonians are implemented, description of the input parameters) can be found in Chapter 4. In section 6.1below we limit our description to those features that are unique to the QM/MM interface implemented in *sander*.

2. More advanced Hamiltonians based on *ab initio* wave function theory (WFT) and density functional theory (DFT) are supported via an interface to external QM software packages the use of which is described in section 6.2.

3. Seamless *ab initio* QM/MM simulations with Hartree-Fock (HF) and density functional theory (DFT) methods are possible via the GPU enabled QM code QUICK, which is distributed with AmberTools. QUICK can be used either as QM program external to *sander* or via the QUICK library linked to *sander* (recommended). Details about QUICK are in chapter 5 and QM/MM simulations via *sander* and QUICK are described in section 6.3 below.

The built-in semi-empirical QM/MM support was written by Ross Walker and Mike Crowley, [80] based originally on public-domain MOPAC codes of J.J.P. Stewart. The QM/MM generalized Born implementation uses the model described by Pellegrini and Field[114] while regular QM/MM Ewald support is based on the work of Nam *et al.*[115] with QM/MM PME support based on the work of Walker *et al.*[80]. SCC-DFTB support was written by Gustavo Seabra, Ross Walker and Adrian Roitberg,[81] and is based on earlier work of Marcus Elstner.[82, 83] The extension for DFTB3 was written by Andreas Goetz. The interface to external QM packages [116]was originally developed by Andreas Goetz but many others have contributed in the meantime. Vinicius Cruzeiro has coupled Amber to QUICK [117]via the API that was developed by Madu Manathunga.

6.1. Built-in semiempirical NDDO methods and SCC-DFTB

When running a QM/MM simulation in *sander* the system is partitioned into two regions, a QM region consisting of the atoms defined by either the *qmmask* or *iqmatoms* keyword, and a MM region consisting of all the atoms that are not part of the QM region. For a typical protein simulation in explicit solvent the number of MM atoms will be much greater than the number of QM atoms. Either region can contain zero atoms, giving either a pure QM simulation or a standard classical simulation. For periodic simulations, the quantum region must be *compact*, so that the extent (or diameter) of the QM region (in any direction) plus twice the QM/MM cutoff must be less than the box size. Hence, you can define an "active site" to be the QM region, but in most cases could not ask that all cysteine residues (for example) be quantum objects. The restrictions are looser for non-periodic (gas-phase or generalized Born) simulations, but the codes are written and tested for the case of a single, compact quantum region.

The partitioned system is characterized by an effective Hamiltonian which operates on the system's wavefunction Ψ, which is dependent on the position of the MM and QM nuclei, to yield the system energy E_{eff}:

$$H_{eff}\Psi(x_e,x_{QM},x_{MM}) = E_{eff}(x_{QM},x_{MM})\Psi(x_e,x_{QM},x_{MM}) \tag{6.1}$$

The effective Hamiltonian consists of three components - one for the QM region, one for the MM region and a term that describes the interaction of the QM and MM regions, implying that likewise the energy of the system can

be divided into three components. If the total energy of the system is re-written as the expectation value of H_{eff} then the MM term can be removed from the integral since it is independent of the position of the electrons:

$$E_{eff} = \langle \Psi | H_{QM} + H_{QM/MM} | \Psi \rangle + E_{MM} \tag{6.2}$$

In the QM/MM implementation in *sander*, E_{MM} is calculated classically from the MM atom positions using the Amber or CHARMM force field equation and parameters, whereas H_{QM} is evaluated using the chosen QM method.

The interaction term $H_{QM/MM}$ is more complicated. By default, *sander* uses an electrostatic embedding scheme (also referred to as additive scheme) in which the interaction of the MM point charges with the electrons of the QM system as well as the interaction between the MM point charges and the QM nuclei (atomic cores for semi-empirical methods) is explicitly taken into account. In other words, the MM region polarizes the QM electron density. For the case where there are no covalent bonds between the atoms of the QM and MM regions the interaction Hamiltonian is thus the sum of an electrostatic term and a Lennard-Jones (VDW) term and can be written as

$$H_{QM/MM} = \sum_q \sum_m \left[Q_m h_{electron}(x_e, x_{MM}) - Q_m Z_q h_{core}(x_{QM}, x_{MM}) + \left(\frac{A}{r_{qm}^{12}} - \frac{B}{r_{qm}^{6}} \right) \right] \tag{6.3}$$

where the subscripts e, m and q refer to the electrons, the MM nuclei and the QM nuclei respectively. Here Q_m is the charge on MM atom m, Z_q is the core charge (nucleus minus core electrons) on QM atom q, r_{qm} is the distance between atoms q and m, and A and B are Lennard-Jones interaction parameters. For systems that have covalent bonds between the QM and MM regions, the situation is more complicated, as discussed later.

A more approximate form of the interaction term $H_{QM/MM}$ is referred to as mechanical embedding (or subtractive QM/MM scheme). In this case the interactions between the QM and the MM region are obtained within the same classical approximation that is used for the MM region, that is

$$H_{QM/MM} = \sum_q \sum_m \left[\frac{Q_m Q_q}{r_{qm}} + \left(\frac{A}{r_{qm}^{12}} - \frac{B}{r_{qm}^{6}} \right) \right] \tag{6.4}$$

where Q_q is the classical MM point charge assigned to an atom in the QM region. Mechanical embedding is useful to impose steric constraints on the embedded QM system, however, the electron density is not polarized by the MM environment. An additional complication of this approach is that the point charges that are assigned to the atoms in the QM region have to represent the electrostatic potential of the QM region during the whole course of a QM/MM simulation.

If one evaluates the expectation values in Eq. 6.2 over a single determinant built from molecular orbitals

$$\phi_i = \sum_j c_{ij} \chi_j \tag{6.5}$$

where the c_{ij} are molecular orbital coefficients and the χ_j are atomic basis functions, the total energy depends upon the c_{ij} and on the positions x_{MM} and x_{QM} of the atoms. The energy is obtained by setting $\partial E_{eff}/\partial c_{ij}$ to zero which leads to a self-consistent (SCF) procedure to determine the c_{ij}, (with a modified Fock matrix that contains the electric field arising from the MM charges in the case of electrostatic embedding). Once the energy is known, the forces on the atoms can be obtained by taking the derivative of the energy expression with respect to the positions of the QM and MM atoms.

The main subtlety that arises in the case of electrostatic embedding is that, for a periodic system, there are formally an infinite number of QM/MM interactions; even for a non-periodic system, the (finite) number of such interactions may be prohibitively large. These problems are addressed in a manner analogous to that used for pure MM systems: a PME approach is used for periodic systems, and a (large) cutoff may be invoked for non-periodic systems. Some details are discussed below.

6.1.1. The QM/MM interface and link atoms

The sections above dealt with situations where there are no covalent bonds between the QM and MM regions. In many protein simulations, however, it is necessary to have the QM/MM boundary cut covalent bonds, and a number of additional approximations have to be made. There are a variety of approaches to this problem, including hybrid orbitals, capping potentials, and explicit link atoms. The last option is the method available in *sander*.

There are a number of ways to implement a link atom approach that deal with the way the link atom is positioned, the way the forces on the link atom are propagated, and the way non-bonding interactions around the link atom are treated. Each time an energy or gradient calculation is to be done, the link atom coordinates are re-generated from the current coordinates of the QM and MM atoms making up the QM-MM covalent pair. The link atom is placed along the bond vector joining the QM and MM atom, at a distance d_{L-QM} from the QM atom. By default d_{L-QM} is set to the equilibrium distance of a methyl C-H atom pair (1.09 Å) but this can be set in the input file. The default link atom type is hydrogen, but this can also be specified as an input.

Since the link atom position is a function of the coordinates of the "real" atoms, it does not introduce any new degrees of freedom into the system. The chain rule is used to re-write forces on the link atom itself in terms of forces on the two real atoms that define its position. This is analogous to the way in which "extra points" or "lone-pairs" are handled in MM force fields.

The remaining details of how the QM-MM boundary is treated are as follows: for the interactions surrounding the link atom, the MM bond term between the QM and MM atoms is calculated classically using the classical force field parameters, as are any angle or dihedral terms that include at least one MM atom. The Lennard-Jones interactions between QM-MM atom pairs are calculated in the same way as described in the section above with exclusion of 1-2 and 1-3 interactions and scaling of 1-4 interactions. What remains is to specify the electrostatic interactions between QM and MM atoms around the region of the link atom.

A number of different schemes have been proposed for handling link-atom electrostatics. Many of these have been tested or calibrated on (small) gas-phase systems, but such testing can neglect some considerations that are very important for more extended, condensed-phase simulations. In choosing our scheme, we wanted to ensure that the total charge of the system is rigorously conserved (at the correct value) during an MD simulation. Further, we strove to have the Mulliken charge on the link atom (and the polarity of its bond to the nearest QM atom) adopt reasonable values and to exhibit only small fluctuations during MD simulations. Link atoms interact with the MM field in exactly the same way as regular QM atoms. That is they interact with the electrostatic field due to all the MM atoms that are within the cutoff, with the exception of the MM link pair atoms (MM atoms that are bound directly to QM atoms). VDW interactions are not calculated for link atoms. These are calculated between all real QM atoms and all MM atoms, including the MM link pair atoms. For Generalized Born simulations the effective Born radii for the link atoms are calculated using the intrinsic radii for the MM link pair atoms that they are replacing.

In the case of electrostatic embedding the atoms that make up the QM region (including the MM link pair atom) have their charges from the prmtop file essentially replaced with Mulliken charges. Hence it is important to consider the issue of charge conservation. The QM region (including the link atoms) by definition must have an integer charge. This is defined by the &qmmm namelist variable *qmcharge*. If the MM atoms (including the MM link pair atoms) that make up the QM region have prmtop charges that sum to the value of *qmcharge* then there is no problem. If not, there are two options for dealing with this charge, defined by the namelist variable *adjust_q*. A value of 1 will distribute the difference in charge equally between the nearest *nlink* MM atoms to the MM link pair atoms. A value of 2 will distribute this charge equally over all of the MM atoms in the simulation (excluding MM link pair atoms).

6.1.2. A reformulated QM/MM interface for PM3

In the current version of Amber, a reformulated QM-MM core-charge potential (denoted as PM3/MM*) has been implemented. This reformulated potential scales the interaction between a QM core and a MM charge for the purpose of better description of the geometry and energy at the QM-MM interface:[118]

$$E_{QM/MM}^{core} = Z_a q_m \left(s_a s_a, s_m s_m\right) \left[1 + \frac{|q_m|}{q_m} \cdot \left(-e^{-f_1^a \cdot R_{am}} + e^{-f_2^a \cdot R_{am}}\right)\right] \tag{6.6}$$

where Z_a is the effective core charge of QM atom a, q_m is the partial charge on MM atom m, s_a is an s orbital on the QM atom, s_m is a notional s orbital on the MM atom, R_{am} is the QM-MM interatomic distance, and f_1^a and f_2^a are exponential scale factors which depend on the QM atom only. Optimal values for f_1^a and f_2^a were determined based on the PM3 Hamiltonian, and are available for H, C, N and O atoms (so the QM region is limited to these four atoms; but the MM region is not restricted). Application of this reformulated potential shows improved prediction of geometry and interaction energy at the QM-MM interface for hydrogen bonded small molecule complexes typical of biomolecular interactions, without significantly impacting the modeling of other interaction types, such as dispersion dominant complexes.[118] In a QM/MM calculation, giving *qmmm_int=3* along with *qm_theory=PM3* will invoke this potential.

Based on PM3/MM*, further developments to the semi-empirical QM/MM coupling method have been introduced – PM3/MMX2 (*qmmm_int*=4 and *qm_theory*=PM3) – which shares the same QM core-MM charge equation with the PM3/MM* model. In addition, a QM parameter, ρ_{mm}, is introduced to each type of QM atoms in order to "fine-tune" the QM electron-MM charge interaction (Eq. 6.7). Although ρ_{mm} is a parameter for QM atom, the subscript *mm* emphasizes that it is a MM-related property (eqn 3.xx). Parameters are currently available for H, C, N, O and S QM atoms (manuscript in preparation).

$$E_{QM/MM}^{electron} = -q_m(\mu_a v_a, s_m s_m) = \sum_{\ell_a}\sum_{\ell_m}\left[M_{l_a k}^a M_{l_m k}^m\right] \tag{6.7}$$

where

$$\left[M_{l_a k}^a M_{l_m k}^m\right] = \frac{e^2}{2^{l_a+l_m}} \sum_{i=1}^{2^{l_a}} \sum_{j=1}^{l} \left[r_{ij}^2 + \left(\rho_{l_a+}^a \rho_{mm}^a\right)^2\right]^{-1/2} \tag{6.8}$$

6.1.3. Generalized Born implicit solvent

The implementation of Generalized Born (GB) for QM/MM calculations is based on the method described by Pellegrini and Field.[114] Here, the total energy is taken to be E_{eff} from Eq. 6.2 plus E_{gb} from Eq. 2.2. In E_{gb}, charges on the QM atoms are taken to be the Mulliken charges determined from the quantum calculation; hence these charges depend upon the molecular orbital coefficients c_{ij} as well as the positions of the atoms.

As with conventional QM/MM simulations, one then solves for the c_{ij} by setting $\partial E_{eff}/\partial c_{ij} = 0$. This leads to a set of SCF equations with a Fock matrix modified not only by the presence of MM atoms (as in "ordinary" QM/MM simulations), but also modified by the presence of the GB polarization terms. Once self-consistency is achieved, the resulting Mulliken charges can be used in the ordinary way to compute the GB contribution to the total energy and forces on the atoms.

6.1.4. Ewald and PME

The support for long range electrostatics in QM/MM calculations using electrostatic embedding is based on a modification of the Nam, Gao and York Ewald method for QM/MM calculations.[115] This approach works in a similar fashion to GB in that Mulliken charges are used to represent long range interactions. Within the cutoff, interactions between QM and MM atoms are calculated using a full multipole treatment. Outside of the cutoff the interaction is based on pairwise point charge interactions. For semiempirical NDDO-type methods this leads to a slight discontinuity at the QM/MM cutoff boundary and thus a small energy drift during QM/MM MD simulations in the NVE ensemble. This energy drift can be avoided by using a switching function at the cutoff (see below).

The implementation in Ref [115] uses an Ewald sum for both QM/QM and QM/MM electrostatic interactions. This can be expensive for large MM regions, and thus *sander* uses a modification of this method by Walker and Crowley[80] that uses a PME model (rather than an Ewald sum) for QM/MM interactions. This is controlled by the *qm_pme* variable discussed below.

When running QM/MM Ewald or PME simulations in *sander*, if QM multipoles are involved in QM-MM interactions (NDDO methods), a discontinuity in the QM-MM electrostatic potential occurs at the cutoff distance due to the sudden change in the potential function (the difference between Eqs. 6.9 and 6.10) , thus resulting in energy conservation problems in the simulation.

$$E_{QM/MM}^{r<cutoff} = -q_m(\mu_a v_a, s_m s_m) + Z_a q_m(s_a s_a, s_m s_m)(1+scale) \tag{6.9}$$

$$E_{QM/MM}^{r>cutoff} = \frac{q_m(Z_a - \sum c_{\mu\mu})}{r} \tag{6.10}$$

This problem can be avoided by applying a switching function to smoothly connect the two different potentials. The QM/MM electrostatic potential using a switching function can thus be written as:

$$E_{QM/MM} = E_{QM/MM}^{r<cutoff} s(r) + E_{QM/MM}^{r>cutoff}(1-s(r))$$

The switching function can be turned on or off via the *&qmmm* namelist variable *qmmm_switch,* for details see section 6.1.6 below.

6.1.5. Hints for running successful QM/MM calculations

Required Parameters and Prmtop Creation

QM/MM calculations without link atoms require mass, charges, van der Waals and GB radii in the *prmtop* file. All bonds, angles, and dihedrals parameters involving QM atoms are neglected. In the case of electrostatic embedding the charges are also neglected. (Note that when SHAKE is applied to the QM reg, the bonds are constrained to the ideal MM values, even when these are part of a QM region; hence, for this case, it is important to have correct bond parameters in the QM region.) The simplest general prescription for setting things up is to use *antechamber* and *LEaP* to create a reference force field, since "placeholders" are required in the *prmtop* file even for things that will be neglected. This also allows you to run comparison simulations between pure MM and QM/MM simulations, which can be helpful if problems are encountered in the QM/MM calculations.

The use of *antechamber* to construct a pure MM reference system is even more useful when there are link atoms, since here MM parameters for bonds, angles and dihedrals that cross the QM/MM boundary are also needed.

Choosing the QM region

There are no good universal rules here. Generally, one might want to have as large a QM region as possible, but having more than 80-100 atoms in the QM region will lead to simulations that are very expensive. One should also remember that for many features of conformational analysis, a good MM force field may be better than a semiempirical or DFTB quantum description. In choosing the QM/MM boundary, it is better to cut non-polar bonds (such as C-C single bonds) than to cut unsaturated or polar bonds. Link atoms are not placed between bonds to hydrogen. Thus cutting across a C-H bond will NOT give you a link atom across that bond. (This is not currently tested for in the code and so it is up to the user to avoid such a situation.) Furthermore, link atoms are restricted to one per MM link pair atom. This is tested for during the detection of link atoms and an error is generated if this requirement is violated. This would seem to be a sensible policy otherwise you could have two link atoms too close together. See the comments in *qm_link_atoms.f* for a more in-depth discussion of this limitation.

Choice of electrostatic cutoff

The implementation of the non-bonded cut off in QM/MM simulations is slightly different than in regular MM simulations. The cut off between MM-MM atoms is still handled in a pairwise fashion. However, for QM atoms any MM atom that is within *qmcut* of ANY QM atom is included in the interaction list for all QM atoms. This means that the value of *qmcut* essentially specifies a shell around the QM region rather than a spherical shell around each individual QM atom. Ideally the cut off should be large enough that the energy as a function of the cutoff has converged. For non-periodic, generalized Born simulations, a cutoff of 15 to 20 Å seems sufficient in some tests. (Remember that long-range electrostatic interactions are reduced by a factor of 80 from their gas-phase counterparts, and by more if a nonzero salt concentration is used.) For periodic simulations, the cutoff only serves to divide the interactions between "direct" and "reciprocal" parts; as with pure MM calculations, a cutoff of 8 or 9 Å is sufficient here.

Parallel simulations

The built-in QM/MM implementation currently supports execution in parallel via the message passing interface (MPI), however, the implementation is not fully parallel. At present all parts of the QM simulation are parallel except the density matrix build and the matrix diagonalisation. For small QM systems these two operations do not take a large percentage of time and so acceptable scaling can be seen to around 8 CPU cores (depending on type of CPU and/or interconnect speed between compute nodes). For large QM systems the matrix diagonalisation time will dominate and so the scaling will not be as good. In this case it may be beneficial to choose a LAPACK diagonalization routine in combination with a threaded library such as the Intel Math Kernel Library (MKL). For details on how to choose the diagonalization routine see Section 4.3. The number of threads to be used for the diagonalization is set via an environment variable of the operating system (typically OMP_NUM_THREADS).

6.1.6. General QM/MM &qmmm Namelist Variables

An example input file for running a simple QM/MM MD simulation is shown here:

```
&cntrl
 imin=0, nstlim=10000,          ! Perform MD for 10,000 steps
 dt=0.002,                      ! 2 fs time step
 ntt=1, tempi=0.1, temp0=300.0, ! Berendsen temperature control
 ntb=1,                         ! Constant volume periodic boundaries
 ntf=2, ntc=2,                  ! Shake hydrogen atoms
 cut=8.0,                       ! 8 angstrom classical non-bond cut off
 ifqnt=1                        ! Switch on QM/MM coupled potential
/
&qmmm
 qmmask=':753',    ! Residue 753 should be treated using QM
 qmcharge=-2,      ! Charge on QM region is -2
 qm_theory='PM3', ! Use the PM3 semi-empirical Hamiltonian
 qmcut=8.0         ! Use 8 angstrom cut off for QM region
/
```

The *&qmmm* namelist contains variables that allow you to control the options used for a QM/MM simulation. This namelist must be present when running QM/MM simulations and at the very least must contain either the *iqmatoms* or *qmmask* variable which define the region to be treated quantum mechanically. If *ifqnt* is set to zero then the contents of this namelist are ignored.

For the QM region definition specify one of either *iqmatoms* or *qmmask*. Link atoms will be added automatically along bonds (as defined in the prmtop file) that cross the QM/MM boundary.

`iqmatoms` comma-separated integer list containing the atom numbers (from the prmtop file) of the atoms to be treated quantum mechanically.

`qmmask` Mask specifying the quantum atoms. E.g. :1-2, = residues 1 and 2. See mask documentation for more info.

`qmcut` Specifies the size of the electrostatic cutoff in Angstroms for QM/MM electrostatic interactions. By default this is the same as the value of cut chosen for the classical region, and the default generally does not need to be changed. Any classical atom that is within qmcut of *any* QM atom is included in the pair list. For PME calculations, this parameter just affects the division of forces between direct and reciprocal space. *Note:* this option only effects the electrostatic interactions between the QM and MM regions. Within the QM region all QM atoms see all other QM atoms regardless of their separation. QM-MM van der Waals interactions are handled classically, using the cutoff value specified by *cut*.

`qm_ewald` This option specifies how long range electrostatics for the QM region should be treated.

= 0 Use a real-space cutoff for QM-QM and QM-MM long range interactions. In this situation QM atoms do not see their images and QM-MM interactions are truncated at the cutoff. This is the default for non-periodic simulations.

= 1 (default) Use PME or an Ewald sum to calculate long range QM-QM and QM-MM electrostatic interactions. This is the default when running QM/MM with periodic boundaries and PME.

= 2 This option is similar to option 1 but instead of varying the charges on the QM images as the central QM region changes the QM image charges are fixed at the Mulliken charges obtained from the previous MD step. This approach offers a speed improvement over *qm_ewald=1*, since the SCF typically converges in fewer steps, with only a minor loss of accuracy in the long range electrostatics. This option has not been extensively tested, although it becomes increasingly accurate as the box size gets larger.

`kmaxqx,y,z` Specifies the maximum number of kspace vectors to use in the x, y and z dimensions respectively when doing an Ewald sum for QM-MM and QM-QM interactions. Higher values give greater accuracy in the long range electrostatics but at the expense of calculation speed. The default value of 8 should be optimal for most systems.

`ksqmaxq` Specifies the maximum number of K squared values for the spherical cut off in reciprocal space when doing a QM-MM Ewald sum. The default value of 100 should be optimal for most systems.

`qm_pme` Specifies whether a PME approach or regular Ewald approach should be used for calculating the long range QM-QM and QM-MM electrostatic interactions.

= 0 Use a regular Ewald approach for calculating QM-MM and QM-QM long range electrostatics. Note this option is often much slower than a pme approach and typically requires very large amounts of memory. It is recommended only for testing purposes.

= 1 (default) Use a QM compatible PME approach to calculate the long range QM-MM electrostatic energies and forces and the long range QM-QM forces. The long range QM-QM energies are calculated using a regular Ewald approach.

`qmmm_switch` Specifies whether a switching function shall be used at the cutoff for long range electrostatics (applies only to NDDO methods). The lower and higher boundaries of the switching function are user definable, see *r_switch_lo* and *r_switch_hi*.

= 0 (default). Do not use a switching function. This leads to slight discontinuities in the potential at the cut off and thus an energy drift in NVE simulations.

= 1 Use a switching function. See also variables *r_switch_hi* and *r_switch_lo*.

`r_switch_hi` Specifies the upper boundary of the switching function in Å (see *qmmm_switch*). Defaults to *qmcut*.

`r_switch_lo` Specifies lower boundary of the switching function in Å (see *qmmm_switch*). Defaults to *r_switch_hi* - 2.

`qmgb` Specifies how the QM region should be treated with generalized Born.

= 2 (default) As described above, the electrostatic and "polarization" fields from the MM charges and the exterior dielectric (respectively) are included in the Fock matrix for the QM Hamiltonian.

= 3 This is intended as a debugging option and should only be used for single point calculations. With this option the GB energy is calculated using the Mulliken charges as with option 2 above but the fock matrix is NOT modified by the GB field. This allows one to calculate what the GB energy would be for a given structure using the gas phase quantum charges. When combined with a simulation using *qmgb=2*, this allows the strain energy from solvation to be calculated.

`qm_theory` Level of theory to use for the QM region of the simulation. (Hamiltonian). Default is to use the semi-empirical hamiltonian PM3. See the *Section 4.3* for details.

`qmmm_int` Controls the way in which QM/MM interactions are handled in the direct space QMMM sum. This controls only the electrostatic interactions. VDW interactions are always calculated classically using the standard 6-12 potential. Note: with the exception of qmmm_int=0 DFTB calculations (qm_theory=DFTB) always use a simple mulliken charge - resp charge interaction and the value of qmmm_int has no influence.

- **= 0** This turns off all electrostatic interaction between QM and MM atoms in the direct space sum. Note QM-MM VDW interactions will still be calculated classically.
- **= 1** (default) QM-MM interactions in direct space are calculated in the same way for all of the various semi-empirical hamiltonians. The interaction is calculated in an analogous way to the the core-core interaction between QM atoms. The MM resp charges are included in the one electron hamiltonian so that QMcore-MMResp and QMelectron-MMResp interactions are calculated.
- **= 2** This is the same as for 1 above except that when AM1, PM3 or Hamiltonians derived from these are in use the extra Gaussian terms that are introduced in these methods to improve the core-core repulsion term in QM-QM interactions are also included for the QM-MM interactions. This is the equivalent to the QM-MM interaction method used in CHARMM and DYNAMO. It tends to slightly reduce the repulsion between QM and MM atoms at small distances. For distances above approximately 3.5 angstroms it makes almost no difference.
- **= 3** Using this along with *qm_theory=PM3* invokes a reformulated QM core-MM charge potential at the QM-MM interface (Eq. 6.6). Current parametrization limits the QM region to H, C, N and O atoms only; MM region is not restricted.[118]
- **= 4** Currently not in use.
- **= 5** Mechanical embedding: The electrostatic interaction between QM and MM atoms is treated on the same level as within the MM region using the classical force field point charges also for the QM atoms. The electronic Hamiltonian does not contain the field generated by the MM region point charges and thus the electron density is not polarized by the MM environment. Does not work with GB. Not extensively tested in presence of link atoms.

`qmshake` Controls whether SHAKE is applied to QM atoms. Using SHAKE on the QM region will allow you to use larger time steps such as 2 fs with NTC=2. If, however, you expect bonds involving hydrogen to be broken during a simulation you should not SHAKE for the QM region. WARNING: the SHAKE routine uses the equilibrium bond lengths as specified in the prmtop file to reset the atom positions. Thus while bond force constants and equilibrium distances are not used in the energy calculation for QM atoms the equilibrium bond length is still required if QM SHAKE is on.

- **= 0** Do not shake QM H atoms.
- **= 1** Shake QM H atoms if SHAKE is turned on (NTC>1) (default).

`printdipole` Controls whether the dipole moment shall be printed every *ntpr* steps.

- **= 0** Do not print the dipole moment (default).
- **= 1** Print the dipole moment of the QM region.
- **= 2** Print the total dipole moment of the QM and MM region.

`writepdb`

- **= 0** Do not write a PDB file of the selected QM region. (default).
- **= 1** Write a PDB file of the QM region. This option is designed to act as an aid to the user to allow easy checking of what atoms were included in the QM region. When this option is set a crude PDB file of the atoms in the QM region will be written on the very first step to the file *qmmm_region.pdb*.

`vsolv` Controls whether solvent molecules shall be included into the QM region (requires settings in the *&vsolv* namelist; see also section **??** on adaptive solvent QM/MM simulations, in particular the namelist information in section **??**).

= 0 Do not include solvent molecules into the QM region (default).

= 1 Include solvent molecules via simple solvent switching (requires *&vsolv* namelist).

= 2 Adaptive solvent QM/MM with fixed number of solvent molecules in A and T regions (requires *&vsolv* and *&adqmmm* namelists).

= 3 Adaptive solvent QM/MM with fixed size of A and T regions (requires *&vsolv* and *&adqmmm* namelists).

In addition to the above parameters, the following variables may be set, as described in Section 4.3:

qm_theory, dftb_disper, dftb_3rd_order , dftb_chg , dftb_telec , dftb_maxiter , qmcharge, spin, qmqmdx, verbosity, tight_p_conv, scfconv, pseudo_diag, pseudo_diag_criteria, diag_routine, printcharges, qxd, parameter_file, peptide_corr, and itrmax.

6.1.7. Link Atom Specific QM/MM &qmmm Namelist Variables

The following options go in the *&qmmm* namelist and control the link atom behaviour.

`lnk_dis` Distance in Å from the QM atom to its link atom. Currently all link atoms must be placed at the same distance. A negative value of lnk_dis specifies that the link atom should be placed directly on top of the MM link pair atom. In this case the distance of the link atom from the QM region changes as a function of time and the actual value of lnk_dis is ignored. Additionally this means that not all link atoms will be placed at the same distance. Negative values of lnk_dis will work with regular link atoms, such as hydrogen, but are really intended for use with pseudo atom / capping approaches. Default = 1.09Å.

`lnk_method` This defines how classical valence terms that cross the QM/MM boundary are dealt with.

=1 (Default) in this case any bond, angle or dihedral that involves at least one MM atom, including the MM link pair atom is included. This means the following (where QM = QM atom, MM = MM atom, MML = MM link pair atom.):

Bonds = MM-MM, MM-MML, MML-QM

Angles = MM-MM-MM, MM-MM-MML, MM-MML-QM, MML-QM-QM

Dihedrals = MM-MM-MM-MM, MM-MM-MM-MML, MM-MM-MM-MML-QM, MM-MML-QM-QM, MML-QM-QM-QM

=2 Only include valence terms that include a full MM atom, that is, count the MM link pair atom as effectively being a QM atom. This option is designed to be used in conjunction with a pseudo atom / capping type approach where the link atom is parameterized specifically to behave like a uni-valent version of the MM atom it replaces. This option gives the following interactions:

Bonds = MM-MM, MM-MML

Angles = MM-MM-MM, MM-MM-MML, MM-MML-QM

Dihedrals = MM-MM-MM-MM, MM-MM-MM-MML, MM-MM-MML-QM, MM-MML-QM-QM

`lnk_atomic_no` The atomic number of the link atoms. This selects what element the link atoms are to be. Default = 1 (Hydrogen). Note this must be an integer and an atomic number supported by the chosen QM theory.

`adjust_q` This controls how charge is conserved during a QMMM calculation involving link atoms. When the QM region is defined the QM atoms and any MM atoms involved in link bonds have their RESP charges zeroed. If the sum of these RESP charges does not exactly match the value of *qmcharge* then the total charge of the system will not be correct.

= 0 No adjustment of the charge is done.

= 1 The charge correction is applied to the nearest *nlink* MM atoms to MM atoms that form link pairs. Typically this will be any MM atom that is bonded to a MM link pair atom (a MM atom that is part of a QM-MM bond). This results in the total charge of QM+QMlink+MM equaling the original total system charge from the prmtop file. Requires natom-nquant-nlink >= nlink and nlink>0.

= 2 (default) - This option is similar to option 1 but instead the correction is divided among all MM atoms (except for those adjacent to link atoms). As with option 1 this ensures that the total charge of the QM/MM system is the same as that in the *prmtop* file. Requires natom-nquant-nlink >= nlink.

6.1.8. Charge-dependent exchange-dispersion corrections of vdW interactions

The *sqm* program provides a new charge-dependent energy model consisting of van der Waals (vdW) and polarization interactions between the quantum mechanical (QM) and molecular mechanical (MM) regions in a combined QM/MM calculation. vdW interactions are commonly treated using empirical Lennard-Jones (L-J) potentials, whose parameters are often chosen based on the QM atom type (e.g., based on hybridization or specific covalent bonding environment). This strategy for determination of QM/MM nonbonding interactions becomes tedious to parametrize and lacks robust transferability. Problems occur in the study of chemical reactions where the "atom type" is a complex function of the reaction coordinate. This is particularly problematic for reactions, where atoms or localized functional groups undergo changes in charge state and hybridization.

In *sqm*, this charge-dependent energy model was implemented based on a scaled overlap model for repulsive exchange and attractive dispersion interactions that is a function of atomic charge. The model is chemically significant since it properly correlates atomic size, softness, polarizability, and dispersion terms with minimal one-body parameters that are functions of the atomic charge[84].

This “Charge-dependent exchange-dispersion corrections of vdW interactions” can be invoked by the “qxd=.true.” switch in the *&qmmm* namelist. Note that this model currently does not have any effect on pure quantum calculations through *sqm*, the qxd correction is only added to QM/MM interactions in *sander*. The default values of qxd parameters are set to reproduce the regular L-J interactions of typical atom types (HC for H, C* for C, N for N, OW for O, and parameters for F and Cl are optimized[84]) when the charge dependence parameters are zero. There are eight qxd parameters (symbols used in the reference[84] are indicated in the parentheses): qxd_s (s), qxd_z0 ($\zeta(0)$), qxd_zq (ζ_q), qxd_d0 (α_1), qxd_dq ($3 \times B$), qxd_q0 (α_2), qxd_qq ($3 \times B$), and qxd_neff ($N_{eff}(0)$). All parameters can be modified through external user-defined parameter files (see the usage of ’parameter_file’ in Section 4.3).

6.2. Interface for *ab initio* and DFT methods

In addition to the built-in semi-empirical methods *sander* also supports QM/MM simulations with *ab initio* wave function theory (WFT) and density functional theory (DFT) potentials via an interface to external QM software packages[116]. The implementation makes use of the existing QM/MM infrastructure that has been developed earlier for the semi-empirical methods. Thus, much of AMBER’s previous QM/MM functionality such as the user-friendly link atom approach are available and the implementation remains simple and transparent to use without any significant additional steps in the simulation setup as compared to semi-empirical QM/MM simulations. At present the interface supports several well-known and widely used QM software packages. Mechanical embedding is available for

- ADF (Amsterdam Density Functional) [119, 120]

- GAMESS-US [121, 122]
- NWChem [123]

Mechanical and electrostatic embedding is available for

- Gaussian [124]
- Orca [125]
- Q-Chem[126][126]
- TeraChem [127]
- QUICK [109, 112, 113]
- MRCC [128, 129]
- Fireball [130]

While ADF, Gaussian, Q-Chem and TeraChem are commercial programs, GAMESS-US, NWChem, Orca, QUICK, MRCC and Fireball are available at no cost for academic research. QUICK is available as standalone QM package but also distributed with AmberTools (see chapter 5). The QUICK library can be linked against *sander* and we recommend using this API based version over the file based interface (FBI), for details see section 6.3. Fireball, which implements a density functional theory-based tight binding approach, requires compilation of *sander* with special flags, see the section on Fireball below for details. The interface has been written in a modular fashion and is easily extensible to support other QM software packages. It is our intention to keep adding support for other software packages. If you are interested in interfacing a specific program, please do not hesitate to contact us.

The interface was developed by Andreas Goetz (SDSC, UCSD) with help of Matthew Clark (SDSC) and support by Ross Walker (SDSC, UCSD). Thanks are due to Christine Isborn and Todd Martinez (Stanford University) for modifications to the TeraChem code to support this interface, to Mark Williamson (University of Cambridge) for an initial version of the module that supports NWChem, Bence Hégely for contributing code that supports MRCC, and Jesús Mendieta and José Ortega Mateo for contributing code that supports Fireball. If you make use of this interface, please cite the following work:

- A. W. Götz, M. A. Clark, R. C. Walker, *An extensible interface for QM/MM molecular dynamics simulations with AMBER*, J. Comput. Chem. **35**, 95-108 (2014), DOI: 10.1002/jcc.23444

If you are using the interface with the TeraChem code, please cite in addition the following work:

- C. M. Isborn, A. W. Götz, M. A. Clark, R. C. Walker, T. J. Martínez, *Electronic Absorption Spectra from MM and ab initio QM/MM Molecular Dynamics: Environmental Effects on the Absorption Spectrum of Photoactive Yellow Protein*, J. Chem. Theory Comput. **8**, 5092-5106 (2012), DOI: 10.1021/ct3006826

If you are using the interface with the MRCC code, please cite in addition the following work:

- B. Hégely, F. Bogár, G. G. Ferenczy, M. Kállay, *A QM/MM program for calculations with frozen localized orbitals based on the Huzinaga equation*, Theoret. Chem. Acc. **134**, 132 (2015), DOI: 10.1007/978-3-662-49825-5_16

If you are using the interface with the Fireball code, please cite in addition the following work:

- J. I. Mendieta-Moreno, R. C. Walker, J. P. Lewis, P. Gómez-Puertas, J. Mendieta, J. Ortega, *FIREBALL/AMBER: An efficient local-orbital DFT QM/MM method for biomolecular systems*, J. Chem. Theory Comput. 10, 2185-2193 (2014), DOI: 10.1021/ct500033w

Access to QM methods not available within Amber is also possible via the Amber interface to the PUPIL simulation framework. For details, see refs. 131, 132. In what follows we will describe the new interface that is native to *sander*.

6.2.1. Theory

As described in section 6.1, the Hamiltonian of a system that is partitioned into a QM region that is treated with WFT and a classical region that is treated with MM consists of three components and the energy associated with this Hamiltonian is obtained as the corresponding expectation value

$$E = \langle \Psi | \mathscr{H}_{QM} + \mathscr{H}_{QM/MM} | \Psi \rangle + E_{MM}. \tag{6.11}$$

A QM/MM calculation therefore requires not only to choose the WFT used in the QM region and the MM model used for the MM region, but in addition also the form of the QM/MM Hamiltonian which describes the interaction between the quantum and the classical region. The most simple approach is to neglect any electronic coupling between the QM and the MM system and include only the classical non-bonded van der Waals (vdW) and electrostatic interactions between the QM and the MM atoms. This is useful to impose steric constraints on the embedded QM system and commonly referred to as mechanical embedding. In most cases, however, it is better to allow for an explicit polarization of the QM system due to the presence of the point charges on the MM atoms. This is referred to as electronic embedding and the resulting interaction energy becomes

$$\begin{aligned} E_{QM/MM}^{electronic} = & \sum_{A \in MM} \int \rho(\mathbf{r}) \frac{Q_A}{|\mathbf{r} - \mathbf{R}_A|} \, d\mathbf{r} + \sum_{A \in QM, B \in MM} \frac{Z_A Q_B}{R_{AB}} \\ & + \sum_{A \in QM, B \in MM} \varepsilon_{AB} \left[\left(\frac{\sigma_{AB}}{R_{AB}} \right)^{12} - \left(\frac{\sigma_{AB}}{R_{AB}} \right)^{6} \right]. \end{aligned} \tag{6.12}$$

This QM/MM energy expression also holds for DFT and the terms represent, in order, the electrostatic interaction between the QM electron density and the MM point charges, the electrostatic interation between the QM point charge nuclei and the MM point charges, and the van der Waals repulsion between the QM and MM atoms.

The forces acting on an atom A in a QM/MM calculation are given in terms of derivatives of the total energy expression (6.11) with respect to the Cartesian coordinates of the atom,

$$\mathbf{F}_A = -\nabla_A E_{QM} - \nabla_A E_{QM/MM} - \nabla_A E_{MM}, \tag{6.13}$$

where $\nabla_A = \partial / \partial \mathbf{R}_A = (\partial / \partial R_A^x, \partial / \partial R_A^y, \partial / \partial R_A^z)$. If a QM and an MM program are coupled for QM/MM calculations, the QM program will calculate the QM forces $-\nabla_A E_{QM}$ acting on QM atoms and the MM program the MM forces $-\nabla_A E_{MM}$ acting on the MM atoms. All that remains, is to calculate the forces acting on QM and MM atoms due to the QM/MM interaction energy, $-\nabla_A E_{QM/MM}$. For mechanical embedding this will be entirely handled by the MM program. For electronic embedding the forces are given as

$$\begin{aligned} \nabla_A E_{QM/MM}^{electronic} &= Z_A \sum_{B \in MM} \frac{Q_B (\mathbf{R}_A - \mathbf{R}_B)}{R_{AB}^3} + \sum_{B \in MM} \int \frac{\partial \rho(\mathbf{r})}{\partial \mathbf{R}_A} \frac{Q_B}{|\mathbf{r} - \mathbf{R}_B|} \, d\mathbf{r} + \sum_{B \in MM} \nabla_A V_{AB}^{LJ} \\ &= -Z_A \mathbf{E}_{MM}(\mathbf{R}_A) - \int \rho(\mathbf{r}) \mathbf{E}_{MM}(\mathbf{r}) \, d\mathbf{r} + \sum_{B \in MM} \nabla_A V_{AB}^{LJ} \end{aligned} \tag{6.14}$$

for the derivatives with respect to the positions of the QM atoms A where $\mathbf{E}_{MM}$ is the electric field generated by the MM point charges and and V_{AB}^{LJ} is the Lennard-Jones potential from (6.12) and

$$\begin{aligned} \nabla_B E_{QM/MM}^{electronic} &= Q_B \sum_{A \in QM} \frac{Z_A (\mathbf{R}_B - \mathbf{R}_A)}{R_{AB}^3} + \int \rho(\mathbf{r}) \frac{Q_B (\mathbf{R}_B - \mathbf{r})}{|\mathbf{r} - \mathbf{R}_B|^3} \, d\mathbf{r} + \nabla_B E_{QM/MM}^{mechanic} \\ &= -Q_B \mathbf{E}_{QM}(\mathbf{R}_B) + \sum_{A \in QM} \nabla_B V_{AB}^{LJ} \end{aligned} \tag{6.15}$$

for the derivatives with respect to the positions of the MM atoms B where $\mathbf{E}_{QM}$ is the electric field due to the QM charge distribution. The contributions to the gradient due to the point charge interactions and due to the interaction

between the MM point charges and the QM electrons is evaluated by the QM program. Some QM programs do not calculate the forces acting on the MM atoms (point charges) due to the presence of the QM system but in general are able to return the electric field $\mathbf{E}_{QM}$ at arbitrary points in space which is then used to obtain these forces. The van der Waals repulsion (Lennard-Jones interaction) between QM and MM atoms is treated by AMBER in the same way as for semiempirical NDDO-type and DFTB methods.

6.2.2. General Remarks

When using the AMBER interface to external QM software packages for performing WFT or DFT based QM or QM/MM MD simulations, it is absolutely critical to be aware of the capabilities and limitations of the QM method to be employed. In particular, QM based MD can be more tricky than MM based MD in the sense that it is more likely that the QM program can fail for example due to SCF convergence problems. This can be the case if the geometry of the QM region is far from its ground state equilibrium, for example because a simulation is started from a bad geometry or performed at high temperature.

We have gone to large efforts and analyzed a large set of test simulations to provide the best default parameters for the supported QM programs such that forces are computed with sufficient accuracy to guarantee energy coservation for constant energy MD simulations. Of particular importance are SCF convergence and associated integral neglect thresholds and the size of the grid used for the numerical quadrature of the exchange-correlation (XC) potential and energy for DFT calculations. However, other than providing appropriate input parameters, AMBER does not have any control over the external program and it is at the user's discretion to employ sensible input parameters for the QM program and to prepare the system such that the simulations are started at a reasonable starting structure.

In any case we highly recommend to write restart files frequently so that a simulation can be restarted without loss of much computational time in the case that a simulation should crash. The interface also stores the last in- and output files of the external QM program during each MD step. Should there be any problems with the QM program, it is therefore possible to analyze the reasons and take appropriate countermeasures.

The interface requires data to be exchanged between *sander* and the QM program. The default operation of the interface is based on file exchange and system calls and, during each step of a geometry optimization or an MD simulation, writes an input file for the external program, starts a single point gradient calculation with the external program, and reads the energy and forces from the external program's output file (binary ADF checkpoint or formatted GAMESS, Gaussian, ORCA, Q-Chem, MRCC or TeraChem output files). Data communication via MPI is also implemented and currently supported by TeraChem. An exception is Fireball, which is interfaced as a linked library against *sander* (see below).

6.2.3. Limitations

In principle, all types of simulations that are possible with sander are supported. There are, however, some restrictions for simulations that require *sander* to run in parallel, in particular path integral molecular dynamics (PIMD) and replica exchange molecular dynamics (REMD), see the discussion of Parallelization below. The interface to external QM programs also lacks some features regarding solvent models in comparison to the semiempirical MNDO and DFTB QM/MM implementation that is available in AMBER, the most critical ones are listed here.

Generalized Born Generalized Born (GB) implicit solvent models are not supported if external QM programs are used for the QM region.

Particle Mesh Ewald (PME) and Periodic Boundary Conditions The PME approach for treating long-range electrostatic QM/MM and QM/QM interactions in periodic systems is currently not supported. It is possible to use periodic boundary conditions but a cutoff is used for the point charges to be included in the QM Hamiltonian (determined by *&qmmm* namelist variable *qmcut*) thus truncating the long-range QM/MM electrostatic interactions in (6.12). This leads to discontinuities in the potential energy surface and poor energy conservation for MD runs in the NVE ensemble. The user may consider running non-periodic simulations with a cutoff that is larger than the system size thus effectively including all interactions.

6.2.4. Performance Considerations

The computational cost of DFT is comparable to Hartree–Fock (HF) theory which is the simplest WFT method that serves as zeroth order approximation for more elaborate correlated WFT methods such as Møller–Plesset perturbation theory, configuration interaction theory and coupled cluster theory. The calculations can be accelerated by using density fitting approaches, sometimes called resolution-of-identity (RI) approximation, which in the case of DFT with exchange-correlation (XC) functionals that do not require admixture of exact HF-exchange, leads to speedups of roughly one order of magnitude without compromising the accuracy of the results. Nevertheless, the computational cost of DFT is in general two to three orders of magnitude higher than that of semiempirical QM models. We recommend to carefully test the performance of the QM program to choose an optimal number of processor counts for parallelized QM calculations. Typical simulation performance for typical QM system sizes of tens of atoms will be on the order of a few picoseconds per day, depending on the underlying QM model chosen.

6.2.5. Parallelization

The MPI parallel executable *sander.MPI* can be used to run QM/MM MD simulations with external QM software in which the MM portion of the calculation is parallelized. However, the computational cost of the MM part is usually small compared to the cost of the QM part. In order to execute the QM part of the calculation in parallel, the external QM program has to be instructed to do so, as described in the sections below.

In the case of PIMD or REMD simulations that require a separate energy and force evaluation for each group at each time step, the parallelized executable *sander.MPI* has to be used. Multiple processes can be launched per group to parallelize the MM calculations. Care has to be taken to choose the right number of parallel threads in the external QM program. For example, on a machine with 32 cores, a simulation with 16 beads or replicas can run the external QM program with 2 threads in parallel to make maximum use of the available processing cores. If the available processors are spread over multiple nodes, special care has to be taken to ensure that the different instances of the external QM program are launched on the correct nodes.

It is possible to execute *sander.MPI* in parallel via MPI while also running MPI or OpenMP parallel versions of the external QM program. Depending on the MPI implementation, this can, however, fail. In our experience, MPICH and MVAPICH work well while OpenMPI does not work.

6.2.6. Usage

All that is required to use the interface is a working installation of AMBER and one or more of the supported QM programs. In order to use the external program from within *sander*, the *&cntrl* namelist variable *ifqnt = 1* must be set to enable QM calculations and the *&qmmm* namelist variable *qm_theory = 'EXTERN'* must be set to enable the external interface. The *&qmmm* namelist variable *qmmask* or *iqmatoms* is used for selecting the QM region just as for QM/MM calculations with the semiempirical NDDO-type and DFTB approaches that are natively available in AMBER. Charge and spin multiplicity for the QM region need to be defined via the variables *qmcharge* and *spin*, respectively, in the *&qmmm* namelist. For a QM MD simulation, the sander input file therefore needs to contain

```
! example input for QM simulation with external QM program
&cntrl
  ...
  ifqnt = 1,              ! switch on QM/MM
/
&qmmm
  qmmask = '@*',          ! select QM atoms (here: make all QM)
  qmcharge = 0,           ! charge on QM region (default = 0)
  spin = 1,               ! spin multiplicity of QM region (default = 1)
  qm_theory = 'EXTERN',   ! use external QM program
/
```

For QM/MM simulations with electronic embedding (this is the default) we recommend to include all MM point charges as external electric field in the QM Hamiltonian to avoid problems with energy conservation. For non-periodic simulations this can be achieved by setting the *&qmmm* namelist variable *qmcut* to a value larger than the system size.

In addition either the *&adf*, *&gms*, *&nw*, *&gau*, *&orc*, *&qc, &mrcc* or *&tc* namelist must be present to use either ADF, GAMESS, NWChem, Gaussian, ORCA, Q-Chem, MRCC or TeraChem, respectively, and to assign parameters for the external QM program. Please refer to the ADF, GAMESS, NWChem, Gaussian, ORCA, Q-Chem, MRCC or TeraChem user manual for details on settings for the *ab initio* or DFT calculations. A list of namelist variables and their default setting is given below. The defaults have been chosen such that energy conserving MD simulations in the NVE ensemble are possible. NWChem has not been extensively tested.

Properties that are calculated along the trajectory are printed to property files with names `adf_job.ext`, `gms_job.ext`, `gau_job.ext`, `orc_job.ext`, `qc_job.ext` and `tc_job.ext`, where `ext` is either `dip` for dipole moment (x, y, z component and absolute value) or `chg` for atomic charges, where supported. These property files are only written if requested and will be deleted at the beginning of a run, so back them up in case a trajectory needs to be restarted.

All calculations with a spin multiplicity larger than one will automatically be performed in the framework of an unrestricted formalism (as opposend to restricted open shell), that is with unrestricted HF (UHF), unrestricted DFT (UDFT) and MP2 with a UHF reference wave function (UMP2).

In addition to controlling the external programs via the *sander* input file, you may supply a template input file for the external program in order to provide input that is not supported via the program specific namelists. To enable this option, you must set `use_template = 1` in the program specific namelist. The format, name, and input requirements for the template file vary with the external program as detailed in the corresponding program's documentation below. If you are using your own template, please make sure that the parameters of the QM method (like SCF convergence threshold and XC quadrature grid size) yield sufficiently accurate forces. Please note that program settings supplied via the program specific namelist are ignored if a template input file is used.

6.2.6.1. AMBER/ADF

To use ADF with the external interface, ADF must be properly installed on the working machine. In particular, the executable `adf` must be in the search path. By default the Becke integration grid with quality "good" and the ZLM fit method with quality "good" is employed. If you prefer to use the old pair fit method (or are using an older ADF version that does not support the ZLM fit), we recommend to use "ZORA/QZ4P" basis set for the density fit for sufficiently accurate forces.

Limitations At present only mechanical embedding is supported.

***&adf* Namelist variables**

`basis`	Basis set type to be used in the DFT calculation. Valid standard basis set types are: SZ, DZ, DZP, TZP, TZ2P, TZ2P+ and ZORA/QZ4P. (Default: basis = 'DZP')
`core`	Type of frozen core to use. Allowed values are: None, Small, Medium, Large. (Default: core = 'None')
`zlmfit`	Quality of density fit with the ZLM fit method. (Default: zlmfit = 'good')
`fit_type`	Fit basis set type to be used for density fitting with the old pair fit method. Valid values are identical to the available basis sets (SZ, DZ etc) in which case the fit basis corresponding to the AO Basis will be used. By default the ZLM fit method will be used (Default: fit_type = '')
`xc`	Exchange-correlation functional to be used. Popular choices are 'LDA VWN', 'GGA BLYP', 'GGA PBE', 'HYBRID B3LYP' and 'HYBRID PBE0'. Consult the ADF manual for all available options. (Default: xc = 'GGA BLYP')
`scf_iter`	Maximum number of SCF cycles allowed. (Default: scf_iter = 50)

`scf_conv` Threshold upon which to stop the SCF procedure. The tested error is the commutator of the Fock matrix and the density matrix. Convergence is considered to be achieved if the maximum element of the commutator (which is zero for an optimized wave function) is smaller than scf_conv. (Default: scf_conv = 1.0d-06)

`beckegrid` Quality of Becke integration grid. Allowed values are: Normal, Good, VeryGood. (Default: core = 'Good')

`integration` Numerical integration accuracy for integration with olde teVelde-Baerends integration grid (Voronoi cells). By default the Becke grid will be used. The old integration grid can be used by specifying a number larger than 0, we recommend at least 5.0. (Default: integration = -1.0)

`num_threads` Number of threads (and thus CPU cores) for ADF to use. Note that this is not required if you are running in a queuing system as ADF will automatically use the full number of reserved cores. (Default: num_threads = 0 [this causes ADF to use all available cores on a machine])

`use_dftb` Specifies whether DFTB shall be used with ADF's DFTB program dftb. If use_dftb = 1 then DFTB will be used and only variables charge and scf_conv will be considered. (Default: use_dftb = 0 [do not use DFTB, regular DFT calculation]) - works only with older DFTB versions (prior to 2011).

`exactdensity` The exact (as opposed to fitted) electron density is used for the evaluation of the exchange-correlation potential if exactdensity = 1. (Default: exactdensity = 0)

`use_template` Determine whether or not to use a user-provided template file for running external programs. (Default: use_template = 0)

`ntpr` Controls frequency of printing for dipole moment to file `adf_job.dip` (Defaults to &cntrl namelist variable ntpr)

`dipole` Toggles writing of dipole moment to file `adf_job.dip` (Default: dipole = 0)

Example An input file for QM or (mechanical embedding) QM/MM MD with ADF using the PBE functional and the TZP basis set therefore would have to contain

```
&adf
  xc = 'GGA PBE',
  basis = 'TZP',
/
```

This would execute a simulation in which the Beckgrid with quality quality good and the ZLM fit with quality good are used (see default values above).

Template input file The template file for ADF should be named `adf_job.tpl` and must contain the following keywords:

```
BASIS ... END
SAVE TAPE21
```

You should not include the following (block) keywords in the template file as these are taken care of by *sander*:

```
UNITS
FRAGMENTS ... END
RESTART
GRADIENT
ATOMS ... END
```

6.2.6.2. AMBER/GAMESS-US

To use GAMESS with the external interface, GAMESS must be compiled on the target system. Make note of the version number you specify during the GAMESS compilation process (default is 00 which makes the GAMESS execution script `rungms` look for the executable `gamess.00.x`). If you use a different version number you must specify it with the *gms_version* namelist variable. `$GMS_PATH` should be set to the path where the script `rungms` is located (for example `/opt/gamess/`). We assume that the `rungms` script copies the output `.dat` files to the directory from which GAMESS is invoked. If this is not the case, please modify the script `rungms` accordingly.

Limitations Only mechanical embedding is supported with GAMESS. The available QM models are limited to HF, DFT and MP2 since only for these analytical gradients are available in GAMESS.

***&gms* Namelist variables**

`basis` Basis set type to be used in the calculation. Presently supported are the Pople type basis sets STO-3G, 6-31G, 6-31G*, 6-31G**, 6-31+G*, 6-31++G*, 6-311G, 6-311G* and 6-311G**. Also supported are the Karlsruhe valence triple zeta basis sets KTZV, KTZVP and KTZVPP (with none, one and two polarization functions, respectively) and the Dunning-type correlation consistent basis sets CCn (n = D, T, Q, 5, 6; officially called cc-pVnZ) and ACCn (as CCn but augmented with a set of diffuse function, officially called aug-cc-pVnZ). (Default: basis = "6-31G*")

`method` QM method to be used. At present, we support 'HF' for Hartree–Fock, 'MP2' for second order Møller-Plesset perturbation theory and any of the supported DFT functionals. Popular choices for for DFT functionals include BP86, BLYP, PBE, B3LYP or PBE0. (Default: method = "BP86")

`nrad` Number of radial points in the Euler-MacLaurin quadrature of the XC potential and energy density. (Default: nrad = 96)

`nleb` Number of angular points in the Lebedev grids for the numerical quadrature of the XC potential and energy density. (Default: nleb = 590 [The GAMESS default of 302 is not accurate enough to conserve energy])

`scf_conv` SCF convergence threshold. Convergence is reached when the absolute density change between two consecutive SCF cycles is less than scf_conv}. (Default: scf_conv = 1.0D-06)

`maxit` Maximum number of SCF iterations. (Default: maxit = 50)

`gms_version` This is the version number specified when building GAMESS. (Default: gms_version = 00)

`num_threads` Number of threads (and thus CPU cores) for GAMESS to use. Note that GAMESS may require a special setup in the `rungms` script to be able to run using multiple threads. Unless num_threads is explicitly specified, GAMESS will only use one thread (run on one core). (Default: num_threads = 1)

`mwords` The maximum replicated memory which your job can use, on every node. This is given in units of 1,000,000 words (as opposed to 1024*1024 words), where a word is defined as 64 bits. You may need to increase this value if GAMESS crashes due to not having enough memory allocated. (Default: mwords = 50)

`use_template` Determine whether or not to use a user-provided template file for running external programs. (Default: use_template = 0)

`ntpr` Controls frequency of printing for dipole moment and atomic charges to files `gms_prop.ext` (Defaults to &cntrl namelist variable ntpr)

`chelpg` CHELPG charges are calculated if chelpg = 1. These charges are written to the file `gms_prop.chg` (Default: chelpg = 0)

`dipole` Toggles writing of dipole moment to file `gms_prop.dip` (Default: dipole = 0)

Example An input file for QM or (mechanical embedding) QM/MM MD with GAMESS using the PBE functional and the 6-31G** basis set that should run GAMESS on 16 CPU cores therefore would have to contain

```
&gms
  method = 'DFT',
  dfttyp = 'PBE',
  basis  = '6-31G**',
  num_threads = 16,
/
```

Template input file The template file for GAMESS should be named `gms_job.tpl` and the `$CONTRL` card must contain the following keywords:

```
RUNTYP=GRADIENT
UNIT=ANGS
COORD=UNIQUE
```

You should not include the `$DATA` card in the template file as it is taken care of by *sander*.

6.2.6.3. AMBER/Gaussian

To use Gaussian with the interface, Gaussian 16, Gaussian 09, or Gaussian 03 must be properly installed on the system and a g16, g09, or g03 executable must be in the path.

Limitations A cutoff is applied to QM/MM interactions in QM/MM simulations using electrostatic embedding with and without PBCs. This leads to discontinuities in the potential energy surface and poor energy conservation. In the case of QM/MM simulations without PBCs, this cutoff (*qmcut* variable in the *&qmmm* namelist) can be set to a number that is larger than the simulated system, thus effectively not applying a cutoff. This is recommended.

&gau Namelist variables

`basis` Basis set type to be used in the calculation. Any basis set that is natively supported by Gaussian can be used. Examples are the single zeta, split valence or triple zeta Pople type basis sets STO-3G, 3-21G, 6-31G and 6-311G. The split-valence or triple zeta basis sets can be augmented with diffuse functions on heavy atoms or additionally hydrogen by adding one or two plus signs, respectively, as in 6-31++G. Polarization functions on heavy atoms or additionally hydrogens are used by adding one or two stars, respectively, as in 6-31G**. (Default: basis = "6-31G*")

`method` Method to be used in the calculation. Can either be one of the WFT models for which Gaussian supports gradients, for example RHF or MP2, or some supported DFT functional. Popular choices are BLYP, PBE and B3LYP. (Default: method = "BLYP")

`scf_conv` Threshold upon which to stop the SCF procedure. The tested error is the commutator of the Fock matrix and the density matrix. Convergence is considered to be achieved if the maximum element of the commutator (which is zero for an optimized wave function) is smaller than scf_conv}. Set in the form of 10^{-N}. (Default: scf_conv = 8)

`num_threads` Number of threads (and thus CPU cores) for Gaussian to use. Unless num_threads is explicitly specified, Gaussian will only use one thread (run on one core). (Default: num_threads = 1)

`executable` Optional name of the Gaussian executable. (Default: If a string for this namelist variable is not specified then g16, g09, and g03 are tried in that order producing a fatal error if none are found. Note that if a string is specified then it is a fatal error if that executable is not found.)

`use_template` Determine whether or not to use a user-provided template file for running external programs. (Default: use_template = 0)

`ntpr` Controls frequency of printing for dipole moment to file `gau_job.dip` (Defaults to &cntrl namelist variable ntpr)

`dipole` Toggles writing of dipole moment to file `gau_job.dip` (Default: dipole = 0)

`mem` String that specifies how much memory Gaussian should be allowed to use. (Default: '256MB')

Example An input file for QM or QM/MM MD with Gaussian using the BP86 functional and the 6-31G** basis set and running in parallel on 8 threads (using 1 GB of memory) therefore would have to contain

```
&gau
  method = 'BP86',
  basis  = '6-31G**',
  num_threads = 8,
  mem='1GB',
/
```

Template input file The template file for Gaussian should be named `gau_job.tpl` and should only contain the route section of a Gaussian input file. The route section defines the method to be used and SCF convergence criteria. Charge and spin multiplicity are specified via the *&qmmm* namelist. For example for a B3LYP calculation with 6-31G* basis set, the route section would be:

```
#P B3LYP/6-31G* SCF=(Conver=8)
```

Do not include any information about coordinates or point charge treatment since this will all be handled by *sander*. Also, do not include any *Link 0 Commands* (line starting with %) since these are handled by *sander*. If you want to run Gaussian in parallel, specify the number of processors via the *num_threads* variable in the *&gau* namelist.

6.2.6.4. AMBER/Orca

To use Orca with the interface, Orca must be properly installed on the system, the Orca executables need to reside in a directory that is in the search path. For convenience of use, namelist parameters in general correspond to Orca keywords, see the Orca manual for details.

Limitations A cutoff is applied to QM/MM interactions in QM/MM simulations with and without PBCs. This leads to discontinuities in the potential energy surface and poor energy conservation. In the case of QM/MM simulations without PBCs, this cutoff (*qmcut* variable in the *qmmm* namelist) can be set to a number that is larger than the simulated system, thus effectively not applying a cutoff. This is recommended.

Also note that ORCA only supports OpenMPI for parallel calculations.

&orc Namelist variables

`basis` Basis set type to be used in the calculation. Possible choices include svp, 6-31g, etc. See Orca manual for a complete list. (Default: basis = "SV(P)")

`cbasis` Auxiliary basis set for correlation fitting. See Orca manual for a complete list. (Default: basis = "NONE")

`jbasis` Auxiliary basis set for Coulomb fitting. See Orca manual for a complete list. (Default: basis = "NONE")

`method` Method to be used in the calculation. Popular choices include hf, pm3, blyp, and mp2. (Default: method = "blyp")

`convkey` General SCF convergence setting for simplified Orca input. Can take values 'TIGHTSCF', 'VERYTIGHTSCF', etc. (Default: convkey='VERYTIGHTSCF')

`scfconv` SCF convergence threshold for the energy. (Default: scfconv = -1, that is, not in use since we use the general convergence settings keyword *convkey*. Otherwise this would lead to SCF energy convergence of 10^{-N}au, if set to N.)

`grid` Grid type used during the SCF for the XC quadrature in DFT. (Default: grid = 4, this corresponds to Intacc = 4.34 for the radial grid and an angular Lebedev grid with 302 points. Conservatively chosen together with finalgrid to conserve energy.)

`finalgrid` Grid type used for the energy and gradient calculation after the SCF for the XC quadrature in DFT. (Default: finalgrid = 6, this corresponds to Intacc = 5.34 for the radial grid and an angular Lebedev grid with 590 points. Conservatively chosen together with *grid* to conserve energy.)

`maxiter` Maximum number of SCF iteractions. (Default maxiter = 100)

`maxcore` Global scratch memory (in MB) used by Orca. You may need to increase this when running larger jobs. See Orca manual for more information. (Default maxcore = 1024)

`num_threads` Number of threads (and thus CPU cores) for Orca to use. Note that Orca only supports OpenMPI. (Default: num_threads = 1)

`use_template` Determine whether or not to use a user-provided template file for running external programs. (Default: use_template = 0)

`ntpr` Controls frequency of printing for the dipole moment to file `orc_job.dip` (Defaults to &cntrl namelist variable ntpr)

`dipole` Toggles writing of the dipole moment to file `orc_job.dip` (Default: dipole = 0)

Example An input file for QM or QM/MM MD with Orca using the BLYP functional, the SVP basis set therefore would have to contain

```
&orc
  method = 'blyp',
  basis  = 'svp',
/
```

Template input file The template file for Orca should be named `orc_job.tpl` and must at least contain keywords specifying the method and basis set to be used in the calculation, for example:

```
# ORCA input file for BLYP/SVP simulation
! BLYP SVP
```

You should not include the following keywords in the template file as these are taken care of by sander (like setting the runtype and adding coordinates):

```
# NOT to be included in ORCA input file
!engrad
!energy  # (or any run type)
%pointcharges
*xyzfile # (or any coordinates)
```

6.2.6.5. AMBER/Q-Chem

To use Q-Chem with the interface, Q-Chem must be properly installed on the system. The q-chem executable needs to reside in a directory that is in the search path. For convenience of use, namelist parameters in general correspond to Q-chem keywords, see the Q-Chem manual for details. The interface has been tested with Q-Chem versions 4.0.0.1 and 4.1.1 for HF, DFT and MP2. Other methods have not been tested and could cause problems - please be careful and verify that forces/energies used by sander are correct in this case.

Limitations A cutoff is applied to QM/MM interactions in QM/MM simulations with and without PBCs. This leads to discontinuities in the potential energy surface and poor energy conservation. In the case of QM/MM simulations without PBCs, this cutoff (*qmcut* variable in the *qmmm* namelist) can be set to a number that is larger than the simulated system, thus effectively not applying a cutoff. This is recommended.

&qc Namelist variables

`basis` Basis set type to be used in the calculation. Possible choices include '6-31g**', 'cc-pVDZ' etc. See the Q-chem manual for a complete list. (Default: basis = '6-31G*' for DFT calculations and basis = 'cc-pVDZ' for MP2)

`auxbasis` Auxiliary basis set for density fitting / RI methods. See Q-Chem manual for a complete list. (Default: basis = 'rimp2-cc-pVDZ' for RI-MP2 calculations, otherwise none)

`method` Method to be used in the calculation. Popular choices include 'BLYP' or other density functionals, 'MP2' and 'RIMP2'. Alternatively, the keywords exchange and correlation can be employed. (Default: method = 'BLYP')

`exchange` Exchange method. Can be specified together with the correlation keyword in place of the combined method keyword. (Default: exchange = '')

`correlation` Correlation method. Can be specified together with the exchange keyword in place of the combined method keyword. (Default: correlation = '')

`scf_conv` SCF convergence threshold. (Default: scfconv = 6)

`num_mpi_prcs` Number of MPI processes for Q-Chem to use. The total number of CPUs to be used is num_mpi_prcs times num_threads. (Default: num_mpi_prcs = 1)

`num_threads` Number of threads for Q-Chem to use for each MPI process. Really this is number of threads. The total number of CPUs to be used is num_mpi_prcs times num_threads. (Default: num_threads = 1)

`use_template` Determine whether or not to use a user-provided template file for running external programs. (Default: use_template = 0)

ntpr Controls frequency of printing for the dipole moment to file `qc_job.dip` (Defaults to &cntrl namelist variable ntpr)

`dipole` Toggles writing of the dipole moment to file `qc_job.dip`. This is currently not supported. (Default: dipole = 0)

`guess` Toggles use of MOs from previous step as initial guess to accelerate SCF convergence. Any string different from 'read' will disable this. (Default: guess = 'read')

Example An input file for QM or QM/MM MD with Q-Chem using MP2 with the cc-pVTZ basis set therefore would have to contain

```
&qc
  method = 'mp2',
  basis  = 'cc-pVTZ',
/
```

Template input file The template file for Q-chem must be named `qc_job.tpl` and must only contain keywords in the Q-Chem $rem input section that specify the QM method and basis set to be used in the calculation, for example:

```
EXCHANGE becke
CORRELATION lyp
BASIS 6-311G**
SCF_CONVERGENCE 7
```

The interface will take care of adding other keywords to the $rem section such as JOBTYPE and writing the $molecule input file sections.

6.2.6.6. AMBER/MRCC

To use MRCC with the interface, the MRCC program suite must be properly installed on the system. The MRCC driver program dmrcc needs to reside in a directory that is in the search path. For convenience of use, namelist parameters in general correspond to MRCC keywords, see the MRCC manual for details. The interface has been tested with the MRCC release from July 15, 2016 for HF and DFT. Other methods have not been tested but should also work - please be careful and verify that forces/energies used by sander are correct in this case.

Limitations A cutoff is applied to QM/MM interactions in QM/MM simulations with and without PBCs. This leads to discontinuities in the potential energy surface and poor energy conservation. In the case of QM/MM simulations without PBCs, this cutoff (*qmcut* variable in the *qmmm* namelist) can be set to a number that is larger than the simulated system, thus effectively not applying a cutoff. This is recommended.

&mrcc Namelist variables

`basis`	Basis set type to be used in the calculation. Possible choices include '6-31g**', 'cc-pVDZ' etc. See the MRCC manual for a complete list. (Default: basis = '6-31G*')
`calc`	Type of calculation, e.g. 'SCF', 'B3LYP', 'MP2', 'CCSD(T)', etc. (Default: calc = 'SCF')
`dft`	Can be specified to request a DFT calculation and specify the DFT method. (Default: dft = 'off')
`mem`	Memory that will be allocated for the calculation. (Default: mem = '256MB')
`verbosity`	Controls the verbosity of the MRCC output file. (Default: verbosity = 2)
`ntpr`	Controls frequency of printing for the dipole moment to file `mrcc_job.dip` (Defaults to &cntrl namelist variable ntpr)
`do_dipole`	Toggles writing of the dipole moment to file `mrcc_job.dip`. (Default: dipole = 0)
`nprintlog`	Frequency of storing MRCC output files during a minimization of molecular dynamics run. (Default: keep only last output file, nprintlog = 0)
`debug`	Toggles debug mode, which prints subroutine calls and additional information about the AMBER/MRCC interface. (Default: no debugging, debug = 0)

`use_template` Requests use of a template file to generate MRCC input files to utilize all the capabilities of that are not available through &mrcc namelist keywords. The template file is basically a truncated MINP file (the default input file for MRCC) which only includes the MRCC keywords. (Default: do not use a template input file, use_template = 0)

The following &mrcc namelist variables control multilayer calculations (i.e. QM/QM/MM or QM/QM/QM/MM embedding[133]; the region highlighted in bold is controlled by the keyword). Only single point calculations are currently possible with such mulitlayer calculations.

`embed` Specifies the method of the embedding QM region (2. layer) in a QM/**QM**/MM (3 layer) calculation or specifies the method of the 3. layer in a QM/QM/**QM**/MM (4 layer) calculation. Please read the MRCC manual for available options. (Default: embed = 'off')

`embedatoms` Specifies the active atoms of the embedded QM region (1. layer) in a **QM**/QM/MM (3 layer) calculation or specifies the active atoms of the 1. and 2. layer in a **QM/QM**/QM/MM (4 layer) calculation. Comma separated list of integers (Default: embedatoms = 0)

`nmo_embed` Specifies the number of active MOs of the embedded QM region (1. layer) in a **QM**/QM/MM (3 layer) calculation or specifies the number of active MOs of the 1. and 2. layer in a **QM/QM**/QM/MM (4 layer) calculation.

= 0 The program automatically determines the MOs of the active region with the Boughton-Pulay (BP) algorithm. (default)

> 0 Number of MOs that will be selected based on the Mulliken charges of the active atoms.

`corembed` Specifies the low-level correlation method of the embedding QM region (2. layer) in a QM/**QM**/MM (3 layer) calculation or specifies the low-level correlation method of the 2. layer in a QM/**QM**/QM/MM (4 layer) calculation. Please read the MRCC manual for available options. (Default: corembed = 'off')

`corembedatoms` Specifies the active atoms of the embedded QM region (1. layer) in a **QM**/QM/MM (3 layer) calculation or specifies the active atoms of the 1. layer in a **QM**/QM/QM/MM (4 layer) calculation. Please note that the corembedatoms have to be a subset of the embedatoms if a 4 layer calculation is requested. Comma separated list of integers (Default: corembedatoms = 0)

`nmo_corembed` Specifies the number of active MOs of the embedded QM region (1. layer) in a **QM**/QM/MM (3 layer) calculation or specifies the number of active MOs of the 1. layer in a **QM**/QM/QM/MM (4 layer) calculation.

= 0 The program automatically determines the MOs of the active region with the Boughton-Pulay (BP) algorithm. (default)

> 0 Number of MOs that will be selected based on the Mulliken charges of the active atoms.

Examples An input file for QM or QM/MM MD with MRCC using DFT with the BLYP functional and the cc-pVTZ basis set therefore would have to contain

```
&mrcc
  calc = 'blyp',
  basis  = 'cc-pVTZ',
/
```

An example input for a multilayer QM/QM/MM calculation with LCCSD(T) for a subset of QM atoms 7 to 12 embedded into the remainder of the QM region described by PBE (i.e. LCCSD(T)/PBE/MM) would be

```
&mrcc
  calc = 'LCCSD(T)',
  basis  = 'cc-pVTZ',
  embed = 'PBE',
  embedatoms = 7,8,9,10,11,12
/
```

This assumes that atoms 7-12 are part of the QM region. A 4-layer QM/QM/QM/MM calculation with LCCSD(T) for atoms 7 to 12 embedded into LMP2 for atoms 13 to 16 and the remainder described by PBE (i.e. LCCSD(T)/LMP2/DFT/MM) would be requested with

```
&mrcc
  calc = 'LCCSD(T)',
  basis  = 'cc-pVTZ',
  embed = 'PBE',
  embedatoms = 7,8,9,10,11,12,13,14,15,16,
  corembed = 'LMP2',
  corembedatoms = 7,8,9,10,11,12,
/
```

Template input file The template file for MRCC must be named `mrcc_job.tpl` and must only contain keywords that specify the QM method and basis set to be used in the calculation. Not to be included are following keywords: qmmm, qmreg, dens, pointcharges, geom, embed, corembed, scfiguess. The interface will take care of adding other keywords and writing the coordinate input file section.

6.2.6.7. AMBER/TeraChem

To use TeraChem with the interface, TeraChem must be properly installed on the system. In particular, the `terachem` executable needs to be in the search path. Namelist parameters correspond to TeraChem keywords, see the TeraChem manual for details.

Limitations A cutoff is applied to QM/MM interactions in QM/MM simulations with and without PBCs. This leads to discontinuities in the potential energy surface and poor energy conservation. In the case of QM/MM simulations without PBCs, this cutoff (*qmcut* variable in the *&qmmm* namelist) can be set to a number that is larger than the simulated system, thus effectively not applying a cutoff. This is recommended.

&tc Namelist variables

`basis` Basis set type to be used in the calculation. Possible choices presently (TeraChem version 1.4) are 'STO-3G', '3-21G', '6-31G' and '6-311G', '3-21++G' and '6-31++G' (Default: basis = '6-31G')

`method` Method to be used in the calculation, can be either 'RHF' or some supported DFT functional. Popular choices are 'BLYP', 'PBE' and 'B3LYP'. (Default: method = 'BLYP')

`dftd` Determines whether dispersion corrections are applied in the case of DFT calculations. (Default: dftd = 'no')

`precision` Precision model setting (single vs double precision). (Default: precision = 'mixed')

`dynamicgrid` Use coarse grid during early SCF iterations. (Default: dynamicgrid = 'yes')

`threall` Determines a variety of thresholds. (Default: threall = 1.0E-11)

`convthre` SCF convergence threshold for the wavefunction. (Default: convthre = 3.0E-05, which leads to SCF energy convergence of approximately 10^{-7} au or 10^{-4} kcal/mol)

`maxit` Maximum number of SCF iterations. (Default: maxit = 100)

`dftgrid` DFT grid to be employed for the numerical XC quadrature in DFT calculations. (Default: dftgrid = 1)

`ngpus` Determines how many GPUs are to be used. (Default: ngpus = 0, which uses all available GPUs)

`gpuids` If *ngpus* has a value other than zero, this determines the IDs of the GPUs to be used for the calculation. (Default: gpuids = 0, 1, 2, etc.)

`executable` Name of the TeraChem executable. (Default: executable = terachem)

`use_template` Determine whether or not to use a user-provided template file for running external programs. (Default: use_template = 0)

`ntpr` Controls frequency of printing for dipole moment and atomic charges to files `tc_job.ext`. (Defaults to &cntrl namelist variable ntpr)

`charge_analysis` Toggles writing of atomic charges to file `tc_job.chg` (Options: 'none' or 'Mulliken'. Default: dipole = 'none')

`dipole` Toggles writing of dipole moment to file `tc_job.dip` (Default: dipole = 0)

Example An input file for QM or QM/MM MD with TeraChem using the PBE functional and the 6-31G* basis set therefore would have to contain

```
&tc
  method = 'PBE',
  basis  = '6-31G*',
/
```

Template input file The template file for Terachem should be named `tc_job.tpl` and must at least contain the following keywords:

```
basis
method
```

Any content of the template file after a line containing the `end` keyword will be ignored.

You should not include the following keywords in the template file as these are taken care of by *sander*. Instead, specify these via the *&qmmm* or *&tc* namelist:

```
run
charge
spinmult
coordinates
pointcharges
amber
gpus
```

6.2.6.8. AMBER/Fireball

To use Fireball with the QM/MM interface, a special version of *sander* must be compiled and linked against the Fireball library (libfireball.a). The Fireball library can be obtained from the fireball-qmd web site at https://fireball-qmd.github. Compilation requires the Intel compilers and Intel MKL library. You can compile a version of *sander* that supports Fireball as follows (bash assumed):

```
export FIREBALLHOME=/path/to/fireball.a
export MKL_HOME=/path/to/Intel/MKL/library
cd $AMBERHOME
./configure -fireball intel
make install
```

It is possible to compile the MPI parallel version of *sander* in the same fashion. However, only the MM part of the calculation will execute in parallel.

Limitations A cutoff is applied to QM/MM interactions in QM/MM simulations with and without PBCs. This leads to discontinuities in the potential energy surface and poor energy conservation. In the case of QM/MM simulations without PBCs, this cutoff (*qmcut* variable in the *&qmmm* namelist) can be set to a number that is larger than the simulated system, thus effectively not applying a cutoff.

Basis set Fireball requires a basis set, commonly provided in an "Fdata" directory. This directory contains all the interactions (different contributions to the electronic Hamiltonian matrix elements) for the different types of atoms (C, H, O, N, etc.) appearing in the QM region. In principle, the Fdata directory should be placed in the working directory. Alternatively, the path where the Fdata directory is located can be defined using the variable *basis* in the *&fb* namelist variables (see below).

This Fdata directory can be downloaded from the fireball-qmd web (https://fireball-qmd.github). Advanced users can also calculate their own Fdata using the *create* set of programs that can be found in the fireball-qmd github repository.

&fb Namelist variables

`basis` Path to the Fdata directory. (Default: basis = './Fdata')

`max_scf_iterations` Maximum number of iterations in the loop for the calculation of the self-consistent charges. (Default: max_scf_iterations = 70)

`sigmatol` Threshold for selc-consistency in the electronic structure calculations. (Default: sigmatol = 1.0E-08)

`idftd3` DFTD3 dispersion correction. (No correction: idftd3 = 0; Dispersion correction for BLYP: idftd3 = 1; Default: idftd3 = 0)

`iwrtcharges` Writes atomic charges in fireball output. (Default: iwrtcharges = 0)

`iwrteigen` Writes energy levels in fireball output. (Default: iwrteigen = 0)

Fora complete list of all *&fb* Namelist variables, please visit http://nanosurf.fzu.cz/wiki/doku.php?id=fireball

Example An input file for QM or QM/MM MD using AMBER/FIREBALL with all the default values would just have to contain an empty *&fb* naemlist

```
&fb
/
```

As another example, a simulation using DFTD3 dispersion corrections for BLYP that also writes out the atomic charges with Fdata in a central location of the user's home directory would need the following input:

```
&fb
  basis = '/home/fireball/Fdata',
  idftd3 = 1,
  iwrtcharges = 1
/
```

To launch the simulation, simply run *sander* as follows:

```
sander -O -i mdin -o mdout -p prmtop -c inpcrd -x mdcrd -r rstrt > amberfireball.out
```

6.3. QM/MM simulations with QUICK

The sander program has the capability to run QM/MM simulations with the quantum chemical code *QUICK* (QUantum Interaction Computational Kernel),[109–113] shipped along with AmberTools. If you use QM/MM simulations with *QUICK* in your work, please cite the following references:

- Manathunga, M.; Jin, C.; Cruzeiro, V. W. D.; Smith, J.; Keipert, K.; Pekurovsky, D.; Mu, D.; Miao, Y.; He, X.; Ayers, K; Brothers, E.; Götz, A. W.; Merz, K. M. QUICK-21.03. University of California San Diego, CA and Michigan State University, East Lansing, MI, 2021

- Cruzeiro, V. W. D.; Manathunga, M.; Merz, K. M.; Götz, A. W.; Open-Source Multi-GPU-Accelerated QM/MM Simulations with AMBER and QUICK. ChemRxiv; 2021. DOI: 10.26434/chemrxiv.13984028.v1.

If you perform DFT calculations please also cite:

- Manathunga, M.; Miao, Y.; Mu, D.; Götz, A. W.; Merz, K. M. Parallel Implementation of Density Functional Theory Methods in the Quantum Interaction Computational Kernel Program. *J. Chem. Theory Comput.* **16**, 4315–4326 (2020).

If you use the GPU accelerated version of QUICK please also cite:

- Manathunga, M.; Jin, C,; Cruzeiro, V. W. D.; Miao, Y.; Mu, D.; Arumugam, K.; Keipert, K.; Aktulga, H. M.; Merz, K. M., Jr.; Götz, A. W. (2021): Harnessing the Power of Multi-GPU Acceleration into the Quantum Interaction Computational Kernel Program. *ChemRxiv*. Preprint. https://doi.org/10.26434/chemrxiv.13769209.v1

- Mia, Y.; Merz, K. M., Jr. Acceleration of High Angular Momentum Electron Repulsion Integrals and Integral Derivatives on Graphics Processing Units. *J. Chem. Theory Comput.* **11**, 1449-1462 (2015).

The *QUICK* QM/MM features are available in two options: 1) a file-based interface (FBI), see also section 6.2 or 2) an application programming interface (API). As shown in reference [117], the API option provides faster calculations due to speedups with I/O operations and to setup the QM calculations. Therefore, we recommend users to use the API interface. The *QUICK* QM/MM features are optional, which implies that users must choose to compile Amber with QUICK support before they can use it. Please refer to section 5.2 for installation instructions. Additionally, a list of *QUICK* features and limitations has been presented in section 5.1.

Important note: both the FBI and API interfaces of *QUICK* support QM/MM simulations with both mechanical embedding and electrostatic embedding. At present the same limitations and caveats apply with respect to QM/MM cutoffs as for external QM codes (see section 6.2).

6.3.1. Usage

As discussed in chapter 5, SANDER can access different *QUICK* installation types for QM/MM simulations: serial, parallel, CUDA serial, and CUDA parallel. If using the file-based interface (FBI), the *sander* executable is capable calling any of the four different QUICK executables: *quick*, *quick.MPI*, *quick.cuda*, or *quick.cuda.MPI*. If using the application programming interface (API), a different SANDER executable must be used for different *QUICK* types: the serial and MPI parallel versions of QUICK can be accessed from the *sander* and *sander.MPI* executables, respectively; furthermore, the serial GPU-accelerated and multi-GPU-accelerated versions of *QUICK* can be accessed from *sander.quick.cuda* and *sander.quick.cuda.MPI*; these executables are identical to *sander* and *sander.MPI* in all SANDER functionalities, except they perform QM/MM calculations with *QUICK* using the GPU-accelerated code through the API.

Examples for how to use both the API and FBI functionalities can be found at the test suites in the following locations within AMBER's source: *$AMBERHOME/test/qmmm_Quick* and *$AMBERHOME-/test/qmmm_EXTERN/*Quick* for, respectively, API and FBI.

Important note: Before running any QM/MM simulations with *QUICK*, users must make sure to source *$AMBERHOME/amber.sh* (or *$AMBERHOME/amber.csh*, depending on your enviroment). This step ensures that the location of the necessary executables and libraries are set in the environmental variables of the operating system.

6.3.1.1. File-based interface (FBI)

Below is an example of the modifications necessary in the SANDER input file to perform a mechanical embedding QM/MM simulation. In this example, the first two residues of the system are assigned to the QM region, and the simulation is executed at the B3LYP/def2-SVP level with *quick.cuda.MPI* using 2 GPUs:

```
&cntrl
 ...
 ifqnt = 1,
/
&qmmm
 qmmask = ':1-2',
 qm_theory = 'extern',
 qmmm_int = 5,
/
&quick
 method   = 'B3LYP',
 basis    = 'def2-svp',
 executable = 'quick.cuda.MPI',
 do_parallel = 'mpirun -np 2',
/
```

where *ifqnt* set to 1 activates the QM/MM functionality, *qmmask* specifies the QM region, *qm_theory* set as 'extern' indicates that the FBI will be used, and *qmmm_int* set to 5 specifies the use of mechanical embedding. The executable flag can be set to any of the four QUICK executables, and the do_parallel flag must be specified only if using one of the MPI parallel versions of *QUICK*. It is important to emphasize that some machines may require a command other than *mpirun*, depending on the MPI library being used. In general the serial sander executable must be used because of limitations to the system calls from within MPI programs (i.e. you cannot use *sander.MPI* if you want to call an external MPI program). This means that the MM portion of the calculation will be executed in serial.

6.3.1.2. Application programming interface (API)

In the example below, we present the modifications necessary in the SANDER input file to perform a QM/MM simulation with electrostatic embedding. Unlike for the FBI case, simulations using serial, parallel, serial GPU-accelerated, and multi-GPU-accelerated *QUICK* functionalities can all use the same input file.

```
&cntrl
```

```
 ...
 ifqnt = 1,
/
&qmmm
 qmmask = ':1-2',
 qm_theory = 'quick',
 qmmm_int = 1,
 qm_ewald = 0,
/
&quick
 method   = 'B3LYP',
 basis    = 'def2-svp',
/
```

where *ifqnt* set to 1 activates the QM/MM functionality, *qmmask* specifies the QM region, *qm_theory* set as 'quick' makes use of API, *qmmm_int* set to 1 specifies the use of electrostatic embedding and *qm_ewald* set to 0 indicates that the QM/MM interactions should be truncated at a given cutoff (the *cut* variable is specified in the &cntrl namelist). This is currently requied in the same way as when external QM codes are used via the FBI because there is no straight forward way for an Ewald based treatment of long-range QM/MM electrostatics with *ab initio* QM methods.

Note: when the simulation is executed with the API, an output file called *quick.out* (the prefix name for this file can be modified; see below) is generated containing the *QUICK* output information for all MD steps.

6.3.1.3. &quick namelist variables

Below we show a list of all variables that can be specified in the &quick namelist. Please notice that some variables are specific to only the API or the FBI.

`method` **= String** Method to be used in the calculation, can be either 'HF' or some supported DFT functional. (Default: BLYP).

`basis` **= String** Basis set type to be used in the calculation. (Default: 6-31G).

`executable` **= String** ***(FBI only)*** *QUICK* executable to be used in the simulation with the FBI. Options are: *quick*, *quick.MPI*, *quick.cuda*, or *quick.cuda.MPI* (Default: quick).

`do_parallel` **= String** ***(FBI only)*** Portion of the command to be placed right before the executable specification for activating the parallelization. Example: 'mpirun -np 2'. (Default: none).

`scf_cyc` **= Integer** Number of SCF cycles. (Default: 200).

`keywords` **= String** ***(API only)*** Instead of specifying the *QUICK* input variables separetely with the flags above, users can use this flag instead to specify the full keywords line that would go on the top of a *QUICK* input file. Example in a simulation with electrostatic embedding: 'B3LYP BASIS=cc-pVDZ CHARGE=0 MULT=1 GRADIENT EXTCHARGES'. (Default: none).

`outfprefix` **= String** ***(API only)*** Prefix to be used in the *QUICK* output file. The name choosen here will be followed by a '.out' suffix. (Default: quick).

`debug` Debugging information.

= 0 (Default) No debugging information is printed.

= 1 Debugging information is printed.

= 2 Extra debugging information is printed if using the FBI.

`use_template` ***(FBI only)*** Use a template input file.

= 0 (Default) No template file is used.

= 1 Template file is used.

7. Setting up crystal simulations

David S. Cerutti

Simulations of biomolecular crystals are in principle no different than any of the simulations that AMBER does in periodic boundary conditions. However, the setup of these systems is not trivial and probably cannot be accomplished with the LEaP software. Of principal importance are the construction of the solvent conditions (packing precise amounts of multiple solvent species into the simulation cell), and tailoring the unit cell dimensions to accommodate the inherently periodic nature of the system. The LEaP software, designed to construct simulations of molecules in solution, will overlay a pre-equilibrated solvent mask over the (biomolecular) solute, tile that mask throughout the simulation cell, and then prune solvent residues which clash with the solute. The result of this procedure is a system which will likely contract under constant pressure dynamics as the pruning process has left vacuum bubbles at the solute:solvent interface. Simulations of biomolecular crystals require that the simulation cell begin at a size corresponding to the crystallographic unit cell, and deviate very little from that size over the course of equilibration and onset of constant pressure dynamics. This demands a different strategy for placing solvent in the simulation cell. Four programs in the *AmberTools* release are designed to accomplish this. An example of their use in given in a web-based tutorial at *https://ambermd.org/tutorials/advanced/tutorial13/XtalTutor1.html* . A recent (2018) review of crystal simulations is also worth consulting.[134]

For brevity, only basic descriptions of the programs are given in this manual. All of the programs may be run with command line input; the input options to each program may be listed by running each program with no arguments.

7.1. UnitCell

A macromolecular crystal contains many repeating unit cells which stack like blocks in three dimensional space just as simulation cells do in periodic boundary conditions. Each unit cell, in turn, may contain multiple symmetry-related clusters of atoms. A PDB file contains one set of coordinates for the irreducible unit of the crystal, the "asymmetric unit," and also information about the crystal space group and unit cell dimensions. The *UnitCell* program reads PDB files, seeking the SMTRY records within the REMARKs to enumerate the rotation and translation operations which may be applied to the coordinates given in the PDB file to reconstruct one complete unit cell.

Usage of the *UnitCell* program is as follows. The simple command rests on a critical assumption, that the PDB file contains an accurate CRYST1 record and that the REMARK 290 SMTRY records provide its space group symmetry operations.

```
UnitCell -p MyProtein.pdb -o UnitCell.pdb
```

7.2. PropPDB

Simulations in periodic boundary conditions require a minimum unit cell size: the simulation cell must be able to enclose a sphere of at least the nonbonded direct space cutoff radius plus a small buffer region for nonbonded pairlist updates. Many biomolecular crystal unit cells come in "shoebox" dimensions that may have one very short side; many unit cells are also not rectangular but triclinic, meaning that the size of the largest sphere they can enclose is further reduced. The workhorse simulation engine, pmemd.cuda, even requires that the simulation cell be at least three times as thick as the cutoff plus some buffer margin in order to run safely: for typical sum conditions this thickness is about 30Å. For these reasons, and perhaps to ensure that the rigid symmetry imposed by periodic boundary conditions does not create artifacts (crystallographic unit cells are equivalent when averaged over all time and space, but are not necessarily identical at any given moment), it may be necessary to include

multiple unit cells within the simulation cell. This is the purpose of the *PropPDB* program: to propagate a unit cell in one or more directions so that the complete simulation cell meets minimum size requirements.

Drawing on the hypothetical example above, if the unit cell is too small we can extend it in the *x* and *z* dimensions:

```
PropPDB -p UnitCell -o ExpandedCell.pdb -ix 2 -iy 1 -iz 2
```

7.3. AddToBox

The *AddToBox* program handles placement of solvent within a crystal unit cell or supercell (as may be created by PropPDB). As described in the introduction, the basic strategy is to place solvent such that added solvent molecules do not clash with biomolecule solutes, but *may* clash with one another initially. This compromise is necessary because enough solvent must be added to the system to ensure that the correct unit cell dimensions are maintained in the long run, but it is not acceptable to place solvent within the interior of a biomolecule where it might not belong and never escape.

The *AddToBox* program takes a PDB file providing the coordinates of a complete biomolecular unit cell or supercell (argument -c), the dimensions by which that supercell repeats in space (the unit cell dimensions are taken from the CRYST1 record of this file), a PDB file describing the solvent residue to add (argument -a), and the number of copies of that solvent molecule to add (argument -na). *AddToBox* inherently assumes that the biomolecular unit cell it is initially presented may contain some amount of solvent already, and according to the AMBER convention of listing macromolecular solute atoms first and solvent last assumes that the first -P atoms in the file are the protein (or biomolecule). *AddToBox* will then color a very fine grid "black" if the grid point is within a certain distance of a biomolecular atom (argument -RP, default 5.0Å) or other solvent atom (argument -RW, default 1.0Å); the grid is "white" otherwise (the grid is stored in binary for memory efficiency). *AddToBox* will the make a copy of the solvent residue and randomly rotate and translate it somewhere within the unit cell. If all atoms of the solvent residue land on "white" grid voxels, the solvent molecule will become part of the system and the grid around the newly added solvent will be blacked out accordingly. If the solvent molecule cannot be placed, this process will be repeated until a million consecutive failures are encountered, at which point the program will terminate. If *AddToBox* has not placed the requested number of solvent molecules by the time it terminates, the -V option can be used to order the program to recursively call itself with progressively smaller solvent buffer distances until all the requested solvent can be placed. The output of the *AddToBox* program is another PDB named by the -o option.

Successful operation of *AddToBox* may take practice. If multiple solvent species are required, as is the case with heterogeneous crystallization solutions, *AddToBox* may be called repeatedly with each input molecular cell being the previous call's output. When considering crystal solvation, the order of addition is important! It is recommended that rare species, such as trace buffer reagents, be added first, with large -RW argument to ensure that they are dispersed throughout the available crystal void zones. Large solvent species such as MPD (an isohexane diol commonly used in crystallization conditions) should be added second, and with a sufficiently large -RW argument that methyl groups and ring systems cannot become interlocked (which will likely lead to SHAKE / vlimit errors). Small and abundant species such as water should be added last, as they can go anywhere that space remains.

Below is an example of the usage for a hypothetical protein with 5431 atoms and a net charge of +6 that is to be neutralized with ammonium sulfate:

```
AddToBox -c ExpandedCell.pdb -a Sulfate.pdb -na 18 -RP 3.0 -P 5431 -o System.pdb
AddToBox -c System.pdb -a Ammonium.pdb -na 30 -RP 3.0 -P 5431 -o System.pdb
AddToBox -c System.pdb -a Water.pdb -na 1089 -RP 3.0 -P 5431 -o System.pdb
```

The use of the -V flag ensures that the desired amounts of each species are included. The protein clipping radius of 3Å is lower than the default, but safe (remember, this radius stipulates that no solvent atom, regardless of the size of the solvent molecule, come within 3Å of the protein). Note how the original protein PDB file serves as the base for system, but thereafter we work with the System.pdb to accumulate more solvent particles. Here, the

ammonium sulfate serves both to neutralize the system and replicate a salty bath, perhaps from a crystallization mother liquor, hence the break from the usual 2:1 stoichiometry of ammonium sulfate ions.

It is likely that the unobservable "void" regions between biomolecules in most crystals *do not* contain solvent species in proportion to their abundance in the crystallization solution–the vast majority of these regions are within a few Ångstroms of some biomolecular surface, and different biomolecular functional groups will preferentially interact with some types of solvent over others. Also, in many crystals some solvent molecules *are* observed; in many of these, the amount of solvent observed is such that it would be impossible to pack other species into the unit cell in proportion to their abundances in the crystallization fluid. In these cases, we recommend estimating the amount of volume that must be filled with solvent *apart from solvent which has already been observed in the crystal*, and filling this void with solvent in proportion to the composition of the crystallization fluid. For example, if a crystal were grown in a 1:1 mole-to-mole water/ethanol mixture, and the crystal coordinates as deposited in the PDB contained 500 water molecules and 3 ethanol molecules, we would use *AddToBox* to add water and ethanol in a 1:1 ratio until the system contained enough solvent to maintain the correct volume during equilibrium dynamics at constant pressure.

Finally, it is difficult to estimate exactly how much solvent will be needed to maintain the correct equilibrium volume; the advisable approach is simply to make an initial guess and script the setup so that, over multiple runs and reconstructions, the correct system composition can be found. We recommend matching the equilibrium unit cell volume to within 0.3% to keep this simulation parameter within the error of most crystallographic measurements. While errors of 0.5-1% will show up quickly after constant pressure dynamics begin, a 10 to 20ns simulation may be needed to ensure that the correct equilibrium volume has been achieved.

7.4. ChBox

After the complex process of adding solvent, the LEaP program may be used to produce a topology and initial set of coordinates based on the PDB file produced by *AddToBox*. By using the *SetBox* command, LEaP will create a periodic system without adding any more solvent on its own. The only problem with using LEaP at this point is that the program will fail to realize that the system *does* tile in three dimensions if only the box dimensions are set properly. If visualized, the output of UnitCell / PropPDB will likely look jagged, but the output of *AddToBox*, containing lots of added water, will make it obvious how parts of biomolecules jutting out one face of the box fit neatly into open spaces on an opposite face. The topology produced by LEaP needs no editing; only the last line of the coordinates does. This can be done manually, but the *ChBox* program automates the process, taking the same coordinates supplied to AddToBox and grafting them into the input coordinates file.

The program is even unnecessary in the case of orthorhombic (rectangular) unit cells, as this the tleap command will substitute:

```
set [unit] box { <x> <y> <z> }
```

For cells that do not have only 90-degree box angles, *ChBox* will do the trick.

8. sander

8.1. Introduction

This is a guide to *sander*, an Amber module which carries out energy minimization, molecular dynamics, and NMR refinements. The acronym stands for **S**imulated **A**nnealing with **N**MR-**D**erived **E**nergy **R**estraints, but this module is used for a variety of simulations that have nothing to do with NMR refinement. Some of the functionality of sander is available with better computational performance in the pmemd module. In general, sander and pmemd are input compatible. sander inputs for features not supported by pmemd should be properly parsed by pmemd and pmemd should report that the requested feature is not supported. There are a few features available in pmemd that are not supported by sander, see Sections**??** and **??** Some general features are outlined in the following paragraphs:

1. *Sander* provides direct support for several force fields for proteins and nucleic acids, and for several water models and other organic solvents. The basic force field implemented here has the following form, which is about the simplest functional form that preserves the essential nature of molecules in condensed phases:

$$\begin{aligned} V(\mathbf{r}) &= \sum_{bonds} K_b(b-b_0)^2 + \sum_{angles} K_\theta(\theta-\theta_o)^2 \\ &+ \sum_{dihedrals} (V_n/2)(1+\cos[n\phi-\delta] \\ &+ \sum_{nonbij} (A_{ij}/r_{ij}^{12}) - (B_{ij}/r_{ij}^{6}) + (q_iq_j/r_{ij}) \end{aligned}$$

   "Non-additive" force fields based on atom-centered dipole polarizabilities can also be used. These add a "polarization" term to what was given above:

$$E_{pol} = -2\sum_i \mu_i \cdot \mathbf{E}_{io}$$

   where μ_i is an induced atomic dipole. In addition, charges that are not centered on atoms, but are off-center (as for lone-pairs or "extra points") can be included in the force field.

2. The particle-mesh Ewald (PME) procedure (or, optionally, a "true" Ewald sum) is used to handle long-range electrostatic interactions. Long-range van der Waals interactions are estimated by a continuum model. Biomolecular simulations in the NVE ensemble (*i.e.* with Newtonian dynamics) conserve energy well over multi-nanosecond runs without modification of the equations of motion.

3. Two periodic imaging geometries are included: rectangular parallelepiped and truncated octahedron (box with corners chopped off). (*Sander* itself can handle many other periodically-replicating boxes, but input and output support in *LEaP* and *ptraj* is only available right now for these two.) The size of the repeating unit can be coupled to a given external pressure, and velocities can be coupled to a given external temperature by several schemes. The external conditions and coupling constants can be varied over time, so various simulated annealing protocols can be specified in a simple and flexible manner.

4. It is also possible to carry out non-periodic simulations in which aqueous solvation effects are represented *implicitly* by a generalized Born/ surface area model by adding the following two terms to the "vacuum" potential function:

$$\Delta G_{sol} = \sum_{ij}(1-\frac{1}{\varepsilon})(q_i q_j / f_{GB}(r_{ij}) + A\sum_i \sigma_i$$

The first term accounts for the polar part of solvation (free) energy, designed to provide an approximation for the reaction field potential, and the second represents the non-polar contribution which is taken to be proportional to the surface area of the molecule.

5. Users can define internal restraints on bonds, valence angles, and torsions, and the force constants and target values for the restraints can vary during the simulation. The relative weights of various terms in the force field can be varied over time, allowing one to implement a variety of simulated annealing protocols in a single run.

6. Internal restraints can be defined to be "time-averaged", that is, restraint forces are applied based on the averaged value of an internal coordinate over the course of the dynamics trajectory, not only on its current value. Alternatively, restraints can be "ensemble-averaged" using the locally-enhanced-sampling (LES) option.

7. Restraints can be directly defined in terms of NOESY intensities (calculated with a relaxation matrix technique), residual dipolar couplings, scalar coupling constants and proton chemical shifts. There are provisions for handling overlapping peaks or ambiguous assignments. In conjunction with distance and angle constraints, this provides a powerful and flexible approach to NMR structural refinements.

8. Replica exchange calculations can allow simultaneous sampling at a variety of conditions (such as temperature), and allow the user to construct Boltzmann samples in ways that converge more quickly than standard MD simulations. Other variants of biased MD simulations can also be used to improve sampling.

9. Restraints can also be defined in terms of the root-mean-square coordinate distance from some reference structure. This allows one to bias trajectories either towards or away from some target. Free energies can be estimated from non-equilibrium simulations based on targetting restraints.

10. Free energy calculations, using thermodynamic integration (TI) with a linear or non-linear mixing of the "unperturbed" and "perturbed" Hamiltonian, can be carried out. Alternatively, potentials of mean force can be computed using umbrella sampling.

11. The empirical valence bond (EVB) scheme can be used to mix "diabatic" states into a potential that can represent many types of chemical reactions that take place in enzymes.

12. QMMM Calculations where part of the system can be treated quantum mechanically allowing bond breaking and formation during a simulation. Semi-empirical and DFTB Hamiltonians are provided directly within *sander*. More advanced *ab initio* and DFT Hamiltonians are available via an interface to external QM software packages.

13. Nuclear quantum effects can be included through path-integral molecular dynamics (PIMD) simulations, and estimates of quantum time-correlation functions can be computed.

8.2. File usage

```
sander [-help] [-O] [-A] -i mdin -o mdout -p prmtop -c inpcrd -r restrt
-ref refc -mtmd mtmd -x mdcrd -y inptraj -v mdvel -frc mdfrc -e mden
-inf mdinfo -radii radii -cpin cpin -cpout cpout -cprestrt cprestrt
 -cein cein -ceout ceout -cerestrt cerestrt -evbin evbin -suffix suffix
-O  Overwrite output files if they exist.
-A  Append output files if they exist (used mainly for replica exchange).
```

Here is a brief description of the files referred to above; the first five files are used for every run, whereas the remainder are only used when certain options are chosen.

mdin *input* control data for the min/md run

mdout *output* user readable state info and diagnostics -o stdout will send output to stdout (to the terminal) instead of to a file.

mdinfo *output* latest mdout-format energy info

prmtop *input* molecular topology, force field, periodic box type, atom and residue names

inpcrd *input* initial coordinates and (optionally) velocities and periodic box size

refc *input* (optional) reference coords for position restraints; also used for targeted MD

mtmd *input* (optional) containing list of files and parameters for targeted MD to multiple targets

mdcrd *output* coordinate sets saved over trajectory

inptraj *input* coordinate sets in trajectory format, when imin=5

mdvel *output* velocity sets saved over trajectory

mdfrc *output* force sets saved over trajectory

mden *output* extensive energy data over trajectory (not synchronized with mdcrd or mdvel)

restrt *output* final coordinates, velocity, and box dimensions if any - for restarting run

inpdip *input* polarizable dipole file, when indmeth=3

rstdip *output* polarizable dipole file, when indmeth=3

cpin *input* protonation state definitions

cprestrt protonation state definitions, final protonation states for restart (same format as cpin)

cpout *output* protonation state data saved over trajectory

cein *input* redox state definitions

cerestrt redox state definitions, final redox states for restart (same format as cein)

ceout *output* redox state data saved over trajectory

evbin *input* input for EVB potentials

suffix *output* this string will be added to all unspecified output files that are printed (for *multisander* runs, it will append this suffix to all output files)

8.3. Example input files

Here are a couple of sample files, just to establish a basic syntax and appearance. There are more examples of NMR-related files later in this chapter.

1. Simple restrained minimization

```
Minimization with Cartesian restraints
&cntrl
imin=1, maxcyc=200, (invoke minimization)
ntpr=5, (print frequency)
ntr=1, (turn on Cartesian restraints)
restraint_wt=1.0, (force constant for restraint)
restraintmask=':1-58', (atoms in residues 1-58 restrained)
/
```

2. "Plain" molecular dynamics run

```
molecular dynamics run
&cntrl
imin=0, irest=1, ntx=5, (restart MD)
ntt=3, temp0=300.0, gamma_ln=5.0, (temperature control)
ntp=1, taup=2.0, (pressure control)
ntb=2, ntc=2, ntf=2, (SHAKE, periodic bc.)
nstlim=500000, (run for 0.5 nsec)
ntwx=1000, ntpr=200, (output frequency)
/
```

3. Self-guided Langevin dynamics run

```
Self-guided Langevin dynamics run
&cntrl
imin=0, irest=0, ntx=1, (start LD)
ntt=3, temp0=300.0,gamma_ln=1.0, (temperature control)
ntc=3, ntf=3, (SHAKE)
nstlim=500000, (run for 0.5 nsec)
ntwx=1000, ntpr=200, (output frequency)
isgld=1, tsgavg=0.2,tempsg=400.0, (SGLD)
/
```

8.4. Namelist Input Syntax

Namelist provides list-directed input, and convenient specification of default values. It dates back to the early 1960's on the IBM 709, but was regrettably not part of Fortran 77. It is a part of the Fortran 90 standard, and is supported as well by most Fortran 77 compilers (including g77).

Namelist input groups take the form:

```
&name
var1=value, var2=value, var3(sub)=value,
var4(sub,sub,sub)=value,value,
var5=repeat*value,value,
/
```

The variables must be names in the Namelist variable list. The order of the variables in the input list is of no significance, except that if a variable is specified more than once, later assignments may overwrite earlier ones.

Blanks may occur anywhere in the input, except embedded in constants (other than string constants, where they count as ordinary characters).

It is common in older inputs for the ending "/" to be replaced by "&end"; this is non-standard-conforming.

Letter case is ignored in all character comparisons, but case is preserved in string constants. String constants must be enclosed by single quotes (’). If the text string itself contains single quotes, indicate them by two consecutive single quotes, e.g. C1’ becomes ’C1”’ as a character string constant.

Array variables may be subscripted or unsubscripted. An unsubscripted array variable is the same as if the subscript (1) had been specified. If a subscript list is given, it must have either one constant, or exactly as many as the number in the declared dimension of the array. Bounds checking is performed for ALL subscript positions, although if only one is given for a multi-dimension array, the check is against the entire array size, not against the first dimension. If more than one constant appears after an array assignment, the values go into successive locations of the array. It is NOT necessary to input all elements of an array.

Any constant may optionally be preceded by a positive (1,2,3,..) integer repeat factor, so that, for example, 25*3.1415 is equivalent to twenty-five successive values 3.1415. The repeat count separator, *, may be preceded and followed by 0 or more blanks. Valid LOGICAL constants are 0, F, .F., .FALSE., 1, T, .T., and .TRUE.; lower case versions of these also work.

8.5. Overview of the information in the input file

General minimization and dynamics input One or more title lines, followed by the (required) &cntrl and (optional) &pb, &ewald, &qmmm, &amoeba or &debugf namelist blocks. Described in Sections 8.6 and 8.7.

Varying conditions Parameters for changing temperature, restraint weights, etc., during the MD run. Each parameter is specified by a separate &wt namelist block, ending with &wt type="END", /. Described in Section 8.9.

File redirection TYPE=*filename* lines. Section ends with the first non-blank line which does not correspond to a recognized redirection. Described in Section 8.10.

Group information Read if *ntr*, *ibelly* or *idecomp* are set to nonzero values, and if some other conditions are satisfied; see sections on these variables, below. Described in Appendix 9.2.

8.6. General minimization and dynamics parameters

Each of the variables listed below is input in a namelist statement with the namelist identifier &cntrl.cmmu can enter the parameters in any order, using keyword identifiers. Variables that are not given in the namelist input retain their default values. Support for namelist input is included in almost all current Fortran compilers, and is a standard feature of Fortran 90. A detailed description of the namelist convention is given in Appendix A.

In general, namelist input consists of an arbitrary number of comment cards, followed by a record whose first seven characters after a " &" (e.g. " &cntrl ") name a group of variables that can be set by name.cmsys is followed by statements of the form " maxcyc=500, diel=2.0, ... ", and is concluded by an " / " token. The first line of input contains a title, which is then followed by the &cntrl namelist. Note that the first character on each line of a namelist block must be a blank.

Some of the options and variables are much more important, and commonlycmrdified, than are others. We have denoted the "common" options by printing them in **boldface** below. In general, you can skip reading about the non-bold options on a first pass, and you should change these from their defaults only if you think you know what you are doing.

8.6.1. General flags describing the calculation

imin Flag to run minimization.

= 0 (default) Run molecular dynamics without any minimization.

= 1 Perform an energy minimization.

= 5 Read in a trajectory for analysis.

Although sander will write energy information in the output files (using *ntpr*), it is often desirable to calculate the energies of a set of structures at a later point. In particular, one may wish to post-process a set of structures using a different energy function than was used to generate the structures. An example of this is MM-PBSA analysis, where the explicit water is removed and replaced with a continuum model.

If *imin* is set to 5, sander will read a trajectory file (the "inptraj" argument, specified using `-y` on the command line), and will perform the functions described in the mdin file (e.g., an energy minimization) for each of the structures in this file. The final structure from each minimization will be written out to the normal mdcrd file. If you wish to read in a binary (i.e., NetCDF format) trajectory, be sure to set *ioutfm* to 1 (see below). Note that this will result in the output trajectory having NetCDF format as well.

For example, when *imin* = 5 and *maxcyc* = 1000, sander will minimize each structure in the trajectory for 1000 steps and write a minimized coordinate set for each frame to the mdcrd file. If *maxcyc* = 1, the output file can be used to extract the energies of each of the coordinate sets in the inptraj file.

Trajectories containing box coordinates can be post-processed. In order to read trajectories with box coordinates, *ntb* should be greater than 0.

IMPORTANT CAVEAT: The initial coordinates input file used (`-c` <inpcrd>) should be the same as the initial coordinates input file usedto generate the original trajectory. This is because sander sets up parameters for PME from the box coordinates in the initial coordinates input file.

nmropt **= 0** (default) No nmr-type analysis will be done.

= 1 NMR restraints and weight changes will be read.

= 2 NMR restraints, weight changes, NOESY volumes, chemical shifts and residual dipolar restraints will be read.

8.6.2. Nature and format of the input

ntx Option to read the initial coordinates, velocities, and box size from the inpcrd file. Option 1 must be used when one is starting from minimized or model-built coordinates. If an MD restrt file is specified for inpcrd then option 5 is generally used (unless you explicitly wish to ignore the velocities that are present).

= 1 (default) Coordinates, but no velocities, will be read; either formatted (ASCII) files or NetCDF files can be used, as the input file type will be auto-detected.

= 5 Coordinates and velocities will be read from either a NetCDF or a formatted (ASCII) coordinate file. Box information will be read if *ntb* > 0. The velocity information will only be used if *irest* = 1 (see below).

irest Flag to restart a simulation.

= 0 (default) Do not restart the simulation; instead, run as a new simulation. Velocities in the input coordinate file, if any, will be ignored, and the time step count will be set to 0 (unless overridden by *t*; see below).

= 1 Restart the simulation, reading coordinates and velocities from a previously saved restart file. The velocity information is necessary when restarting, so *ntx* (see above) must be 5 (for Amber versions much older than 20, ntx must be greater than or equal to 4), if *irest* = 1.

8.6.3. Nature and format of the output

ntxo Format of the final coordinates, velocities, and box size (if a constant volume or pressure run) written to file "restrt".

= 1 Formatted (ASCII)

= 2 (default) NetCDF file (recommended, unless you have a workflow that requires the formatted form.)

ntpr Every *ntpr* steps, energy information will be printed in human-readable form to files "mdout" and "mdinfo". "mdinfo" is closed and reopened each time, so it always contains the most recent energy and temperature. Default 50.

ntave Every *ntave* steps of dynamics, running averages of average energies and fluctuations over the last *ntave* steps will be printed out. A value of 0 disables this printout. Setting *ntave* to a value 1/2 or 1/4 of *nstlim* provides a simple way to look at convergence during the simulation. Default = 0 (disabled).

ntwr Every *ntwr* steps during dynamics, the “restrt” file will be written, ensuring that recovery from a crash will not be so painful. No matter what the value of *ntwr*, a restrt file will be written at the end of the run, i.e., after *nstlim* steps (for dynamics) or *maxcyc* steps (for minimization). If *ntwr* < 0, a unique copy of the file, “restrt_<nstep>”, is written every abs(*ntwr*) steps. This option is useful if for example one wants to run free energy perturbations from multiple starting points or save a series of restrt files for minimization. Default = nstlim.

iwrap If *iwrap* = 1, the coordinates written to the restart and trajectory files will be "wrapped" into a primary box. This means that for each molecule, its periodic image closest to the middle of the "primary box" (with x coordinates between 0 and a, y coordinates between 0 and b, and z coordinates between 0 and c) will be the one written to the output file. This often makes the resulting structures look better visually, but has no effect on the energy or forces. Performing such wrapping, however, can mess up diffusion and other calculations. If *iwrap* = 0, no wrapping will be performed, in which case it is typical to use *cpptraj* as a post-processing program to translate molecules back to the primary box. For very long runs, setting *iwrap* = 1 may be required to keep the coordinate output from overflowing the trajectory and restart file formats, especially if trajectories are written in ASCII format instead of NetCDF (see also the *ioutfm* option). Default = 0.

ntwx Every *ntwx* steps, the coordinates will be written to the mdcrd file. If *ntwx* = 0, no coordinate trajectory file will be written. Default = 0.

ntwv Every *ntwv* steps, the velocities will be written to the mdvel file. If *ntwv* = 0, no velocity trajectory file will be written. If *ntwv* = -1, velocities will be written to mdcrd, which then becomes a combined coordinate/velocity trajectory file, at the interval defined by *ntwx*. This option is available only for binary NetCDF output (*ioutfm* = 1). Most users will have no need for a velocity trajectory file and so can safely leave *ntwv* at the default. Default = 0. Note that dumping velocities frequently, like forces or coordinates, will introduce potentially significant I/O and communication overhead, hurting both performance and parallel scaling.

`ionstepvelocities` Controls whether to print the half-step-ahead velocities (0, default) or on-step velocities (1). The half-step-ahead velocities can potentially be used to restart calculations, but the on-step velocities correspond to calculated kinetic energy/temperature.

`ntwf` Every *ntwf* steps, the forces will be written to the mdfrc file. If *ntwf* = 0, no force trajectory file will be written. If *ntwf* = -1, forces will be written to the mdcrd, which then becomes a combind coordinate/force trajectory file, at the interval defined by *ntwx*. This option is available only for binary NetCDF output (*ioutfm* = 1). Most users will have no need for a force trajectory file and

so can safely leave *ntwf* at the default. Default = 0. Note that dumping forces frequently, like velocities or coordinates, will introduce potentially significant I/O and communication overhead, hurting both performance and parallel scaling.

ntwe Every *ntwe* steps, the energies and temperatures will be written to file "mden" in a compact form. If *ntwe* = 0 then no mden file will be written. Note that energies in the mden file are not synchronized with coordinates or velocities in the mdcrd or mdvel file(s). Assuming identical *ntwe* and *ntwx* values the energies are one time step before the coordinates (as well as the velocities which are synchronized with the coordinates). Consequently, an mden file is rarely written. Default = 0.

ioutfm The format of coordinate and velocity trajectory files (mdcrd, mdvel and inptraj). As of Amber 9, the binary format used in previous versions is no longer supported; binary output is now in NetCDF trajectory format. Binary trajectory files have many advantages: they are smaller, higher precision, much faster to read and write, and able to accept a wider range of coordinate (or velocity) values than formatted trajectory files.

= 0 Formatted ASCII trajectory

= 1 (default) Binary NetCDF trajectory

ntwprt The number of atoms to include in trajectory files (mdcrd and mdvel). This flag can be used to decrease the size of the these files, by including only the first part of the system, which is usually of greater interest (for instance, one might include only the solute and not the solvent). If *ntwprt* = 0, all atoms will be included.

= 0 (default) Include all atoms of the system when writing trajectories.

> 0 Include only atoms 1 to *ntwprt* when writing trajectories.

idecomp Perform energy decomposition according to a chosen scheme. In former distributions this option was only really useful in conjunction with mm_pbsa, where it is turned on automatically if required. Now, a decomposition of $\langle \partial V / \partial \lambda \rangle$ on a per-residue basis in thermodynamic integration (TI) simulations is also possible.[135] The options are:

= 0 (default) Do not decompose energies.

= 1 Decompose energies on a per-residue basis; 1-4 EEL + 1-4 VDW are added to internal (bond, angle, dihedral) energies.

= 2 Decompose energies on a per-residue basis; 1-4 EEL + 1-4 VDW are added to EEL and VDW.

= 3 Decompose energies on a pairwise per-residue basis; otherwise equivalent to *idecomp* = 1. Not available in TI simulations.

= 4 Decompose energies on a pairwise per-residue basis; otherwise equivalent to *idecomp* = 2. Not available in TI simulations.

If energy decomposition is requested, residues may be chosen by the RRES and/or LRES card. The RES card is used to select the residues about which information is written out. See chapters 11.1 for more information. Use of *idecomp* > 0 is incompatible with *ntr* > 0 or *ibelly* > 0.

8.6.4. Frozen or restrained atoms

ibelly Flag for belly type dynamics. If set to 1, a subset of the atoms in the system will be allowed to move, and the coordinates of the rest will be frozen. The *moving* atoms are specified with *bellymask*. This option is not available when *igb>0*. When belly type dynamics is in use, bonded energy terms, vdW interactions, and direct space electrostatic interactions are *not calculated* for pairs of frozen atoms. Note that this does *not* provide any significant speed advantage. Freezing atoms can be useful for some applications but is maintained primarily for backwards compatibility with older versions of Amber. Most applications should use the *ntr* variable instead to restrain parts of the system to stay close to some initial configuration. Default = 0.

ntr Flag for restraining specified atoms in Cartesian space using a harmonic potential, if ntr > 0. The restrained atoms are determined by the *restraintmask* string. The force constant is given by *restraint_wt*. The coordinates are read in "restrt" format from the "refc" file. Default = 0.

restraint_wt The weight (kcal $\cdot$ mol$^{-1} \cdot$ Å^{-2}) for the positional restraints. The restraint is of the form $k(\Delta x)^2$, where k is the value given by this variable, and Δx is the difference between one of the Cartesian coordinates of a restrained atom and its reference position. There is a term like this for each Cartesian coordinate of each restrainted atom.

restraintmask String that specifies the *restrained* atoms when *ntr=1*.

bellymask String that specifies the *moving* atoms when *ibelly=1*.

The syntax for both *restraintmask* and *bellymask* is given in Section 9.1.1. Note that these mask strings are limited to a maximum of 256 characters.

8.6.5. Energy minimization

maxcyc The maximum number of cycles of minimization. Default = 1.

ncyc If NTMIN is 1 then the method of minimization will be switched from steepest descent to conjugate gradient after NCYC cycles. Default 10.

ntmin Flag for the method of minimization.

= 0 Full conjugate gradient minimization. The first 4 cycles are steepest descent at the start of the run and after every nonbonded pairlist update.

= 1 For NCYC cycles the steepest descent method is used then conjugate gradient is switched on (default).

= 2 Only the steepest descent method is used.

= 3 The XMIN method is used, see Section 10.6.1.

= 4 The LMOD method is used, see Section 10.6.2.

dx0 The initial step length. If the initial step length is too big then will give a huge energy; however the minimizer is smart enough to adjust itself. Default 0.01.

drms The convergence criterion for the energy Derivative: minimization will halt when the Root-Mean-Square of the Cartesian elements of the gradient of the energy is less than this. Default is 10^{-4}kcal $\cdot$ mol$^{-1} \cdot$ Å^{-1}.

8.6.6. Molecular dynamics

nstlim Number of MD-steps to be performed. Default 1.

nscm Flag for the removal of translational and rotational center-of-mass (COM) motion at regular intervals (default is 1000). For non-periodic simulations, after every NSCM steps, translational and rotational motion will be removed. For periodic systems, just the translational center-of-mass motion will be removed. This flag is ignored for belly simulations.

For Langevin dynamics, the *position* of the center-of-mass of the molecule is reset to zero every NSCM steps, but the velocities are not affected. Hence there is no change to either the translation or rotational components of the momenta. (Doing anything else would destroy the way in which temperature is regulated in a Langevin dynamics system.) The only reason to even reset the coordinates is to prevent the molecule from diffusing so far away from the origin that its coordinates overflow the format used in restart or trajectory files.

`t` The time at the start (psec) this is for your own reference and is not critical. Start time is taken from the coordinate input file if IREST=1. Default 0.0.

`dt` The time step (psec). Recommended MAXIMUM is .002 if SHAKE is used, or .001 if it isn't. Note that for temperatures above 300K, the step size should be reduced since greater temperatures mean increased velocities and longer distance traveled between each force evaluation, which can lead to anomalously high energies and system blowup. Default 0.001.
The use of Hydrogen Mass Repartitioning (HMR) (see [136] and references therein for more information), together with SHAKE, allows the time step to be increased in a stable fashion by about a factor of two (up to .004) by slowing down the high frequency hydrogen motion in the system. To use HMR, the masses in the topology file need to be altered before starting the simulation. ParmEd can do this automatically with the HMassRepartition option; see Section **??** .

`nrespa` This variable allows the user to evaluate slowly-varying terms in the force field less frequently. For PME, "slowly-varying" (now) means the reciprocal sum. For generalized Born runs, the "slowly-varying" forces are those involving derivatives with respect to the effective radii, and pair interactions whose distances are greater than the "inner" cutoff, currently hard-wired at 8 Å. If NRESPA>1 these slowly-varying forces are evaluated every *nrespa* steps. The forces are adjusted appropriately, leading to an impulse at that step. If *nrespa*dt* is less than or equal to 4 fs then the energy conservation is not seriously compromised. However if *nrespa*dt* $>$ 4 fs then the simulation becomes less stable. Note that energies and related quantities are only accessible every *nrespa* steps, since the values at other times are meaningless.

8.6.7. Temperature regulation

Note: Flag "ntt" is used for the temperature regulation in the default thermostat scheme as shown below. The "middle" thermostat scheme [Section 8.6.10] is much more efficient than the default scheme to accurately sample the configuration/conformation space in the molecular dynamics simulation for the NVT ensemble. Please read Section 8.6.10 for more details.

`ntt` Switch for temperature scaling. Note that setting *ntt*=0 corresponds to the microcanonical (NVE) ensemble (which should approach the canonical one for large numbers of degrees of freedom). Some aspects of the "weak-coupling ensemble" (*ntt*=1) have been examined, and roughly interpolate between the microcanonical and canonical ensembles.[137, 138] The *ntt*=2 and 3 options correspond to the canonical (constant T) ensemble.

- **= 0** Constant total energy classical dynamics (assuming that *ntb*<2, as should probably always be the case when *ntt=0*).
- **= 1** Constant temperature, using the weak-coupling algorithm.[139] A single scaling factor is used for all atoms. Note that this algorithm just ensures that the total kinetic energy is appropriate for the desired temperature; it does nothing to ensure that the temperature is even over all parts of the molecule. Atomic collisions will tend to ensure an even temperature distribution, but this is not guaranteed, and there are many subtle problems that can arise with weak temperature coupling.[140] Using *ntt=1* is especially dangerous for generalized Born simulations, where there are no collisions with solvent to aid in thermalization.) Other temperature coupling options (especially *ntt*=3) should be used instead.
- **= 2** Andersen-like temperature coupling scheme,[141] in which imaginary "collisions" randomize the velocities to a distribution corresponding to *temp0* every *vrand* steps. Note that in between these "massive collisions", the dynamics is Newtonian. Hence, time correlation functions (etc.) can be computed in these sections, and the results averaged over an initial canonical distribution. Note also that too high a collision rate (too small a value of *vrand*) will slow down the speed at which the molecules explore configuration space, whereas too low a rate means that the canonical distribution of energies will be sampled slowly. A discussion of this rate is given by Andersen.[142] Note that this option is not equivalent to the original thermostat described by Andersen[142].

= 3 Use Langevin dynamics with the collision frequency γ given by *gamma_ln*, discussed below. Note that when γ has its default value of zero, this is the same as setting *ntt* = 0. Since Langevin simulations are highly susceptible to "synchronization" artifacts,[143, 144] you should explicitly set the *ig* variable (described below) to a different value at each restart of a given simulation.

= 9 Optimized Isokinetic Nose-Hoover chain ensemble (OIN) [145, 146]. Constant temperature simulation utilizing Nose-Hoover chains and an isokinetic constraint on the particle and thermostat velocities, implemented for use in multiple time-stepping methods, namely for 3D-RISM and RESPA. Stabilizes and smooths particle dynamics and mitigates resonance instabilities, allowing for larger intermediate times steps, up to 16 fs for RESPA (*nrespa*=16 for *dt*=0.001) and 8 fs for 3D-RISM MTS size (*rismnrespa*=8). Each atom is coupled to three Nose-Hoover chains per atom and the thermostat coupling constant (relaxation time) is determined from 1/*gamma_ln*, hence *gamma_ln* must be > 0 if *ntt*=9 invoked. Variable *nkija* specifies the number of substeps of *dt* to use for integrating the equations of motion and *idistr* specifies the frequency at which the thermostat velocity distribution functions are accumulated (if > 0). Such functions are written at frequency *ntpr*. Two additional files containing the thermostat and chain restart velocities, *tfreeze.rst* and *vfreez.rst*, are written at frequency *ntwr*.

= 10 Stochastic Isokinetic Nose-Hoover RESPA integrator [147]. A novel isokinetic integrator developed by Tuckerman and co-workers that invokes an isokinetic constraint on the particle velocities combined with *nkija* (see below) auxiliary thermostat velocities *v1* and *v2*. The integrator includes a stochastic component in the equations of motion, which introduces white noise into the system, for the purpose of minimizing resonance instabilities in the velocities, ultimately allowing for larger RESPA steps. The isokinetic constraint has the form $mv^2 + \frac{L}{L+1}\sum_{i=1}^{L} Q_1 v_{1i}^2 = Lk_BT$. Here L is the number of additional thermostat degrees of freedom, defined in AMBER as *nkija* (see below), and $Q_1$is the thermostat mass, determined from *sinrtau* (below), v is the particle velocity and v_1 is one of two auxiliary velocities (e.g. thermostat velocities), and m, k_B, and T, are the particle mass, Boltzmann constant, and system temperature (*temp0*), respectively. In using this integrator, the system is placed in the isokinetic ensemble, as such the velocities are NOT canonical and no thermodynamic observables can be derived from them. This will lead to anomalous temperature readings throughout the simulation - for 1 thermostat degree of freedom (L = *nkija* = 1) the temperature will appear about one-half the specified temperature (*temp0*), and with additional thermostat DOF, the temperature will approach, but never exceed, the desired temperature, *temp0*. However, the particle coordinates ARE canonical and it can be said the configurations obtained from a simulation were sampled from a Boltzmann distribution at the specified temperature (*temp0*).

= 11 Stochastic version of Berendsen thermostat, also known as Bussi thermostat [148]. This thermostat samples canonical distribution by scaling all velocities to a random temperature probed from canonical distribution. Collision frequency with thermostat is controled by *tautp*.

temp0 Reference temperature at which the system is to be kept, if *ntt* > 0. Note that for temperatures above 300K, the step size should be reduced since increased distance traveled between evaluations can lead to SHAKE and other problems. Default 300.

temp0les This is the target temperature for all LES particles (see Chapter 6). If *temp0les*<0, a single temperature bath is used for all atoms, otherwise separate thermostats are used for LES and non-LES particles. Default is -1, corresponding to a single (weak-coupling) temperature bath.

tempi Initial temperature. For the initial dynamics run, (*ntx* = 1 or for Amber versions much older than 20, ntx < 3) the velocities are assigned from a Maxwellian distribution at *tempi* K. If *tempi* = 0.0, the velocities will be calculated from the forces instead. *tempi* has no effect if *ntx* = 5 (for Amber versions much older than 20, if ntx > 3). Default 0.0.

`ig` The seed for the pseudo-random number generator. The MD starting velocity is dependent on the random number generator seed if *tempi* is nonzero and *ntx* = 1 (for Amber versions much older than 20, if ntx < 3). The value of this seed also affects the set of pseudo-random values used for Langevin dynamics or Andersen-like coupling, and hence should be set to a different value on each restart if *ntt* = 2 or 3. If *ig* = -1 (the default) then the random seed will be based on the current date and time, and hence will be different for every run. Unless you specifically desire reproducibility, it is recommended that you set *ig* = -1 for all runs involving *ntt* = 2 or 3.

`tautp` Time constant, in ps, for heat bath coupling for the system, if *ntt* = 1. Default is 1.0. Generally, values for TAUTP should be in the range of 0.5-5.0 ps, with a smaller value providing tighter coupling to the heat bath and, thus, faster heating and a less natural trajectory. Smaller values of TAUTP result in smaller fluctuations in kinetic energy, but larger fluctuations in the total energy. Values much larger than the length of the simulation result in a return to constant energy conditions.

`gamma_ln` The collision frequency γ, in ps^{-1}, when *ntt* = 3. Default is 0. A simple Leapfrog integrator is used to propagate the dynamics, with the kinetic energy adjusted to be correct for the harmonic oscillator case.[149, 150] Note that it is not necessary that γ approximate the physical collision frequency, which is about 50 ps^{-1} for liquid water. In fact, it is often advantageous, in terms of sampling[150, 151] or stability of integration[152], to use much smaller values, around 2 to 5 ps^{-1}.[150, 152] For implicit solvent (GB), even much lower values may be useful: for example, setting gamma_ln to 0.01 ps^{-1} can lead to significant, up to 100-fold in some cases, speedup of conformational sampling.[5] Also used to determine thermostat coupling constant for the Optimized Isokinetic Nose-Hoover chain integrator (OIN, ntt=9), which is equal to 1/*gamma_ln* [146], so the specified *gamma_ln* must be > 0. A *gamma_ln* of 10 ps^{-1} represents a coupling constant of 100 fs. For *ntt=10*, this is the friction constant associated with the stochastic component of the integrator, essentially serving the same role as in the Langevin integrator [147]. This parameter is required for *ntt=10* and must be > 0.

`vrand` If *vrand>0* and *ntt=2*, the velocities will be randomized to temperature TEMP0 every *vrand* steps. Default is 1000.

`vlimit` If not equal to 0.0, then any component of the velocity that is greater than abs(VLIMIT) will be reduced to VLIMIT (preserving the sign). This can be used to avoid occasional instabilities in molecular dynamics runs. VLIMIT should generally be set to a value like 20 (the default), which is well above the most probable velocity in a Maxwell-Boltzmann distribution at room temperature. A warning message will be printed whenever the velocities are modified. Runs that have more than a few such warnings should be carefully examined.

`nkija` For use with *ntt*=9 and *ntt=10*., For *ntt=9*, this the number of substeps of *dt* when integrating the thermostat equations of motion, for greater accuracy. For *ntt=10*, this specifies the number of additional auxiliary velocity variables *v1* and *v2*, which will total nkija$\times$*v1* +nkija$\times$*v2 [147].* Default is 1 for both integrators.

`idistr` For the isokinetic integrator (*ntt*=9), the frequency at which the thermostat velocity distribution functions are accumulated.

`sinrtau` For the SINR (Stochastic Isokinetic Nose-Hoover RESPA) integrator (*ntt=10*), this specifies the time scale for determining the masses associated with the two auxiliary velocity variables *v1* and *v2* (e.g. thermostat velocities) and hence the magnitude of the coupling of the physical velocities with the auxiliary velocities. Generally this should be related to the time scale of the system. See [147] for more explanation. Default is 1.0.

8.6.8. Pressure regulation

In "constant pressure" dynamics, the volume of the unit cell is adjusted (by small amounts on each step) to make the computed pressure approach the target pressure, *pres0*. Equilibration with *ntp* > 0 is generally necessary to

adjust the density of the system to appropriate values. Note that fluctuations in the instantaneous pressure on each step will appear to be large (several hundred bar), but the average value over many steps should be close to the target pressure. Pressure regulation only applies when Constant Pressure periodic boundary conditions are used (*ntp > 0*). The two available pressure coupling algorithms available in Amber are of the "weak-coupling" variety, analogous to temperature coupling,[139] and the use of the Monte Carlo barostat. While the Berendsen barostat yields the correct target density, it does not strictly sample from the isothermal-isobaric ensemble and typically yields volume fluctuations that are too low. The Monte Carlo barostat, on the other hand, samples rigorously from the isobaric-isothermal ensemble and does not necessitate computing the virial. Please note: in general you will need to equilibrate the temperature to something like the final temperature using constant volume (*ntp*=0) *before* switching on constant pressure simulations to adjust the system to the correct density. If you fail to do this, the program will try to adjust the density too quickly, and bad things (such as SHAKE failures) are likely to happen.

ntp Flag for constant pressure dynamics. This option should be set to 1 or 2 when Constant Pressure periodic boundary conditions are used.

- **= 0** No pressure scaling (Default)
- **= 1** md with isotropic position scaling
- **= 2** md with anisotropic (x-,y-,z-) pressure scaling: this should only be used with orthogonal boxes (i.e. with all angles set to 90 degrees). Anisotropic scaling is primarily intended for non-isotropic systems, such as membrane simulations, where the surface tensions are different in different directions; it is generally not appropriate for solutes dissolved in water.)[146] Anisotropic pressure scaling can also be applied to just one specified direction (x, y or z) with the directional pressure scaling option (baroscalingdir > 0). In this case the box scales along the one chosen direction only, and its dimensions along the other two directions remain fixed. This type of directional pressure control option can be useful in situations where one needs to keep the solvent box unchanged along two direction, while still maintaining a constant pressure in the system. For example, a phase boundary can be created by placing two boxes from different simulations in contact with each other along a common face, which can be useful for simulating phase transitions such as water to ice[153].
- **= 3** md with semiisotropic pressure scaling: this is only available with constant surface tension (csurften > 0) and orthogonal boxes. This links the pressure coupling in the two directions tangential to the interface.

barostat Flag used to control which barostat to use in order to control the pressure.

- **= 1** Berendsen (Default)
- **= 2** Monte Carlo barostat

`mcbarint` Number of steps between volume change attempts performed as part of the Monte Carlo barostat. Default is 100.

pres0 Reference pressure (in units of bars, where 1 bar $\approx$ 0.987 atm) at which the system is maintained (when NTP $>$ 0). Default 1.0.

`comp` compressibility of the system when NTP $>$ 0. The units are in 1.0×10^{-6} bar^{-1}; a value of 44.6 (default) is appropriate for water.

taup Pressure relaxation time (in ps), when NTP $>$ 0. The recommended value is between 1.0 and 5.0 psec. Default value is 1.0, but larger values may sometimes be necessary (if your trajec)[146]tories seem unstable).

baroscalingdir Flag for pressure scaling direction control. Applicable when using Monte Carlo barostat (barostat = 2) with anisotropic pressure scaling (ntp = 2).

- **= 0** box size scales randomly (x, y or z) each scaling step (default)
- **= 1** box scales only along x-direction, dimensions along y-, z-axes are fixed

= 2 box scales only along y-direction, dimensions along x-, z-axes are fixed

= 3 box scales only along z-direction, dimensions along x-, y-axes are fixed

Surface tension regulation

Constant surface tension is used in statistical ensembles for simulating liquid interfaces. This is primarily intended for lipid membrane simulations with two or more interfaces. Constant surface tension is only available for simulations with anisotropic pressure or semiisotropic scaling. This algorithm is an extension to the Berendsen pressure scaling algorithm that adjusts the tangential pressure evaluation in order to maintain a "constant" surface tension.[154] Since the surface tension is a function of the pressure tensor, fluctuations of the surface tension will be large.

In order to use constant surface tension, periodic boundary conditions (ntb = 2), anisotropic or semiisotropic pressure scaling (ntp = 2 or ntp =3), and an orthogonal box must be used.

csurften Flag for constant surface tension dynamics.

= 0 No constant surface tension (default)

= 1 Constant surface tension with interfaces in the yz plane

= 2 Constant surface tension with interfaces in the xz plane

= 3 Constant surface tension with interfaces in the xy plane

gamma_ten Surface tension value in units of dyne/cm. Default value is 0.0 dyne/cm.

ninterface Number of interfaces in the periodic box. There must be at least two interfaces in the periodic box. Two interfaces is appropriate for a lipid bilayer system and is the default value.

8.6.9. SHAKE bond length constraints

ntc Flag for SHAKE to perform bond length constraints.[155] (See also NTF in the **Potential function** section. In particular, typically NTF = NTC.) The SHAKE option should be used for most MD calculations. The size of the MD timestep is determined by the fastest motions in the system. SHAKE removes the bond stretching freedom, which is the fastest motion, and consequently allows a larger timestep to be used. For water models, a special "three-point" algorithm is used.[156] Consequently, to employ TIP3P set NTF = NTC = 2.

Since SHAKE is an algorithm based on dynamics, the minimizer is not aware of what SHAKE is doing; for this reason, minimizations generally should be carried out without SHAKE. One exception is short minimizations whose purpose is to remove bad contacts before dynamics can begin.

For parallel versions of *sander* only intramolecular atoms can be constrained. Thus, such atoms must be in the same chain of the originating PDB file.

= 1 SHAKE is not performed (default)

= 2 bonds involving hydrogen are constrained

= 3 all bonds are constrained (not available for parallel or qmmm runs in *sander*)

tol Relative geometrical tolerance for coordinate resetting in shake. Recommended maximum: <0.00005 Angstrom Default 0.00001.

jfastw Fast water definition flag. By default, the system is searched for water residues, and special routines are used to SHAKE these systems.[156]

= 0 Normal operation. Waters are identified by the default names (given below), unless they are redefined, as described below.

= 4 Do not use the fast SHAKE routines for waters.

The following variables allow redefinition of the default residue and atom names used by the program to determine which residues are waters.

WATNAM The residue name the program expects for water. Default 'WAT '.

OWTNM The atom name the program expects for the oxygen of water. Default 'O '.

HWTNM1 The atom name the program expects for the 1st H of water. Default 'H1 '.

HWTNM2 The atom name the program expects for the 2nd H of water. Default 'H2 '.

`noshakemask` String that specifies atoms that are not to be shaken (assuming that ntc>1). Any bond that would otherwise be shaken by virtue of the *ntc* flag, but which involves an atom flagged here, will *not* be shaken. The syntax for this string is given in Chap. 13.5. Default is an empty string, which matches nothing. A typical use would be to remove SHAKE constraints from all or part of a solute, while still shaking rigid water models like TIPnP or SPC/E. Another use would be to turn off SHAKE constraints for the parts of the system that are being changed with thermodynamic integration, or which are the EVB or quantum regions of the system.

If this option is invoked, then all parts of the potential must be evaluated, that is, *ntf* must be one. The code enforces this by setting *ntf* to 1 when a *noshakemask* string is present in the input.

If you want the *noshakemask* to apply to all or part of the water molecules, you must also set *jfastw=4*, to turn off the special code for water SHAKE. (If you are not shaking waters, you presumably also want to issue the "set default FlexibleWater on" command in LEaP; see that chapter for more information.)

8.6.10. The "middle" scheme

8.6.10.1. Introduction

The "middle" scheme offers a unified framework to develop efficient thermostatting algorithms for configurational sampling for the canonical ensemble, as described in Refs. [157–161]. It can be implemented for performing molecular dynamics (MD) or path integral molecular dynamics (PIMD), either with or without holonomic constraints. The "middle" scheme allows the use of much larger time intervals (i.e., time stepsizes) Δt to maintain the same accuracy, which significantly improves the configurational sampling efficiency. That is, it is efficient for calculating structural properties and thermodynamic observables that depend on coordinate variables. Most thermostats control the temperature by updating momenta of the system. Some prevailing thermostats include stochastic ones (such as the Andersen thermostat and Langevin dynamics) and deterministic ones (such as the Nosé-Hoover thermostat and Nosé-Hoover chain). In the "middle" scheme, immediately after the coordinate-updating step for half a time interval, the thermostat process for a full time interval takes place, which is then followed by the coordinate-updating step for another half time interval[157, 161].

Here we present a brief introduction to the "middle" scheme. For many thermostats, the integration in one time step Δt can be splitted into three parts, the steps for updating coordinates, momenta and thermostat, denoted as "x", "p" and "T", respectively. In this case the "equation of motion" may be expressed as

$$\begin{bmatrix} \mathrm{d}\mathbf{x}_t \\ \mathrm{d}\mathbf{p}_t \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{M}^{-1}\mathbf{p}_t \\ 0 \end{bmatrix} \mathrm{d}t}_{\mathrm{x}} + \underbrace{\begin{bmatrix} 0 \\ -\nabla_{\mathbf{x}_t} U(\mathbf{x}_t) \end{bmatrix} \mathrm{d}t}_{\mathrm{p}} + \underbrace{\begin{bmatrix} \text{thermostat} \end{bmatrix}}_{\mathrm{T}} \tag{8.1}$$

Here, U is the potential energy, $\mathbf{M}$ is the diagonal mass matrix, $\mathbf{x}$ and $\mathbf{p}$ are the vectors of coordinate and momentum, respectively. Equation (8.1) is, however, not convenient to do the analysis.

A more useful approach is to employ the forward Kolmogorov equation to express the evolution of the density distribution in the phase space $\rho(\mathbf{x},\mathbf{p})$.

$$\begin{aligned} \frac{\partial}{\partial t}\rho &= \mathscr{L}\rho \\ &= (\mathscr{L}_{\mathrm{x}} + \mathscr{L}_{\mathrm{p}} + \mathscr{L}_{\mathrm{T}})\rho \end{aligned} \tag{8.2}$$

The relevant Kolmogorov operators for the 1st and 2nd terms of the right-hand side (RHS) are

$$\mathscr{L}_{\mathrm{x}}\rho = -\mathbf{p}^T\mathbf{M}^{-1}\nabla_{\mathbf{x}}\rho \tag{8.3}$$

$$\mathscr{L}_{\mathrm{p}}\rho = \nabla_{\mathbf{x}}U(\mathbf{x})\cdot\nabla_{\mathbf{p}}\rho \tag{8.4}$$

respectively. The definition of $\mathscr{L}_{\mathrm{T}}$ depends on the specific thermostat. The phase space propagators for a time interval Δt for the three parts are $e^{\mathscr{L}_{\mathrm{x}}\Delta t}$, $e^{\mathscr{L}_{\mathrm{p}}\Delta t}$, and $e^{\mathscr{L}_{\mathrm{T}}\Delta t}$, respectively.

The propagation in each time step with the velocity Verlet (VV) algorithm is performed as

$$e^{\mathscr{L}\Delta t} \approx e^{\mathscr{L}^{\mathrm{VV}}_{\mathrm{middle}}\Delta t} = e^{\mathscr{L}_{\mathrm{p}}\Delta t/2}e^{\mathscr{L}_{\mathrm{x}}\Delta t/2}e^{\mathscr{L}_{\mathrm{T}}\Delta t}e^{\mathscr{L}_{\mathrm{x}}\Delta t/2}e^{\mathscr{L}_{\mathrm{p}}\Delta t/2} \tag{8.5}$$

The phase space propagator for the thermostat part $e^{\mathscr{L}_{\mathrm{T}}\Delta t}$ is designed in the middle. Equation (8.5) is denoted as "VVMiddle". The numerical algorithm reads

$$\begin{aligned}
\text{Update Momenta for half a step:} \quad & \mathbf{p} \leftarrow \mathbf{p} - \frac{\partial U}{\partial \mathbf{x}}\frac{\Delta t}{2} \\
\text{Update Coordinates for half a step:} \quad & \mathbf{x} \leftarrow \mathbf{x} + \mathbf{M}^{-1}\mathbf{p}\frac{\Delta t}{2} \\
\text{Thermostat for a full time step:} \quad & \textit{thermostat_step} \\
\text{Update Coordinates for another half step:} \quad & \mathbf{x} \leftarrow \mathbf{x} + \mathbf{M}^{-1}\mathbf{p}\frac{\Delta t}{2} \\
\text{Update Momenta for another half step:} \quad & \mathbf{p} \leftarrow \mathbf{p} - \frac{\partial U}{\partial \mathbf{x}}\frac{\Delta t}{2}
\end{aligned} \tag{8.6}$$

where *thermostat_step* is the subroutine for the thermostat process, which is determined according to the thermostat method of choice.

The stationary state distribution of "VVMiddle" for a harmonic system $U(\mathbf{x}) = \frac{1}{2}(\mathbf{x}-\mathbf{x}_{\mathrm{eq}})^T\mathbf{A}(\mathbf{x}-\mathbf{x}_{\mathrm{eq}})$ is

$$\begin{aligned}
\rho^{\mathrm{VV}}_{\mathrm{middle}}(\mathbf{x},\mathbf{p}) = & \frac{1}{Z_N}\exp\left\{-\beta\left[\frac{1}{2}\mathbf{p}^T\left(\mathbf{M}-\mathbf{A}\frac{\Delta t^2}{4}\right)^{-1}\mathbf{p}\right.\right. \\
& \left.\left.+\frac{1}{2}(\mathbf{x}-\mathbf{x}_{\mathrm{eq}})^T\mathbf{A}(\mathbf{x}-\mathbf{x}_{\mathrm{eq}})\right]\right\}
\end{aligned} \tag{8.7}$$

as long as the thermostat process keeps the Maxwell (or Maxwell-Boltzmann) momentum distribution unchanged, i.e.

$$e^{\mathscr{L}_{\mathrm{T}}\Delta t}\rho_{\mathrm{MB}}(\mathbf{p}) = \rho_{\mathrm{MB}}(\mathbf{p}) \tag{8.8}$$

where the Maxwell momentum distribution is

$$\rho_{\mathrm{MB}}(\mathbf{p}) = \left(\frac{\beta}{2\pi}\right)^{3N/2}|\mathbf{M}|^{-1/2}\exp\left[-\beta\left(\frac{1}{2}\mathbf{p}^T\mathbf{M}^{-1}\mathbf{p}\right)\right] \tag{8.9}$$

Here $\beta = \frac{1}{k_B T}$ with k_B as the Boltzmann constant, T is the temperature of the system. (N is the number of particles.) It is then easy to verify that the marginal distribution of coordinates for "VVMiddle" is exact in the harmonic limit. Many types of thermostats satisfy the criteria (thermostat process keeps the Maxwell momentum distribution unchanged, Equation (8.8)), which include, but not limited to, the thermostats listed below.

- Andersen thermostat (real dynamics case)
  In this thermostat, each particle of the system stochastically collides with a fictitious heat bath, and once the collision occurs, the momentum of this particle is chosen afresh from the Maxwell-Boltzmann momentum

distribution. The explicit form for the thermostat process can be expressed as

$$\begin{aligned}\mathbf{p}^{(j)} &\leftarrow \sqrt{\frac{m_j}{\beta}}\boldsymbol{\theta}_j,\ (j=\overline{1,N}) \\ &\text{if } \mu_j < \nu\Delta t \text{ (or more precisely } \mu_j < 1-e^{-\nu\Delta t})\end{aligned} \tag{8.10}$$

Here ν is the collision frequency, $\boldsymbol{\theta}_j$ is a vector of independent Gaussian-distributed random numbers with zero mean and unit variance, m_j the mass for the jth atom, μ_j is a uniformly distributed random number in the range (0,1). Here μ_j may be different for each particle ($j=\overline{1,N}$). In the current version of AMBER μ_j is the same for all particles. (I.e., the global Andersen thermostat is employed.)

The phase space propagator for the thermostat process is

$$\begin{aligned}e^{\mathscr{L}_{\mathrm{T}}\Delta t}\rho =& e^{-\nu\Delta t}\rho(\mathbf{x},\mathbf{p}) \\ &+(1-e^{-\nu\Delta t})\rho_{\mathrm{MB}}(\mathbf{p})\int_{-\infty}^{\infty}\rho(\mathbf{x},\mathbf{p})\mathrm{d}\mathbf{p}\end{aligned} \tag{8.11}$$

- Andersen thermostat (virtual dynamics case)
  The explicit form for the virtual dynamics case of the Andersen thermostat is expressed as

$$\left.\begin{aligned}\mathbf{p}^{(j)} &\leftarrow \sqrt{\frac{m_j}{\beta}}\boldsymbol{\theta}_j, \text{ if } \mu_j < 1-e^{-\nu\Delta t} \\ \mathbf{p}^{(j)} &\leftarrow -\mathbf{p}^{(j)}, \quad \text{otherwise}\end{aligned}\right\} (j=\overline{1,N}) \tag{8.12}$$

  The phase space propagator for the thermostat process is

$$\begin{aligned}e^{\mathscr{L}_{\mathrm{T}}\Delta t}\rho =& e^{-\nu\Delta t}\rho(\mathbf{x},-\mathbf{p}) \\ &+(1-e^{-\nu\Delta t})\rho_{\mathrm{MB}}(\mathbf{p})\int_{-\infty}^{\infty}\rho(\mathbf{x},\mathbf{p})\mathrm{d}\mathbf{p}\end{aligned} \tag{8.13}$$

- Langevin dynamics (real dynamics case)
  The thermostat process is the solution to the Ornstein-Uhlenbeck (OU) process

$$\mathbf{p} \leftarrow e^{-\boldsymbol{\gamma}\Delta t}\mathbf{p} + \sqrt{\frac{1}{\beta}}\mathbf{M}^{1/2}(\mathbf{1}-e^{-2\boldsymbol{\gamma}\Delta t})^{1/2}\boldsymbol{\eta} \tag{8.14}$$

  Here, $\boldsymbol{\eta}$ is a vector of independent Gaussian-distributed random numbers with zero mean and unit variance, $\boldsymbol{\gamma}$ is the diagonal friction coefficient matrix. In the current version of AMBER all diagonal elements of $\boldsymbol{\gamma}$ are set to be the same. (That is, the friction coefficient is the same for all particles. The global Langevin thermostat is used.)
  The phase space propagator for the thermostat process is

$$\begin{aligned}e^{\mathscr{L}_{\mathrm{T}}\Delta t}\rho =& \left(\frac{\beta}{2\pi}\right)^{3N/2}|\mathbf{M}(\mathbf{1}-e^{-2\boldsymbol{\gamma}\Delta t})|^{-1/2} \\ &\cdot\int \mathrm{d}\mathbf{p}_0\,\rho(\mathbf{x},\mathbf{p}_0)\exp\left[-\frac{\beta}{2}(\mathbf{p}-e^{-\boldsymbol{\gamma}\Delta t}\mathbf{p}_0)^T\right. \\ &\left.\cdot\mathbf{M}^{-1}(\mathbf{1}-e^{-2\boldsymbol{\gamma}\Delta t})^{-1}(\mathbf{p}-e^{-\boldsymbol{\gamma}\Delta t}\mathbf{p}_0)\right]\end{aligned} \tag{8.15}$$

- Langevin dynamics (virtual dynamics case)
  The virtual dynamics case represents another type of discrete evolution that may not correspond to a contin-

uous, real dynamical counterpart of the Langevin equation.

$$\mathbf{p} \leftarrow -e^{-\boldsymbol{\gamma}\Delta t}\mathbf{p} + \sqrt{\frac{1}{\beta}}\mathbf{M}^{1/2}(\mathbf{1}-e^{-2\boldsymbol{\gamma}\Delta t})^{1/2}\boldsymbol{\eta} \tag{8.16}$$

The virtual dynamics case is also able to produce the desired stationary distribution. The phase space propagator for the thermostat process is then

$$\begin{aligned} e^{\mathscr{L}_{\mathrm{T}}\Delta t}\rho = &\left(\frac{\beta}{2\pi}\right)^{3N/2}|\mathbf{M}(\mathbf{1}-e^{-2\boldsymbol{\gamma}\Delta t})|^{-1/2} \\ &\cdot\int \mathrm{d}\mathbf{p}_0\,\rho(\mathbf{x},\mathbf{p}_0)\exp\left[-\frac{\beta}{2}(\mathbf{p}+e^{-\boldsymbol{\gamma}\Delta t}\mathbf{p}_0)^T\right. \\ &\left.\cdot\mathbf{M}^{-1}(\mathbf{1}-e^{-2\boldsymbol{\gamma}\Delta t})^{-1}(\mathbf{p}+e^{-\boldsymbol{\gamma}\Delta t}\mathbf{p}_0)\right] \end{aligned} \tag{8.17}$$

- Nosé-Hoover (NH) thermostat and Nosé-Hoover chain (NHC)
  See Ref. [157] for more detailed discussions.

The "middle" scheme of a thermostat includes both real and virtual dynamics cases. (See Refs. [159, 160].) It is proved in Ref. [159] that, while the Langevin equation algorithm (BAOAB) proposed in Ref. [162] is simply only the real dynamics case of "VVMiddle", another Langevin equation algorithm proposed (without employing the Lie-Trotter splitting) in Ref. [163] is equivalent to "VVMiddle" for Langevin dynamics. The real dynamics case for the Andersen thermostat and that for Langevin dynamics have been implemented in the current version of AMBER.

When the leapfrog algorithm, rather than the velocity-Verlet algorithm, is employed in the "middle" scheme, it is denoted as "LFMiddle"[161]. The propagation in each time step with the leapfrog (LF) algorithm is performed as

$$e^{\mathscr{L}\Delta t} \approx e^{\mathscr{L}^{\mathrm{LF}}_{\mathrm{middle}}\Delta t} = e^{\mathscr{L}_{\mathrm{x}}\Delta t/2}e^{\mathscr{L}_{\mathrm{T}}\Delta t}e^{\mathscr{L}_{\mathrm{x}}\Delta t/2}e^{\mathscr{L}_{\mathrm{p}}\Delta t} \tag{8.18}$$

The numerical algorithm of "LFMiddle" reads

$$\begin{aligned} \text{Update Momenta for a full time step:} &\quad \mathbf{p} \leftarrow \mathbf{p} - \frac{\partial U}{\partial \mathbf{x}}\Delta t \\ \text{Update Coordinates for half a step:} &\quad \mathbf{x} \leftarrow \mathbf{x} + \mathbf{M}^{-1}\mathbf{p}\frac{\Delta t}{2} \\ \text{Thermostat for a full time step:} &\quad \textit{thermostat\_step} \\ \text{Update Coordinates for another half step:} &\quad \mathbf{x} \leftarrow \mathbf{x} + \mathbf{M}^{-1}\mathbf{p}\frac{\Delta t}{2} \end{aligned} \tag{8.19}$$

For any general systems, "LFMiddle" shares the same accuracy as "VVMiddle" for sampling the marginal distribution of coordinates. In addition, "LFMiddle" leads to the exact marginal distribution of momenta in the harmonic limit[161]. For simplicity and compatibility, only "LFMiddle" is integrated into AMBER.

The "middle" scheme with holonomic constraints (such as bond length constraints) is also implemented. While MD with holonomic constraints are widely used in biological simulations, PIMD with holonomic constraints may help understand nuclear quantum effects of different motions in molecular systems. For instance, help assign spectral features as shown in Ref. [164]. In the "middle" scheme, when holonomic constraints are applied, the SHAKE [155] and RATTLE [165] algorithms are used for fixing coordinates and velocities, respectively. Particularly for the molecular system that contains water molecules, the analytical SETTLE algorithm [156] may be used to apply the constraint to the water molecule.

Besides Refs. [157–161], one more paper is in preparation for the "middle" scheme [166].

While the "middle" scheme for PIMD with the staging transformation (staging PIMD) was first demonstrated in Ref. [158], that for PIMD with the normal-mode transformation (normal-mode PIMD) was first proposed in the supplemental material of Ref. [158] in 2016, which can be found via the URLs provided by the publisher:

- https://aip.scitation.org/doi/suppl/10.1063/1.4954990/suppl_file/supplemental+material-submitted.docx
- ftp://ftp.aip.org/epaps/journ_chem_phys/E-JCPSA6-145-007626

In addition, the arXiv preprint (that includes Ref. [158] and its supplemental material) is also available (https://arxiv.org/ftp/arxiv/papers/1611/1611.06331.pdf). In the current version, the "middle" scheme is implemented for the primitive version of PIMD (PRIMPIMD) of AMBER. The staging PIMD or normal-mode PIMD algorithms in the "middle" scheme will also be implemented into AMBER soon.

8.6.10.2. Input parameters

In order to perform MD or PRIMPIMD simulations with the "middle" scheme, three additional flags should be added in the *mdin* file, which are used to distinguish different methods.

ischeme Flag for choosing an integration scheme for molecular dynamics.

=0 (default) conventional scheme in AMBER.

=1 "middle" scheme based on the leapfrog algorithm.

ithermostat Flag for different thermostats when the "middle" scheme is employed. Two types of thermostats are currently available.

=1 Langevin dynamics.

=2 Andersen thermostat.

therm_par The parameter used in a thermostatting method of the "middle" scheme, in the unit of ps^{-1}, which should always be a positive number. It refers to the friction coefficient for Langevin dynamics (*ithermostat* = 1) or the collision frequency for the Andersen thermostat (*ithermostat* = 2).

The recommended value for *therm_par* is related to the characteristic frequency ($\tilde{\omega}$) of the specific system. The characteristic time of the potential energy autocorrelation function is

$$\tau_{UU} = \int_0^\infty \frac{\langle U(0)U(t)\rangle - \langle U\rangle^2}{\langle U^2\rangle - \langle U\rangle^2}\,\mathrm{d}t \tag{8.20}$$

The optimal value of the thermostat parameter that produces the minimum correlation time of the potential is $\xi^{opt} \approx \tilde{\omega}$ for Langevin dynamics and $\xi^{opt} \approx \sqrt{2}\tilde{\omega}$ for the Andersen thermostat, as the time interval Δt approaches zero. E.g. for a HO molecule, the frequency of the O-H stretch is around 3600 cm^{-1}(680 ps^{-1}), so one can choose 680 ps^{-1} as the value of *therm_par* when Langevin dynamics is used, or 960 ps^{-1} when the Andersen thermostat is employed. When the time interval Δt is finite in the two thermostatting methods, while the characteristic correlation time goes to infinity as the thermostat parameter approaches zero, the characteristic correlation time gradually reaches a plateau as the thermostat parameter increases. (Please see Refs. [159–161] for more discussions.) When condensed phase systems are simulated, it is not straightforward to estimate the optimal thermostat parameter(s) that could be related to the mixing of frequencies (or time scales) of the system [142]. Some numerical tests are necessary for obtaining the reasonable region for the thermostat parameter such that the characteristic time divided by the time interval is relatively small. (This is true not only for the "middle" scheme but for all thermostat algorithms.) For a liquid water system (216 water molecules in a cell with periodic boundary conditions) with no holonomic constraints, the thermostat parameter is usually chosen to be $2-50$ ps^{-1}.

8.6.10.3. Examples

Examples include a liquid water system (216 water molecules in a cell with the periodic boundary conditions) with the q-SPC/Fw model, an alanine dipeptide (ACE-ALA-NME) solved in a box with 401 methol molecules, and a peptide chain ACE-ALA-ALA-ALA-NME in vacuum. One is also encouraged to check the test cases in *$AMBERHOME/test/middle-scheme*. In AMBER the analytical SETTLE algorithm is the default (*jfastw*=0) for applying the constraints for the water molecule. (Note that in some liquid water models, the intramolecular H-H

is specified as a bond in the topology file, so all the intramolecular O-H and H-H distances are constrained when *ntc*=2 is employed in AMBER.)

Molecular dynamics (for classical statistics)

(1) MD input using the Langevin thermostat with the "LFMiddle" scheme for liquid water.
Test: $AMBERHOME/test/middle-scheme/MD_Unconstr_Langevin_water

```
MD: NVT simulation of liquid water
&cntrl
ipimd = 0, nstlim = 10        ! MD for 10 steps
ntx = 1, irest = 0            ! read coordinates
temp0 = 300, tempi = 300      ! temperature: target and initial
dt = 0.001                    ! time step in ps
cut = 7.0                     ! non-bond cut off
ischeme = 1                   !! leapfrog middle scheme
ithermostat = 1               !! Langevin thermostat
therm_par = 5.0               !! thermostat parameter in 1/ps
ig = 1000                     ! random seed
ntc = 1, ntf = 1              ! no constraints
ntpr = 1, ntwr = 5, ntwx = 5  ! output settings
/
```

One can run either a serial job (using *sander*):

```
$ sander -O -i md_LGV.in -p qspcfw216.top -c nvt.rst -o md_LGV.out \
  -r lgv.rst -info lgv.info
```

or a parallel job (using *sander.MPI*):

```
$ mpirun -np 4 sander.MPI -O -i md_LGV.in -p qspcfw216.top -c nvt.rst \
  -o md_LGV.out -r lgv.rst -info lgv.info
```

(2) MD input using Langevin dynamics with the "LFMiddle" scheme for the liquid water. Lengths of the bonds having hydrogen atoms are constrained.
Test: $AMBERHOME/test/middle-scheme/MD_Constr_Langevin_water

```
MD: NVT simulation of liquid water
&cntrl
ipimd = 0, nstlim = 10        ! MD for 10 steps
ntx = 1, irest = 0            ! read coordinates
temp0 = 300, tempi = 300      ! temperature: target and initial
dt = 0.004                    ! time step in ps
cut = 7.0                     ! non-bond cut off
ischeme = 1,                  !! leapfrog middle scheme
ithermostat = 1,              !! Langevin thermostat, random seed is default value
therm_par = 5.0               !! thermostat parameter, in 1/ps
ntc = 2, ntf = 2              ! constrain lengths of the bonds having hydrogen atoms
ntpr = 1, ntwr = 5, ntwx = 5  ! output settings
/
```

Run either a serial way (using *sander*):

```
$ sander -O -i md_LGV.in -p qspcfw216.top -c nvt.rst -o md_LGV.out \
  -r lgv.rst -info lgv.info
```

or a parallel job (using *sander.MPI*):

```
$ mpirun -np 4 sander.MPI -O -i md_LGV.in -p qspcfw216.top -c nvt.rst \
  -o md_LGV.out -r lgv.rst -info lgv.info
```

Path integral molecular dynamics (for quantum statistics)

(1) PRIMPIMD input using the Andersen thermostat with the "LFMiddle" scheme for the liquid water. Lengths of the bonds having hydrogen atoms are constrained.

Test: $AMBERHOME/test/middle-scheme/PIMD_Constr_Andersen_water

```
PRIMPIMD: NVT simulation of liquid water
&cntrl
ipimd = 1, nstlim = 10   ! PRIMPIMD for 10 steps
ntx = 5, irest = 0       ! read coordinates
temp0 = 300, tempi = 300 ! target temperature and initial temperature
dt = 0.002               ! time step in ps
cut = 7.0                ! non-bond cut-off
ischeme = 1,             !! leapfrog middle scheme
ithermostat = 2,         !! Andersen thermostat
therm_par = 8.0          !! thermostat parameter, in 1/ps
ig = 777                 ! random seed
ntc = 2,ntf = 2          ! constrain lengths of the bonds having hydrogen atoms
ntpr=1, ntwr=5, ntwx=5   ! output settings
/
```

(2) PRIMPIMD input using Langevin dynamics with the "LFMiddle" scheme for the liquid water. No constraints are applied.

Test: $AMBERHOME/test/middle-scheme/PIMD_Langevin_water

```
PRIMPIMD: NVT simulation of liquid water
&cntrl
ipimd = 1,nstlim = 10   ! PRIMPIMD for 10 steps
ntx = 5,irest = 0       ! read coordinates,
                        ! and run as a new simulation.
temp0 = 300,tempi = 300 ! target and initial temperature
dt = 0.001              ! time step, in ps
cut = 7.0               ! non-bond cut off
ischeme = 1,            !! leapfrog middle scheme
ithermostat = 1,        !! Langevin thermostat
therm_par = 5.0         !! thermostat parameter, in 1/ps
ntc = 1                 ! no constraints, default
ntpr=1, ntwr=5, ntwx=5  ! output settings
/
```

When one runs PIMD in AMBER, a groupfile is needed. The groupfile *gf_pimd* may look like:

```
-O -i pimd.in -p qspcfw216.top -c nvt1.rst -o bead1.out -r bead1.rst
 -x bead1.mdcrd -inf bead1.mdinfo
-O -i pimd.in -p qspcfw216.top -c nvt2.rst -o bead2.out -r bead2.rst
 -x bead2.mdcrd -inf bead2.mdinfo
-O -i pimd.in -p qspcfw216.top -c nvt3.rst -o bead3.out -r bead3.rst
 -x bead3.mdcrd -inf bead3.mdinfo
-O -i pimd.in -p qspcfw216.top -c nvt4.rst -o bead4.out -r bead4.rst
 -x bead4.mdcrd -inf bead4.mdinfo
```

Note that each line starts with "-O" and ends with "-inf <info>". The groupfile above contains 4 lines, which means 4 path integral beads are used.

sander.MPI is executed via the following command:

```
$ mpirun -np 8 sander.MPI -ng 4 -groupfile gf_pimd
```

The number of processes (8) that is specified by "-np" is a multiple of the number of groups (4). In this case 2 CPU processes are used on each path integral bead.

QM/MM molecular dynamics

QM/MM MD input using the Langevin thermostat with the "LFMiddle" scheme for the alanine dipeptide solved in methol box. Lengths of the bonds having hydrogen atoms are constrained for the MM part, while no constraints are applied to the QM part.

Test: $AMBERHOME/test/middle-scheme/QMMM_Constr_ALA_Methol

```
constrained MD NVT: Alanine dipeptide in meoh (explicit solvent).
&cntrl
ipimd = 0, nstlim = 10,    ! MD for 10 steps
irest = 0, ntx = 1,        ! read coordinates
temp0 = 300                ! target temperature
tempi = 300                ! initial temperature
dt = 0.002,                ! time step, in ps
cut = 8,                   ! non-bond cut off
ig = 6666,                 ! random seed for reproducing results
ischeme = 1,               !! leapfrog middle scheme
ithermostat = 1            !! Langevin thermostat
therm_par = 5.0,           !! thermostat parameter, in 1/ps
ntc=2,ntf=2                ! constrain lengths of bonds having hydrogen atoms atoms
ntpr=1, ntwr=1, ntwx=1     ! output settings
ifqnt=1                    ! switch on QM/MM coupled potential
/
&qmmm qmmask=':ACE,ALA,NME', ! residues treated using QM
qmcharge=0,             ! charge on QM region is 0
qmshake=0,              ! no SHAKE for QM region
qm_theory='PM3',        ! use the PM3 semi-empirical Hamiltonian
qmcut=8.0               ! use 8 angstrom cut off for QM region
/
```

One can run either a serial job (using *sander*):

```
$ sander -O -i qmmm.in -p ala.top -c ala.crd -o qmmm.out -r qmmm.rst \
  -x qmmm.crd -info qmmm.info
```

or a parallel job (using *sander.MPI*):

```
$ mpirun -np 4 sander.MPI -O -i qmmm.in -p ala.top -c ala.crd \
  -o qmmm.out -r qmmm.rst -x qmmm.crd -info qmmm.info
```

Replica exchange molecular dynamics

REMD input using the Langevin thermostat with the "LFMiddle" scheme for the ACE-ALA-ALA-ALA-NME in vacuum.Lengths of the bonds having hydrogen atoms are constrained.

Test: $AMBERHOME/test/middle-scheme/REMD_Constr_ALA

Below is the input file for one of the replicas. The target temperatures are 300, 325, 350, and 400K for the 4 replicas, respectively.

```
REMD test with 4 replicas
&cntrl
imin = 0, nstlim = 100,     ! MD for 100 steps
irest=1, ntx = 5,           ! read coordinates and velocities
tempi = 0.0, temp0 = 300.0, ! initial and target temperature
ischeme= 1,                 !! leapfrog middle scheme
ithermostat = 1,            !! Langevin thermostat
therm_par = 1.0,            !! thermostat parameter,in 1/ps
dt = 0.002,                 ! time step, in ps
ig=6666,                    ! random seed
```

```
ntc = 2, ntf = 2,           ! constrain lengths of the bonds having hydrogen atoms
ntwx = 50, ntwr =50, ntpr = 50,   ! output setting
ntb=0,                      ! no periodicity
cut = 99.0,                 ! non bond cut off
numexchg=5,                 ! exchange frequency
&end
```

When one runs REMD in AMBER, a groupfile is needed. The groupfile *groupfile* may look like:

```
-O -rem 1 -remlog rem.log -i rem.in.000 -p ala3.top -c mdrestrt -o rem.out.000
 -r rem.r.000 -inf reminfo.000
-O -rem 1 -remlog rem.log -i rem.in.001 -p ala3.top -c mdrestrt -o rem.out.001
 -r rem.r.001 -inf reminfo.001
-O -rem 1 -remlog rem.log -i rem.in.002 -p ala3.top -c mdrestrt -o rem.out.002
 -r rem.r.002 -inf reminfo.002
-O -rem 1 -remlog rem.log -i rem.in.003 -p ala3.top -c mdrestrt -o rem.out.003
 -r rem.r.003 -inf reminfo.003
```

Note that each line starts with "-O" and ends with "-inf <info>". The groupfile has 4 lines, which means 4 replicas are employed in REMD.

sander.MPI is executed via the following command:

```
$ mpirun -np 4 sander.MPI -ng 4 -groupfile groupfile
```

The number of processes (4) that is specified by "-np" can be replaced by any multiple of the number of replicas used in REMD (4 in this case).

One is also encouraged to access the tutorial for the "middle" scheme on the webpage
http://jianliugroup.pku.edu.cn/tutorials.html

8.6.11. Water cap

`ivcap` Flag to control cap option. The "cap" refers to a spherical portion of water centered on a point in the solute and restrained by a soft half-harmonic potential. For the best physical realism, this option should be combined with *igb=10*, in order to include the reaction field of waters that are beyond the cap radius.

- **= 0** Cap will be in effect if it is in the *prmtop* file (default).
- **= 1** With this option, a cap can be excised from a larger box of water. For this, cutcap (i.e., the radius of the cap), xcap, ycap, and zcap (i.e., the location of the center of the cap) need to be specified in the &cntrl namelist. Note that the cap parameters must be chosen such that the whole solute is covered by solvent. Solvent molecules (and counterions) located outside the cap are ignored. Although this option also works for minimization and dynamics calculations in general, it is intended to post-process snapshots in the realm of MM-PBSA to get a linear-response approximation of the solvation free energy, output as 'Protein-solvent interactions'.
- **= 2** Cap will be inactivated, even if parameters are present in the *prmtop* file.
- **= 5** With this option, a shell of water around a solute can be excised from a larger box of water. For this, cutcap (i.e., the thickness of the shell) needs to be specified in the &cntrl namelist. Solvent molecules (and counterions) located outside the cap are ignored. This option only works for a single-step minimization. It is intended to post-process snapshots in the realm of MM-PBSA to get a linear-response approximation of the solvation free energy, output as 'Protein-solvent interactions'.

`fcap` The force constant for the cap restraint potential.

`cutcap` Radius of the cap, if *ivcap=1* is used.

`xcap,ycap,zcap` Location of the cap center, if *ivcap=1* is used.

8.6.12. NMR refinement options

(Users to should consult the section NMR refinement to see the context of how the following parameters would be used.)

`iscale` Number of additional variables to optimize beyond the 3N structural parameters. (Default = 0). At present, this is only used with residual dipolar coupling and CSA or pseudo-CSA restraints.

`noeskp` The NOESY volumes will only be evaluated if mod(nstep, noeskp) = 0; otherwise the last computed values for intensities and derivatives will be used. (default = 1, i.e. evaluate volumes at every step)

`ipnlty` This parameter determines the the functional form of the penalty function for NOESY volume and chemical shift restraints.

= 1 the program will minimize the sum of the absolute values of the errors; this is akin to minimizing the crystallographic R-factor (default).

= 2 the program will optimize the sum of the squares of the errors.

= 3 For NOESY intensities, the penalty will be of the form $awt[I_c^{1/6} - I_o^{1/6}]^2$. Chemical shift penalties will be as for *ipnlty=1*.

`mxsub` Maximum number of submolecules that will be used. This is used to determine how much space to allocate for the NOESY calculations. Default 1.

`scalm` "Mass" for the additional scaling parameters. Right now they are restricted to all have the same value. The larger this value, the slower these extra variables will respond to their environment. Default 100 amu.

`pencut` In the summaries of the constraint deviations, entries will only be made if the penalty for that term is greater than PENCUT. Default 0.1.

`tausw` For noesy volume calculations (*NMROPT = 2*), intensities with mixing times less that TAUSW (in seconds) will be computed using perturbation theory, whereas those greater than TAUSW will use a more exact theory. See the theory section (below) for details. To always use the "exact" intensities and derivatives, set TAUSW = 0.0; to always use perturbation theory, set TAUSW to a value larger than the largest mixing time in the input. Default is TAUSW of 0.1 second, which should work pretty well for most systems.

8.6.13. EMAP restraints

EMAP restraints are used to perform targeted conformational search (TCS)[167]. EMAP uses maps to define restraints to maintain conformations and/or to induce simulation systems to the target conformations. The restraint map can be either obtained from electron microscopy experiments or derived from known protein structures, or defined from initial simulation coordinates. EMAP can be used to do rigid docking of molecules into maps and to do flexible fitting to obtain conformations defined by experimental maps. EMAP can also be used to maintain conformations of protein domains when studying large scale conformational change. Users should consult the section 13.1 to see how to define EMAP restraints.

`iemap` Turn on EMAP restrained simulation when *iemap*>0. (Default = 0). EMAP restraint information must be input from *&emap* namelists in the input file.

`gammamap` Friction constant for the EMAP restraint maps when allowed to move. (Default=1/ps). (See Section 13.1)

8.7. Potential function parameters

The parameters in this section generally control what sort of force field (or potential function) is used for the simulation.

8.7.1. Generic parameters

ntf Force evaluation. Note: If SHAKE is used (see NTC), it is not necessary to calculate forces for the constrained bonds.

= 1 complete interaction is calculated (default)

= 2 bond interactions involving H-atoms omitted (use with NTC=2)

= 3 all the bond interactions are omitted (use with NTC=3)

= 4 angle involving H-atoms and all bonds are omitted

= 5 all bond and angle interactions are omitted

= 6 dihedrals involving H-atoms and all bonds and all angle interactions are omitted

= 7 all bond, angle and dihedral interactions are omitted

= 8 all bond, angle, dihedral and non-bonded interactions are omitted

ntb This variable controls whether or not periodic boundaries are imposed on the system during the calculation of non-bonded interactions. Bonds spanning periodic boundaries are not yet supported. There is no longer any need to set this variable, since it can be determined from igb and ntp parameters. The "proper" default for ntb is chosen (ntb=0 when igb > 0, ntb=2 when ntp > 0, and ntb=1 otherwise). This behavior can be overridden by supplying an explicit value, although this is discouraged to prevent errors. The allowed values for NTB are

= 0 no periodicity is applied and PME is off (default when igb > 0)

= 1 constant volume (default when igb and ntp are both 0, which are their defaults)

= 2 constant pressure (default when ntp > 0)

If NTB is nonzero then there must be a periodic boundary in the topology file. Constant pressure is not used in minimization (IMIN=1, above).

For a periodic system, constant pressure is the only way to equilibrate density if the starting state is not correct. For example, the solvent packing scheme used in LEaP can result in a net void when solvent molecules are subtracted which can aggregate into "vacuum bubbles" in a constant volume run. Another potential problem are small gaps at the edges of the box. The upshot is that almost every system needs to be equilibrated at constant pressure (*ntb=2, ntp>0*) to get to a proper density. But be sure to equilibrate first (at constant volume) to something close to the final temperature, before turning on constant pressure.

`dielc` Dielectric multiplicative constant for the electrostatic interactions. Default is 1.0. Please note this is NOT related to dielectric constants for generalized Born or Poisson-Boltzmann calculations. It should only be used for quasi-vacuum simulations.

cut This is used to specify the nonbonded cutoff, in Angstroms. For PME, the cutoff is used to limit direct space sum, and 8.0 is usually a good value. When *igb>0*, the cutoff is used to truncate nonbonded pairs (on an atom-by-atom basis); here a larger value than the default is generally required. A separate parameter (**RGBMAX**) controls the maximum distance between atom pairs that will be considered in carrying out the pairwise summation involved in calculating the effective Born radii, see the generalized Born section below.
When *igb > 0*, the default is 9999.0 (effectively infinite)
When *igb==0*, the default is 8.0.

`fswitch` When off, *fswitch<=0* , uses a truncation cutoff. When on *fswitch>0*, sets a force switching region where the force cutoff smoothly approaches 0 between the region of the fswitch value to the cut value. Force values below the fswitch value follow the standard Lennard-Jones force. Default is -1. This option is not supported for use with GB (i.e., only *igb=0* and *ntb>0*), nor is it compatible with the 12-6-4 Lennard-Jones model (*lj1264=1*). Due to performance regressions (about 20%) with running with the force switching on, it is recommended that simulations run with fswitch off unless using a force field that requires or recommends using the force switch.

`nsnb` Determines the frequency of nonbonded list updates when *igb=0* and *nbflag=0*; see the description of *nbflag* for more information. Default is 25.

`ipol` When set to 1, use a polarizable force field. See Section 8.7.5 for more information. Default is 0.

`ipgm` When set to 1, use the polarizable Gaussian Multipole force field. See Section 8.8 for more information. Default is 0.

`ifqnt` Flag for QM/MM run; if set to 1, you must also include a &qmmm namelist. See Section 6.4 for details on this option. Default is 0.

`igb` Flag for using the generalized Born implicit solvent models. See Chapter 2 for information about using this option. Default is 0.

`ipb` Flag for using the Poisson-Boltzmann implicit solvent models. See Chapter **??** for information about using this option. Default is 0.

`irism` Flag for 3D-reference interaction site model (RISM) molecular solvation method. See Section3.4for information about this option. Default is 0.

`ievb` If set to 1, use the empirical valence bond method to compute energies and forces. See Section 6.3 for information about this option. Default is 0.

`iamoeba` Flag for using the *amoeba* polarizable potentials of Ren and Ponder.[168, 169] When this option is set to 1, you need to prepare an amoeba namelist with additional parameters. Also, the *prmtop* file is built in a special way. See Section **??** for more information about this option. Default is 0.

`lj1264` In general, you should rarely have to set this variable. When the Lennard-Jones C-coefficient is found in your prmtop file, the default value is set to 1 (meaning it is active). When this flag is *not* present in the prmtop file, the default value is set to 0 (meaning the 12-6-4 potential [170] is inactive). Setting this to 0 when the C-coefficient is present will forcibly turn off the 12-6-4 potential. Setting lj1264 to 1 when no C-coefficient is present will result in a fatal error. Therefore, this flag can be used to quickly disable the r^{-4} term. However, the remaining L-J parameters will still be optimized for the 12-6-4 potential, so this should only be done when testing! It currently only supports *sander* and *pmemd* (both the serial and MPI versions) but not *pmemd.cuda*. It is currently only compatible with the Particle Mesh Ewald method for long-range electrostatics. For more information please see Section **??**. For adding it to your topology file, see Subsection **??**.

`efx` This sets the x component of the electric field in kcal/(mol*A*e). Electric fields are naturally off if efx, efy, efz are 0. Default value is 0. It currently only supports *pmemd* (both the serial and MPI versions).

`efy` This sets the y component of the electric field in kcal/(mol*A*e). Electric fields are naturally off if efx, efy, efz are 0. Default value is 0. It currently only supports *pmemd* (both the serial and MPI versions).

`efz` This sets the z component of the electric field in kcal/(mol*A*e). Electric fields are naturally off if efx, efy, efz are 0. Default value is 0. It currently only supports *pmemd* (both the serial and MPI versions).

`efn` If efn is on (efn=1), the x, y, z (efx, efy, efz) components are scaled to box size. For example efx/x length of box size, efy/y length of box size, efz/z length of box size. This normalizes the electric field charge to your box size. It is off when it is 0. It currently only supports *pmemd* (both the serial and MPI versions).

`efphase` efphase sets the timestep phase for the electric field using the equation$\cos((2\pi \times efreq/1000)(dt \times step) - (\pi \times efphase/180))$. It currently only supports *pmemd* (both the serial and MPI versions).

`effreq` effreq sets the timestep frequency for the electric field using the equation $\cos((2\pi \times effreq/1000)(dt \times step) - (\pi \times efphase/180))$. It currently only supports *pmemd* (both the serial and MPI versions).

`mcwat` controls the Monte Carlo (MC) water equilibration function. Set 1 to run, 0 otherwise. mcint, mcrescyc, mcwatmaxdif, and mcboxshift are variables control the frequency and functionality of this feature. Currently only supported on *pmemd* and *pmemd.cuda.*

`mcint` Number of MD steps between each cycle of MC. Preliminary recommendation is 1,000.

`mcrescyc` Number of MC move attempts in each MC cycle. Preliminary recommendation is 10,000-100,000.

`mcwatmaxdif` Sets the maximum absolute difference for MC acceptance between old and new energy in kcal/-mol (recommended value 100). This variable is intended to prevent artifacts from numerical "rollover", where an energy is so high, due to a severe clash at the trial water position, that Fortran rolls it over to a large negative value which would be accepted by the Metropolis criterion.

`mcboxshift` Trims the region in which waters are moved away from the edges of the simulation box, to reduce the number of uninteresting "bulk to bulk" moves, and instead focus on moves connecting bulk with the solute at the middle of the box.. If the system was prepared with cubic periodic boundary conditions with an equal amount of solvent padding along all three axes, it is recommended that this value be set to amount of padding (default is 10 Angstroms).

`ramdboost` Sets default random boost acceleration for ramd (default 1). This boost is multiplied by the mass of each atom in the ligand to determine the force each atom receives. This value is in internal acceleration units refer to the Amber units. Ramd is a pmemd and pmemd.cuda only feature and does not support MPI.

`ramdboostfreq` Sets number of steps between each time ramd boost strength is increased (defualt 0).

`ramdboostrate` Sets the amount to increase the ramdboost acceleration each time ramdboostfreq (default is 0).

`ramdint` Sets the time step interval to apply ramd boost on to the ligand (default is 0).

`ramdmaxdist` Determines the end condition for the simulation (ramd terminates when nstlim is reached or when ramdmaxdist is satisfied). ramdmaxdist is the amount of angstrom displacement from initial center of mass distance of protein and ligand to when this displacement increases by ramdmaxdist.

`ramdligmask` Amber selection mask for what is considered the ligand that needs to be boosted in ramd.

`ramdprotmask` Amber selection mask for what is considered the protein that is used to calculate the distance the ligand has moved.

`reweight` Allows the re-evaluation of trajectories (usually with a new parameter file). Set 1 to turn on. When running this command, in the topology command of the run file (-c) place the trajectory instead of the topology file. This supports netcdf only. Do note if matching against an older run, this does not capture step 0 because step 0 usually evaluates off of the topology which the trajectory generally does not contain. It is recommended to rerun with the same parameter file to check if the feature is

working as intended before proceeding with a modified parameter topology file as Amber has a lot of features and not all of them were tested for this feature (was primarily written for TI calculation reweighting). Reweight is supported in pmemd and pmemd.cuda, and does not support MPI.

`midpoint` Turns on midpoint optimizations (usage of 3-D spatial decomposition). 1 is on, 0 is off (default). This switch is currently experimental. Please consult ambermd.org/intel/midpoint.htm for currently supported features and advanced user compilations. Currently only supported on *pmemd.MPI.*

8.7.2. Particle Mesh Ewald

The Particle Mesh Ewald (PME) method is always "on", unless *ntb* = 0. PME is a fast implementation of the Ewald summation method for calculating the full electrostatic energy of a unit cell (periodic box) in a macroscopic lattice of repeating images. The PME method is fast since the reciprocal space Ewald sums are B-spline interpolated on a grid and since the convolutions necessary to evaluate the sums are calculated via fast Fourier transforms (FFTs). Note that the accuracy of the PME method is related to the density of the charge grid (NFFT1, NFFT2, and NFFT3), the spline interpolation order (ORDER), and the direct sum tolerance (DSUM_TOL); see the descriptions below for more information.

The PME method was implemented originally in Amber 3a by Tom Darden and has been developed in subsequent versions by many people, in particular by Tom Darden, Celeste Sagui, Tom Cheatham and Mike Crowley.[171–174] Generalizations of this method to systems with polarizable dipoles and electrostatic multipoles are described in Refs. [175, 176].

The &`ewald` namelist is read immediately after the `&cntrl` namelist. We have tried hard to make the defaults for these parameters appropriate for solvated simulations. *Please take care in changing any values from their defaults.* The `&ewald` namelist has the following variables:

`nfft1, nfft2, nfft3` These give the size of the charge grid (upon which the reciprocal sums are interpolated) in each dimension. Higher values lead to higher accuracy (when the DSUM_TOL is also lowered) but considerably slow the calculation. Generally it has been found that reasonable results are obtained when NFFT1, NFFT2 and NFFT3 are approximately equal to A, B and C, respectively, leading to a grid spacing (A/NFFT1, etc.) of 1.0 Å. Significant performance enhancement in the calculation of the fast Fourier transform is obtained by having each of the integer NFFT1, NFFT2 and NFFT3 values be a *product of powers* of 2, 3, and/or 5. If the values are not given, the program will chose values to meet these criteria.

`order` The order of the B-spline interpolation. The higher the order, the better the accuracy (unless the charge grid is too coarse). The minimum order is 3. An order of 4 (the default) implies a cubic spline approximation which is a good standard value. Note that the cost of the PME goes as roughly the order to the third power.

`verbose` Standard use is to have VERBOSE = 0. Setting VERBOSE to higher values (up to a maximum of 3) leads to voluminous output of information about the PME run.

`ew_type` Standard use is to have EW_TYPE = 0 which turns on the particle mesh ewald (PME) method. When EW_TYPE = 1, instead of the approximate, interpolated PME, a *regular* Ewald calculation is run. The number of reciprocal vectors used depends upon RSUM_TOL, or can be set by the user. The exact Ewald summation is present mainly to serve as an accuracy check allowing users to determine if the PME grid spacing, order and direct sum tolerance lead to acceptable results. Although the cost of the exact Ewald method formally increases with system size at a much higher rate than the PME, it may be faster for small numbers of atoms (< 500). For larger, macromolecular systems, with > 500 atoms, the PME method is significantly faster.

`dsum_tol` This relates to the width of the direct sum part of the Ewald sum, requiring that the value of the direct sum at the Lennard-Jones cutoff value (specified in CUT as during standard dynamics) be less than DSUM_TOL. In practice it has been found that the relative error in the Ewald forces

(RMS) due to cutting off the direct sum at CUT is between 10.0 and 50.0 times DSUM_TOL. Standard values for DSUM_TOL are in the range of 10^{-6} to 10^{-5} , leading to estimated RMS deviation force errors of 0.00001 to 0.0005. Default is 10^{-5}.

`rsum_tol` This serves as a way to generate the number of reciprocal vectors used in an Ewald sum. Typically the relative RMS reciprocal sum error is about 5-10 times RSUM_TOL. Default is 5 x 10^{-5}.

`mlimit(1,2,3)` This allows the user to explicitly set the number of reciprocal vectors used in a regular Ewald run. Note that the sum goes from -MLIMIT(2) to MLIMIT(2) and -MLIMIT(3) to MLIMIT(3) with symmetry being used in first dimension. Note also the sum is truncated outside an automatically chosen sphere.

`ew_coeff` Ewald coefficient, in Å^{-1} . Default is determined by *dsum_tol* and *cutoff*. If it is explicitly inputed then that value is used, and *dsum_tol* is computed from *ew_coeff* and *cutoff*.

`nbflag` If *nbflag = 0*, construct the direct sum nonbonded list in the "old" way, *i.e*. update the list every *nsnb* steps. If *nbflag = 1* (the default when *imin = 0* or *ntb* > 0), *nsnb* is ignored, and the list is updated whenever any atom has moved more than 1/2 *skinnb* since the last list update.

`skinnb` Width of the nonbonded "skin". The direct sum nonbonded list is extended to *cut* + *skinnb*, and the van der Waals and direct electrostatic interactions are truncated at *cut*. Default is 2.0 Å. Use of this parameter is required for energy conservation, and recommended for all PME runs.

`skin_permit` (*pmemd.cuda* only) The threshold, as a fraction of *skinnb*, at which particle migration will trigger a non-bonded pair list rebuild. Enter values between 0.5 (minimum, default) and 1.0 (maximum). Once a particle has traveled more than half the non-bonded pair list margin *skinnb*, it is possible, although improbable, that another particle has also traveled this distance towards the first, and the pair is then within the non-bonded cutoff but not counted in the pair list. However, as the system gets larger, the probability that any one particle will travel 0.5 times the margin grows linearly, while the likelihood of a pair of nearby particles causing a violation remains constant and low. The frequency of pair list updates is a major factor in the moderate decrease in performance seen in very large systems (the scaling of the FFT is a smaller factor). Furthermore, if an interaction is missing, it will be at the periphery of the cutoff---the threshold at which non-bonded interactions are omitted by construction. Other codes have already implemented “sloppy pair lists,” so Amber is following suit and letting the user control the level of risk. By permitting particles to travel up to 0.75 times the pair list margin, pair list updates can be reduced by approximately half and miss one interaction in tens of millions. The most aggressive setting, 1.0, will see the pair list rebuilt at a third of the original rate and omit about one of every million valid interactions. A setting of 0.75 is recommended for the best tradeoff of performance to safety.

`nbtell` If *nbtell = 1*, a message is printed when any atom has moved far enough to trigger a list update. Use only for debugging or analysis. Default of 0 inhibits the message.

`netfrc` The basic "smooth" PME implementation used here does not necessarily conserve momentum. If *netfrc = 1*, (the default) the total force on the system is artificially removed at every step. This parameter is set to 0 if minimization is requested, which implies that the gradient is an accurate derivative of the energy. You should only change this parameter if you really know what you are doing.

`vdwmeth` Determines the method used for van der Waals interactions beyond those included in the direct sum. A value of 0 includes no correction; the default value of 1 uses a continuum model correction for energy and pressure.

`eedmeth` Determines how the switch function for the direct sum Coulomb interaction is evaluated. The default value of 1 uses a cubic spline. A value of 2 implies a linear table lookup. A value of three implies use of an "exact" subroutine call.

`eedtbdns` Density of spline or linear lookup table, if *eedmeth* is 1 or 2. Default is 500 points per unit.

`column_fft` 1 or 0 flag to turn on or off, respectively, column-mode fft for parallel runs. The default mode is slab mode which is efficient for low processor counts. The column method can be faster for larger processor counts since there can be more columns than slabs and the communications pattern is less congested. This flag has no effect on non-parallel runs. Users should test the efficiency of the method in comparison to the default method before performing long calculations. Default is 0 (off).

8.7.3. Using IPS for the calculation of nonbonded interactions

Isotropic Periodic Sum (IPS) is a method for long-range interaction calculation.[177–182] Unlike the Ewald method, which uses periodic boundary images to calculate long range interactions, IPS uses isotropic periodic images of a local region to calculate the long-range contributions.

The IPS method in the current version is different from that implemented in Amber10. All IPS potentials use rationalized polynomial forms and the electrostatic interaction is calculated using the polar IPS potential. [181] In addition, the 3D IPS/DFFT algorithm [180] is implemented to handle heterogeneous systems as well as finite systems. A homogeneous system is defined as the one where a cutoff region (with *cut* as its radius) has similar composition throughout the system, such as small molecular solutions. Otherwise, a system is defined as a heterogeneous system, such as interfacial systems or finite systems. For heterogeneous systems, a local region larger than the cutoff region, normally equal or larger than the periodic boundary box, must be used to produce accurate long range interactions. For homogeneous systems, it is recommended to use the 3D IPS method ($ips \leq 3$), which uses the cutoff distance, *cut*, to define the local region radius. *cut* is typically around 10 Å. The 3D IPS/DFFT method ($ips \geq 4$) can be used for any type of systems, but is recommended for heterogeneous systems only due to the extra discrete fast Fourier transform (DFFT) expense.

For the amoeba polarizable potentials in *sander*, 3D IPS is implemented for interactions between charges, dipoles, and multipoles. The local region radius takes the value of *ee_dsum_cut* in the amoeba namelist, typically, 7 Å.

`ips` Flag to control nonbonded interaction calculation method. The *cut* value will be used to define the local region radius for $ips \leq 3$. When IPS is used for electrostatic interaction, PME will be turned off. When using the amoeba polarizable potentials, *iamoeba=1*, *ips*>0 (same as *ips*=2)will turn on 3D IPS for all charge, dipole, and quadrupole interactions and the *ee_dsum_cut* value will be used to define the local region radius.

- **= 0** IPS will not be used (default).
- **= 1** 3D IPS will be used for both electrostatic and L-J interactions.
- **= 2** 3D IPS will be used only for electrostatic, including all multipole, interactions.
- **= 3** 3D IPS will be used only for L-J interactions.
- **= 4** 3D IPS/DFFT will be used for both electrostatic and L-J interactions.
- **= 5** 3D IPS/DFFT will be used only for electrostatic interactions.
- **= 6** 3D IPS/DFFT will be used only for L-J interactions.

`raips` Local region radius. *raips* is automatically set to *cut* for 3D IPS calculations ($ips \leq 3$) and should be set larger than *cut* for 3D IPS/DFFT calculations ($ips \geq 4$). A negative value indicates that it is set to the longest box side of a simulation system. For finite systems, i.e., system without periodic boundary conditions, $raips = \infty$, which corresponding no image interaction. The default value is -1 Å.

`mipsx,mipsy,mipsz` Number of grids along three periodic boundary sides when using 3D IPS/DFFT method ($ips \geq 4$). Negative values indicate they are calculated based on the grid size, *gridips*. Typical numbers are the lengths of box sides (in Å) divided by 2 Å. Default values are -1. When *ips*=6 and PME is used for electrostatic interaction, they are set to *nfft1*, *nfft2*, and *nfft3* defined for PME, respectively.

`mipso` The order of the B-spline interpolation (*ips*≥4). The higher the order, the better the accuracy (unless the charge grid is too coarse). The minimum order is 3. An order of 4 (the default) implies a cubic spline approximation which is a good standard value. The cost for the DFFT calculation goes as roughly the order to the third power. For *ips*=6 and PME is used to electrostatic interaction, it is set to *order* defined for PME.

`gridips` Grid size for 3D IPS/DFFT calculation (*ips*≥4). The default value is 2 Å.

`dvbips` Volume tolerance for updating IPS function grids (*ips*≥4). When volume changes like in *NPT* simulations, the grid size changes and IPS function on grid points need be updated. The updating only happens when the volume change ratio is more than *dvbips*. The default value is 1×10^{-8}.

8.7.4. Extra point options

Several parameters deal with "extra-points" (sometimes called lone-pairs), which are force centers that are not at atomic positions. These are currently defined as atoms with "EP" in their names. These input variables are really only for the convenience of force-field developers; *do not change the defaults unless you know what you are doing, and have read the code.* These variables are set in the &ewald namelist.

`frameon` If *frameon* is set to 1, (default) the bonds, angles and dihedral interactions involving the lone pairs/extra points are removed except for constraints added during parm. The lone pairs are kept in ideal geometry relative to local atoms, and resulting torques are transferred to these atoms. To treat extra points as regular atoms, set frameon=0.

`chngmask` If *chngmask*=1 (default), new 1-1, 1-2, 1-3 and 1-4 interactions are calculated. An extra point belonging to an atom has a 1-1 interaction with it, and participates in any 1-2, 1-3 or 1-4 interaction that atom has. For example, suppose (excusing the geometry) C1,C2,C3,C4 form a dihedral and each has 1 extra point attached as below

```
C1------C2------C3------C4
 |       |       |       |
Ep1     Ep2     Ep3     Ep4
```

The 1-4 interactions include C1-C4, Ep1-C4, C1-Ep4, and Ep1-Ep4. (To see a printout of all 1-1, 1-2, 1-3 and 1-4 interactions set verbose=1.) These interactions are masked out of nonbonds. Thus the amber mask list is rebuilt from these 1-1, 1-2, 1-3 and 1-4 pairs. A separate list of 1-4 nonbonds is then compiled. This list does not agree in general with the above 1-4, since a 1-4 could also be a 1-3 if its in a ring. See the *ephi()* routine for the precise algorithm involved here. The list of 1-4 nonbonds is printed if *verbose=1*.

8.7.5. Polarizable potentials

The following parameters are relevant for *polarizable potentials*, that is, when *ipol* is set to 1 in the &cntrl namelist. These variables are set in the &ewald namelist.

`indmeth` If indmeth is 0, 1, or 2 then the nonbond force is called iteratively until successive estimates of the induced dipoles agree to within DIPTOL (default 0.0001 debye) in the root mean square sense. The difference between indmeth = 0, 1, or 2 have to do with the level of extrapolation (1st, 2nd or 3rd-order) used from previous time steps for the initial guess for dipoles to begin the iterative loop. So far 2nd order (indmeth=1) seems to work best.

If indmeth = 3, use a Car-Parinello scheme wherein dipoles are assigned a fictitious mass and integrated each time step. This is much more efficient and is the current default. Note that this method is unstable for dt $>$ 1 fs.

`diptol` Convergence criterion for dipoles in the iterative methods. Default is 0.0001 Debye.

`maxiter` For iterative methods (indmeth<3), this is the maximum number of iterations allowed per time step. Default is 20.

`dipmass` The fictitious mass assigned to dipoles. Default value is 0.33, which works well for 1 fs time steps. If dipmass is set much below this, the dynamics are rapidly unstable. If set much above this the dynamics of the system are affected.

`diptau` This is used for temperature control of the dipoles (for indmeth=3). If *diptau* is greater than 10 (ps units) temperature control of dipoles is turned off. Experiments so far indicate that running the system in NVE with no temperature control on induced dipoles leads to a slow heating, barely noticeable on the 100ps time scale. For runs of length 10ps, the energy conservation with this method rivals that of SPME for standard fixed charge systems. For long runs, we recommend setting a weak temperature control (e.g. 9.99 ps) on dipoles as well as on the atoms. Note that to achieve good energy conservation with iterative method, the diptol must be below 10 -7 debye, which is much more expensive. Default is 11 ps (*i.e.* default is turned off).

`irstdip` If indmeth=3, a restart file for dipole positions and velocities is written along with the restart for atomic coordinates and velocities. If irstdip=1, the dipolar positions and velocities from the inpdip file are read in. If irstdip=0, an iterative method is used for step 1, after which Car-Parrinello is used.

`scaldip` To scale 1-4 charge-dipole and dipole-dipole interactions the same as 1-4 charge-charge (i.e. divided by scee) set scaldip=1 (default). If scaldip=0 the 1-4 charge-dipole and dipole-dipole interactions are treated the same as other dipolar interactions (i.e. divided by 1).

8.7.6. Dipole Printing

By including a &dipoles namelist containing a series of groups, at the end of the input file, the printing of permanent, induced and total dipoles is enabled.

The X, Y and Z components of the dipole (in debye) for each group will be written to mdout every NTPR steps. In order to avoid ambiguity with charged groups all of the dipoles for a given group are calculated with respect to the centre of mass of that group.

It should be noted that the permanent, inducible and total dipoles will be printed regardless of whether a *polarizable potential* is in use. However, only the permanent dipole will have any physical meaning when *non-polarizable potentials* are in use.

It should also be noted that the groups used in the dipole printing routines are not exclusive to these routines and so the dipole printing procedure can only be used when group input is *not* in use for something else (*i.e.* restraints).

8.7.7. Detailed MPI Timings

`profile_mpi` Adjusts whether detailed per thread timings should be written to a file called profile_mpi when running sander in parallel. By default only average timings are printed to the output file. This is done for performance reasons, especially when running *multisander* runs. However for development it is useful to know the individual timings for each mpi thread. When running in serial the value of profile_mpi is ignored.

= 0 No detailed MPI timings will be written (default).

= 1 A detailed breakdown of the timings for each MPI thread will be written to the file: profile_mpi.

8.8. Polariable Gaussian Multipole Model

8.8.1. Introduction

The polarizable Gaussian Multipole (pGM) force field is a polarizable Amber force field currently under development. Its valence terms, including bond, angle, and dihedral terms and the nonbonded van der Waals interactions are kept the same as the Amber additive force fields. The difference is only in its treatment of electrostatic interactions. Specifically, the pGM model represents atomic charge distributions as Gaussian-shaped multipole expansions at atomic centers. In the current version, electrostatic interactions are modeled with both permanent and induced multipoles, and both of which are truncated at the dipole level. Therefore the pGM model has a more sophisticated electrostatic framework than previous Amber polarizable force fields for which only induced dipoles are added to atomic charges. The theoretical framework in the use of pGM electrostatics in molecular dynamics is rigorously derived, including the interface of PME and pGM for liquid phase molecular simulations.[183]

The *sander* executables are capable of conducting pGM simulations if correct pGM prmtop files are provided. The input parameters and the namelist are listed below. A test case is also provided for a 512-water box for NVE simulations. Similar to other polarizable dipole models, the energy conservation can be achieved if the induction convergence criterion is set to 10^{-8}, though the induction iteration algorithm is still not optimized in the current release, so this is the bottomneck of using pGM electrostatics for molecular dynamics simulations. The NVT condition is also supported with the available thermal baths in *sander*. We are working to add the NPT condition, expected in the next release.

The PME parameters `ew_coeff`, `nfft1`, `nfft2`, `nfft3`, and `order` from the &`ewald` namelist are all related to the accuracy of the overall pGM/PME electrostatics in *sander*. Due to the high accuracy requirement of polarizable systems, `order` (i.e. the B-spline polynomial degree plus one) is recommended to be set to at least 6.[183] The `ew_coeff` together with the direct sum cutoff (`ee_dsum_cut`, see below) controls the accuracy in the Ewald direct sum, and `ew_coeff` together with the PME grid dimensions `nfft1/2/3` and `order` control the accuracy in the reciprocal sum. Since pGM model requires higher accuracy than classic point charge model, we recommend values exceed those in typical PME simulations with point-charge models. Typical values from our testing are `ew_coeff` = 0.35, `order` = 8, and `nfft1/2/3` are approximately equal to the cell length in the relevant direction, i.e. particle mesh grid spacing $\leq$ 1 Ångstrom.[183]

8.8.2. Input Variables

&cntrl Namelist input:

`ipgm` Set to 1 to use the pGM force field. When pGM is used, a &pol_gauss namelist is required (see below).

&pol_gauss Namelist input:

`pol_gauss_verbose` In addition to the usual sander output, by setting pgm_verbose=1, extra printing of energy and forces can be found in the output file. Default to 0.

`ee_dsum_cut` The ewald direct sum cutoff. It is recommend to be set to at least 9 Ångstrom.

`dipole_scf_tol` The induced dipoles in the pGM model are solutions to a set of linear equations. These equations are solved iteratively by the method of successive over-relaxation. `dipole_scf_tol` is the convergence criterion for the iterative solution to the linear equations. The iteration terminates when the RMS difference between induced dipoles of consecutive iteration steps is less than the tolerance. To achieve good energy conservation in NVE simulations (i.e. similar to that observed for additive force fields at otherwise identical conditions), a convergence criterion of 10^{-8} is needed.[183]

`scf_sor_coefficient` This is the successive over-relaxation parameter, which can be adjusted to reduce the number of iterations needed to achieve convergence. Recommend value is 0.7. Good values seem to be in the range 0.6 – 0.8.

scf_sor_niter The maximum iterations when solving the polarization equations. Recommended value is 20 for dipole_scf_tol of 10^{-6}. Add 10 iterations for every ten-fold increase in accuracy in induction convergence.

8.9. Varying conditions

This section of information is read (*if NMROPT > 0*) as a series of namelist specifications, with name "&wt". This namelist is read repeatedly until a namelist &wt statement is found with TYPE=END.

TYPE Defines quantity being varied; valid options are listed below.

ISTEP1,ISTEP2 This change is applied over steps/iterations ISTEP1 through ISTEP2. If ISTEP2 = 0, this change will remain in effect from step ISTEP1 to the end of the run at a value of VALUE1 (VALUE2 is ignored in this case). *(default= both 0)*

VALUE1,VALUE2 Values of the change corresponding to ISTEP1 and ISTEP2, respectively. If ISTEP2=0, the change is fixed at VALUE1 for the remainder of the run, once step ISTEP1 is reached.

IINC If IINC > 0, then the change is applied as a step function, with IINC steps/iterations between each change in the target VALUE (ignored if ISTEP2=0). If IINC =0, the change is done continuously. *(default=0)*

IMULT If IMULT=0, then the change will be linearly interpolated from VALUE1 to VALUE2 as the step number increases from ISTEP1 to ISTEP2. *(default)* If IMULT=1, then the change will be effected by a series of multiplicative scalings, using a single factor, R, for all scalings. i.e.

VALUE2 = (R**INCREMENTS) * VALUE1.

INCREMENTS is the number of times the target value changes, which is determined by ISTEP1, ISTEP2, and IINC.

The remainder of this section describes the options for the TYPE parameter. For a few types of cards, the meanings of the other variables differ from that described above; such differences are noted below. Valid Options for TYPE (you must use uppercase) are:

BOND Varies the relative weighting of bond energy terms.

ANGLE Varies the relative weighting of valence angle energy terms.

TORSION Varies the relative weighting of torsion (and J-coupling) energy terms. Note that any restraints defined in the input to the PARM program are included in the above. Improper torsions are handled separately (IMPROP).

IMPROP Varies the relative weighting of the "improper" torsional terms. These are not included in TORSION.

VDW Varies the relative weighting of van der Waals energy terms. This is equivalent to changing the well depth (epsilon) by the given factor.

HB Varies the relative weighting of hydrogen-bonding energy terms.

ELEC Varies the relative weighting of electrostatic energy terms.

NB Varies the relative weights of the non-bonded (VDW, HB, and ELEC) terms.

ATTRACT Varies the relative weights of the attractive parts of the van der waals and h-bond terms.

REPULSE Varies the relative weights of the repulsive parts of the van der waals and h-bond terms.

RSTAR — Varies the effective van der Waals radii for the van der Waals (VDW) interactions by the given factor. Note that this is done by changing the relative attractive and repulsive coefficients, so ATTRACT/REPULSE should not be used over the same step range as RSTAR.

INTERN — Varies the relative weights of the BOND, ANGLE and TORSION terms. "Improper" torsions (IMPROP) must be varied separately.

ALL — Varies the relative weights of all the energy terms above (BOND, ANGLE, TORSION, VDW, HB, and ELEC; does not affect RSTAR or IMPROP).

REST — Varies the relative weights of *all* the NMR restraint energy terms.

RESTS — Varies the weights of the "short-range" NMR restraints. Short- range restraints are defined by the SHORT instruction (see below).

RESTL — Varies the weights of any NMR restraints which are not defined as "short range" by the SHORT instruction (see below). When no SHORT instruction is given, RESTL is equivalent to REST.

NOESY — Varies the overall weight for NOESY volume restraints. Note that this value multiplies the individual weights read into the "awt" array. (Only if NMROPT=2; see Section 4 below).

SHIFTS — Varies the overall weight for chemical shift restraints. Note that this value multiplies the individual weights read into the "wt" array. (Only if NMROPT=2; see section 4 below).

SHORT — Defines the short-range restraints. For this instruction, ISTEP1, ISTEP2, VALUE1, and VALUE2 have different meanings. A short-range restraint can be defined in two ways.

(1) If the residues containing each pair of bonded atoms comprising the restraint are close enough in the primary sequence:

$$\text{ISTEP1} \leq \text{ABS(delta\_residue)} \leq \text{ISTEP2},$$

where delta_residue is the difference in the numbers of the residues containing the pair of bonded atoms.

(2) If the distances between each pair of bonded atoms in the restraint fall within a prescribed range:

$$\text{VALUE1} \leq \text{distance} \leq \text{VALUE2}.$$

Only one SHORT command can be issued, and the values of ISTEP1, ISTEP2, VALUE1, and VALUE2 remain fixed throughout the run. However, if IINC>0, then the short-range interaction list will be re-evaluated every IINC steps.

TGTRMSD — Varies the RMSD target value for targeted MD.

TEMP0 — Varies the target temperature TEMP0.

TEMP0LES — Varies the LES target temperature TEMP0LES.

TAUTP — Varies the coupling parameter, TAUTP, used in temperature scaling when temperature coupling options NTT=1 is used.

CUT — Varies the non-bonded cutoff distance.

NSTEP0 If present, this instruction will reset the initial value of the step counter (against which ISTEP1/ISTEP2 and NSTEP1/NSTEP2 are compared) to the value ISTEP1. This only affects the way in which NMR weight restraints are calculated. It does not affect the value of NSTEP that is printed as part of the dynamics output. An NSTEP0 instruction only has an effect at the beginning of a run. For this card (only) ISTEP2, VALUE1, VALUE2 and IINC are ignored. If this card is omitted, NSTEP0 = 0. This card can be useful for simulation restarts, where NSTEP0 is set to the final step on the previous run.

STPMLT If present, the NMR step counter will be changed in increments of STPMLT for each actual dynamics step. For this card, only VALUE1 is read. ISTEP1, ISTEP2, VALUE2, IINC, and IMULT are ignored. Default = 1.0.

DISAVE, ANGAVE, TORAVE If present, then by default time-averaged values (rather than instantaneous values) for the appropriate set of restraints will be used. DISAVE controls distance data, ANGAVE controls angle data, TORAVE controls torsion data. See below for the functional form used in generating time-averaged data.

For these cards: VALUE1 = τ (characteristic time for exponential decay) VALUE2 = POWER (power used in averaging; the nearest integer of value2 is used) Note that the range (ISTEP1→ISTEP2) applies only to TAU; The value of POWER is not changed by subsequent cards with the same ITYPE field, and time-averaging will always be turned on for the entire run if one of these cards appears.

Note also that, due to the way that the time averaged internals are calculated, changing τ at any time after the start of the run will only affect the relative weighting of steps occurring after the change in τ . Separate values for τ and POWER are used for bond, angle, and torsion averaging.

The default value of τ (if it is 0.0 here) is 1.0D+6, which results in no exponential decay weighting. Any value of $\tau \geq$ 1.D+6 will result in no exponential decay.

If DISAVE,ANGAVE, or TORAVE is chosen, one can still force use of an instantaneous value for specific restraints of the particular type (bond, angle, or torsion) by setting the IFNTYP field to "1" when the restraint is defined (IFNTYP is defined in the DISANG file).

If time-averaging for a particular class of restraints is being performed, all restraints of that class that are being averaged (that is, all restraints of that class except those for which IFNTYP=1) *must* have the same values of NSTEP1 and NSTEP2 (NSTEP1 and NSTEP2 are defined below). (For these cards, IINC and IMULT are ignored) See the discussion of time-averaged restraints following the input descriptions.

DISAVI, ANGAVI, TORAVI **ISTEP1:** Ignored.

ISTEP2: Sets IDMPAV. If IDMPAV $>$ 0, *and* a dump file has been specified (DUMPAVE is set in the file redirection section below), then the time-averaged values of the restraints will be written every IDMPAV steps. Only one value of IDMPAV can be set (corresponding to the first DISAVI/ANGAVI/TORAVI card with ISTEP2 $>$ 0), and *all* restraints (even those with IFNTYP=1) will be "dumped" to this file every IDMPAV steps. The values reported reflect the current value of τ.

VALUE1: The integral which gives the time-averaged values is undefined for the first step. By default, for each time-averaged internal, the integral is assigned the current value of the internal on the first step. If VALUE1$\neq$0, this initial value of internal r is reset as follows:

```
-1000. < VALUE1 < 1000.: Initial value = r_initial + VALUE
 VALUE1 <= -1000.: Initial value = r_target + 1000.
 1000. <= VALUE1 : Initial value = r_target - 1000.
```

r_target is the target value of the internal, given by R2+R3 (or just R3, if R2 is 0). VALUE1 is in angstroms for bonds, in degrees for angles.

VALUE2: This field can be used to set the value of τ used in calculating the time-averaged values of the internal restraints reported at the end of a simulation (if LISTOUT is specified in the redirection section below). By default, no exponential decay weighting is used in calculating the final reported values, regardless of what value of τ was used during the simulation. If VALUE2>0, then τ = VALUE2 will be used in calculating these final reported averages. Note that the value of VALUE2 = τ specified here only affects the reported averaged values in at the end of a simulation. It does not affect the time-averaged values used during the simulation (those are changed by the VALUE1 field of DISAVE, ANGAVE and TORAVE instructions).

IINC: If IINC = 0, then forces for the class of time-averaged restraints will be calculated exactly as (dE/dr_ave) (dr_ave/dx). If IINC = 1, then then forces for the class of time-averaged restraints will be calculated as (dE/dr_ave) (dr(t)/dx). Note that this latter method results in a non-conservative force, and does not integrate to a standard form. But this latter formulation helps avoid the large forces due to the $(1+i)$ term in the exact derivative calculation–and may avert instabilities in the molecular dynamics trajectory for some systems. See the discussion of time-averaged restraints following the input description. Note that the DISAVI, ANGAVI, and TORAVI instructions will have no affect unless the corresponding time average request card (DISAVE, ANGAVE or TORAVE, respectively) is also present.

`DUMPFREQ` Istep1 is the only parameter read, and it sets the frequency at which the coordinates in the distance or angle restraints are dumped to the file specified by the DUMPAVE command in the I/O redirection section. (For these cards, ISTEP1 and IMULT are ignored).

`END` END of this section.

NOTES:

1. All weights are relative to a default of 1.0 in the standard force field.
2. Weights are not cumulative.
3. For any range where the weight of a term is not modified by the above, the weight reverts to 1.0. For any range where TEMP0, SOFTR or CUTOFF is not specified, the value of the relevant constant is set to that specified in the input file.
4. If a weight is set to 0.0, it is set internally to 1.0D-7. This can be overridden by setting the weight to a negative number. In this case, a weight of exactly 0.0 will be used. *However,* if any weight is set to exactly 0.0, it cannot be changed again during this run of the program.
5. If two (or more) cards change a particular weight over the same range, the weight given on the last applicable card will be the one used.
6. Once any weight change for which NSTEP2=0 becomes active (i.e. one which will be effective for the remainder of the run), the weight of this term cannot be further modified by other instructions.
7. Changes to RSTAR result in exponential weighting changes to the attractive and repulsive terms (proportional to the scale factor**6 and **12, respectively). For this reason, scaling RSTAR to a very small value (e.g. $\leq$0.1) may result in a zeroing-out of the vdw term.

8.10. File redirection commands

Input/output redirection information can be read as described here. Redirection cards must follow the end of the weight change information. Redirection card input is terminated by the first non-blank line which does not start with a recognized redirection TYPE (e.g. LISTIN, LISTOUT, etc.).

The format of the redirection cards is

TYPE = filename

where TYPE is any valid redirection keyword (see below), and filename is any character string. The equals sign ("=") is required, and TYPE must be given in *uppercase* letters.

Valid redirection keywords are:

`LISTIN` An output listing of the restraints which have been read, and their deviations from the target distances *before* the simulation has been run. By default, this listing is not printed. cedureIf POUT is used for the filename, these deviations will be printed in the normal output file.

`LISTOUT` An output listing of the restraints which have been read, and their deviations from the target distances *_after* the simulation has finished. By default, this listing is not printed. If POUT is used for the filename, these deviations will be printed in the normal output file.

`DISANG` The file from which the distance and angle restraint information described below (Section 12.1) will be read.

`NOESY` File from which NOESY volume information (Section 12.2) will be read.

`SHIFTS` File from which chemical shift information (Section 12.3) will be read.

`PCSHIFT` File from which paramagnetic shift information (Section 12.3) will be read.

`DIPOLE` File from which residual dipolar couplings (Section 12.5) will be read.

`CSA` File from which CSA or pseduo-CSA restraints (Section 12.6) will be read.

`DUMPAVE` File to which the time-averaged values of all restraints will be written. If DISAVI / ANGAVI / TORAVI has been used to set IDMPAV≠0, then averaged values will be output. If the DUMPFREQ command has been used, the instantaneous values will be output.

8.11. Getting debugging information

The debug options in *sander* are there principally to help developers test new options or to test results between two machines or versions of code, but can also be useful to users who want to test the effect of parameters on the accuracy of their ewald or pme calculations. If the debug options are set, *sander* will exit after performing the debug tasks set by the user.

To access the debug options, include a &debugf namelist. Input parameters are:

`do_debugf` Flag to perform this module. Possible values are zero or one. Default is zero. Set to one to turn on debug options.

One set of options is to test that the atomic forces agree with numerical differentiation of energy.

`atomn` Array of atom numbers to test atomic forces on. Up to 25 atom numbers can be specified, separated by commas.

`nranatm` number of random atoms to test atomic forces on. Atom numbers are generated via a random number generator.

`ranseed` seed of random number generator used in generating atom numbers default is 71277

`neglgdel` negative log of delta used in numerical differentiating; e.g. 4 means delta is 10^{-4} Angstroms. Default is 5. *Note*: In general it does no good to set nelgdel larger than about 6. This is because the relative force error is at best the square root of the numerical error in the energy, which ranges from 10^{-15} up to 10^{-12} for energies involving a large number of terms.

`chkvir` Flag to test the atomic and molecular virials numerically. Default is zero. Set to one to test virials.

`dumpfrc` Flag to dump energies, forces and virials, as well as components of forces (bond, angle forces etc.) to the file "forcedump.dat" This produces an ascii file. Default is zero. Set to one to dump forces.

`rmsfrc` Flag to compare energies forces and virials as well as components of forces (bond, angle forces etc.) to those in the file "forcedump.dat". Default is zero. Set to one to compare forces.

Several other options are also possible to modify the calculated forces.

`zerochg` Flag to zero all charges before calculating forces. Default zero. Set to one to remove charges.

`zerovdw` Flag to remove all van der Waals interactions before calculating forces. Default zero. Set to one to remove van der Waals.

`zerodip` Flag to remove all atomic dipoles before calculating forces. Only relevant when polarizability is invoked.

`do_dir, do_rec, do_adj, do_self, do_bond, do_cbond, do_angle, do_ephi, do_xconst, do_cap` These are flags which turn on or off the subroutines they refer to. The defaults are one. Set to zero to prevent a subroutine from running. For example, set do_dir=0 to turn off the direct sum interactions (van der Waals as well as electrostatic). These options, as well as the zerochg, zerovdw, zerodip flags, can be used to fine tune a test of forces, accuracy, etc.

EXAMPLES:

This input list tests the reciprocal sum forces on atom 14 numerically, using a delta of 10^{-4}.

```
&debugf
neglgdel=4, nranatm = 0, atomn = 14,
do_debugf = 1,do_dir = 0,do_adj = 0,do_rec = 1, do_self = 0,
do_bond = 1,do_angle = 0,do_ephi = 0, zerovdw = 0, zerochg = 0,
chkvir = 0,
dumpfrc = 0,
rmsfrc = 0,
/
```

This input list causes a dump of force components to "forcedump.dat". The bond, angle and dihedral forces are not calculated, and van der Waals interactions are removed, so the total force is the Ewald electrostatic force, and the only nonzero force components calculated are electrostatic.

```
&debugf
neglgdel=4, nranatm = 0, atomn = 0,
do_debugf = 1,do_dir = 1,do_adj = 1,do_rec = 1, do_self = 1,
do_bond = 0,do_angle = 0,do_ephi = 0, zerovdw = 1, zerochg = 0,
chkvir = 0,
dumpfrc = 1,
rmsfrc = 0,
/
```

In this case the same force components as above are calculated, and compared to those in "forcedump.dat". Typically this is used to get an RMS force error for the Ewald method in use. To do this, when doing the force dump use ewald or pme parameters to get high accuracy, and then normal parameters for the force compare:

```
&debugf
neglgdel=4, nranatm = 0, atomn = 0,
do_debugf = 1,do_dir = 1,do_adj = 1,do_rec = 1, do_self = 1,
do_bond = 0,do_angle = 0,do_ephi = 0, zerovdw = 1, zerochg = 0,
chkvir = 0,
dumpfrc = 0,
rmsfrc = 1,
/
```

For example, if you have a 40x40x40 unit cell and want to see the error for default pme options (cubic spline, 40x40x40 grid), run 2 jobs—— (assume box params on last line of inpcrd file)

Sample input for 1st job:

```
&cntrl
dielc =1.0,
cut = 11.0, nsnb = 5, ibelly = 0,
ntx = 5, irest = 1,
ntf = 2, ntc = 2, tol = 0.0000005,
ntb = 1, ntp = 0, temp0 = 300.0, tautp = 1.0,
nstlim = 1, dt = 0.002, maxcyc = 5, imin = 0, ntmin = 2,
ntpr = 1, ntwx = 0, ntt = 0, ntr = 0,
jfastw = 0, nmrmax=0, ntave = 25,
/
&debugf
do_debugf = 1,do_dir = 1,do_adj = 1,do_rec = 1, do_self = 1,
do_bond = 0,do_angle = 0,do_ephi = 0, zerovdw = 1, zerochg = 0,
chkvir = 0,
dumpfrc = 1,
rmsfrc = 0,
/
&ewald
nfft1=60,nfft2=60,nfft3=60,order=6, ew_coeff=0.35,
/
```

Sample input for 2nd job:

```
&cntrl
dielc =1.0,
cut = 8.0, nsnb = 5, ibelly = 0,
ntx = 5, irest = 1,
ntf = 2, ntc = 2, tol = 0.0000005,
ntb = 1, ntp = 0, temp0 = 300.0, tautp = 1.0,
nstlim = 1, dt = 0.002, maxcyc = 5, imin = 0, ntmin = 2,
ntpr = 1, ntwx = 0, ntt = 0, ntr = 0,
jfastw = 0, nmrmax=0, ntave = 25,
/
&debugf
do_debugf = 1,do_dir = 1,do_adj = 1,do_rec = 1, do_self = 1,
do_bond = 0,do_angle = 0,do_ephi = 0, zerovdw = 1, zerochg = 0,
chkvir = 0,
dumpfrc = 0,
rmsfrc = 1,
/
&ewald
ew_coeff=0.35,
/
```

Note that an Ewald coefficient of 0.35 is close to the default error for an 8 Angstrom cutoff. However, the first job used an 11 Angstrom cutoff. The direct sum forces calculated in the 2nd job are compared to these, giving the RMS error due to an 8 Angstrom cutoff, with this value of ew_coeff. The reciprocal sum error calculated in the 2nd job is with respect to the pme reciprocal forces in the 1st job considered as "exact".

Note further that if in these two jobs you had not specified "ew_coeff" *sander* would have calculated ew_coeff according to the cutoff and the direct sum tolerance, defaulted to 10^{-5}. This would give two different ewald coefficients. Under these circumstances the direct, reciprocal and adjust energies and forces would not agree well

between the two jobs. However the total energy and forces should agree reasonrespanably, (forces to within about $5x10^{-4}$ relative RMS force error) Since the totals are invariant to the coefficient.

Finally, note that if other force components are calculated, such as van der Waals, bond, angle, etc., then the total force will include these, and the relative RMS force errors will be with respect to this total force in the denominator.

8.12. multisander

The *multisander* functionality is available in the parallel versions of the programs (i.e., *sander.MPI*). This mode allows multiple independent simulations, or replicas, to be run in the same program instance. It is particularly useful for computer clusters in which priority is given to large CPU-count jobs. In this case, the command-line usage of sander and pmemd is slightly altered, as shown below:

```
mpirun -np <#proc> sander.MPI -ng <#groups> -groupfile groupfile
```

In this case, `#proc` processors will be evenly divided among `#groups` individual simulations (`#proc` must be a multiple of `#group`!). The groupfile consists of a number of lines which is the command-line for each of the `#groups` simulations you wish to run. Comment lines (i.e., those with # in the first column) are ignored, after which the first `#groups` lines are read as the command-line flags of the N^{th} simulation.

The multisander and multipmemd mechanisms are also utilized for methods requiring multiple simulations to communicate with one another, such as thermodynamic integration in sander and replica exchange molecular dynamics (both described later). An example groupfile and program call are shown below.

Groupfile:

```
# Comment lines must start with a pound sign
# and there can be as many comment lines as you
# want, wherever you want them.
-O -p prmtop1 -c inpcrd1 -i replica1.mdin -suffix replica1
-O -p prmtop2 -c inpcrd2 -i replica2.mdin -suffix replica2
-O -p prmtop3 -c inpcrd3 -i replica3.mdin -suffix replica3
-O -p prmtop4 -c inpcrd4 -i replica4.mdin -suffix replica4
```

The `-suffix` flag behaves slightly differently than it does for classical use. In standard simulations (*i.e.*, without *multisander* or *multipmemd*), the provided suffix will be applied only to output files that are printed but were not given names on the command-line. With multisander, however, each thread has to produce different output files so that different replicas do not try to write to the same file. As a result, a default suffix of 000, 001, 002, etc. is given to the replicas and is added to every unspecified output file. If a `-suffix` is specified in the groupfile, as shown above, every output file—including those given an explicit name for that replica—are given the additional suffix.

The four simulations shown in the groupfile above can be run on 8 processors each with the following command (note, running *sander.MPI* may differ on your system).

```
mpirun -np 32 sander.MPI -ng 4 -groupfile groupfile
```

The *multisander* and *multipmemd* concepts are implemented via the use of MPI communicators. Each replica is assigned a replica-wide communicator along which all communications required for standard MD simulations are performed (called `commsander` and `pmemd_comm` in *sander* and *pmemd*, respectively). Each replica communicator has a master thread (rank 0 in that communicator), and the master thread of each replica are joined in another MPI communicator of replica masters (called `commmaster` and `pmemd_master_comm` in *sander* and *pmemd*, respectively). All inter-replica communication is performed via `commmaster` or `pmemd_master_comm`.

By default, all N threads are allocated to each of the M groups by dividing the threads sequentially. That is, the first N/M threads are assigned to replica 0, the second group of N/M threads are assigned to replica 1, etc. The `-ng-nonsequential` flag will stripe the thread assignments. Replica 0 will receive threads 0, $N-1$, $2N-1$, etc., while replica 1 receives threads 1, N, $2N$, etc.

8.13. Programmer's Corner: The sander API

By Jason M. Swails

This section describes a new feature of *sander*—an application programmer interface (API) that encapsulates some of *sander*'s basic functionality into a library that can be included in your own programs. *sander* was originally written in Fortran as a standalone program that made extensive use of common blocks (i.e., global variables) and uses MPI for the parallel implementation rather than a type of shared-memory parallelization scheme like OpenMP or pthreads. This design conferred a number of constraints on the resulting API.

1. Only one system can be set up for use with the API at a time. Switching Hamiltonians or input parameters requires a lot of deallocation and reallocation and will be inefficient if done very frequently.
2. Only serial execution is supported.
3. LES and non-LES functionality cannot be combined in the same library.
4. File names have a fixed maximum length (256 characters). This can be extended only by adjusting the *sander* source code and recompiling.

Despite these limitations, the *sander* API provides functionality unavailable in other libraries, including QM/MM forces and energies, PB and GB energies, and LES functionality (through a separate library). Although originally written in Fortran, an API has been provided for four languages: Fortran, C, C++, and Python.

The next sections describe the general API design and then the Fortran, C, C++, and Python APIs specifically. **Note:** the python version of this API is sometimes referred to as *pysander*. However, there is no program called *pysander*; that term is rather a shorthand for using "import sander" within a python driver script.

8.13.1. General API Design

This section describes the functions that are available in each variant of the *sander* API. The exact syntax for how the various functions and subroutines are called—and what, if anything, they return—is listed in the following sections for each API.

8.13.1.1. Data Structures

The following data structures are provided as part of the API. These data structures provide a way to provide input or query output from the API. They are the equivalent of C `structs` (Fortran `type` and Python `class`). All floating point data types are double precision (`double` in C and C++, `double precision` in Fortran, and `float` in Python). All integer data types are standard integers (`int` in C and C++, `integer` in Fortran, and `int` in Python).

sander_input

This contains variables used to provide input for the sander API. The attributes that are exposed here have the same name, options, and function as the input options with the same name described earlier in this chapter (and some in earlier chapters). These attributes are listed below, with their data type (float or integer) listed in parentheses at the end.

extdiel External dielectric constant for GB calculations. (float)

intdiel Internal dielectric constant for GB calculations. (float)

rgbmax Distance cutoff in Angstroms to use when computing effective GB radii. (float)

saltcon Salt concentration, in Molarity, to use when modeling ionic strength effects in a GB calculation. (float)

cut Nonbonded cutoff in Angstroms. (float)

dielc Dielectric constant to use for all electrostatic interactions. You should use extdiel or intdiel, described above, for GB calculations (this option should only be used if you are sure it is what you want—it is usually not what you want). (float)

rdt This is an option specific to GB calculations with LES (and only has an effect when using the sanderles library). When using GB with LES, non-LES atoms require multiple effective radii due to alternate descreening effects from the different copies. When the multiple radii differ by less than rdt, only a single radius will be used for this atom. Default is 0.0. See Chapter 14 for more information. (float)

igb GB model to use for GB calculations. Allowable values are 0 (no GB), 1, 2, 5, 6 (vacuum), 7, 8, and 10 (PB). More information is available on page 11. (integer)

alpb If 1, use the analytical linearized Poisson-Boltzmann approximation. See Section **??** for more information. (integer)

gbsa If set to 0, no SASA-based nonpolar free energy of solvation correction is used. If set to 1, the SASA is approximated using the linear combination of pairwise overlaps method (LCPO). If set to 2, the SASA is approximated using a recursive algorithm constructing spheres around each atom. Note, gradients (forces) are not available from this model, so the forces returned by the API will be incorrect if this option is used. (integer)

lj1264 If 1, use the 12-6-4 Lennard Jones potential form designed for divalent metal ions. If 0, do not use the 12-6-4 Lennard-Jones model. The topology file must be set up correctly to use the 12-6-4 model first! (integer)

ipb Option to compute the solvation free energy using the Poisson-Boltzmann equation. Allowable values are 0 (no PB equation), 1, 2, and 4. See Chapter **??** for more information. (integer)

vdwmeth For periodic simulations only (i.e., when *ntb*, below, is set to 1). When set to 1, a long-range dispersion correction based on an analytical integral assuming an isotropic, uniform bulk particle distribution beyond the cutoff is added to the van der Waals energy. When set to 0, no correction is used. (integer)

ew_type For periodic simulations only (i.e., when *ntb*, below, is set to 1). When set to 0, the particle-mesh Ewald method is used to compute long-range electrostatics. When set to 1, a traditional Ewald method is used to compute long-range electrostatics (PME is *much* faster for systems with more than 500 atoms or so). (integer)

ntb If set to 0, periodic boundaries are not applied. If set to 1, periodic boundaries are used. The value of 2 does not (yet) apply to the API (i.e., constant pressure), as this is an MD-specific option. (integer)

ifqnt If set to 1, a QM/MM potential is used (and you must provide a set of valid QM options as well). If set to 0, no QM/MM potential is used. (integer)

jfastw Fast water definition flag. By default, the system is searched for water residues, and special routines are used to SHAKE these systems (i.e., they are constrained using the analytical SETTLE algorithm). If set to 0, this default behavior is triggered. If set to 4, the numerical SHAKE routines are used. (integer)

ntf Flag to determine which, if any, interactions to omit from the energy calculation. (integer)

ntc Flag to determine whether to use the SHAKE algorithm to constrain bond distances. (integer)

There are two subroutines that will initialize a sander_input instance with default values—one that prepares the input for a periodic simulation and one that prepares the input for an aperiodic system (either gas phase or GB calculation). The default values assigned are summarized in table 8.1.

Table 8.1.: *Summary of default values assigned to* `sander_input` *variables by the two initialization subroutines provided in the* sander *API. When alternate values are given for* `gas_sander_input`*, the latter corresponds to the value assigned if a GB model is requested.*

`sander_input` variable	`gas_sander_input` default	`pme_sander_input` default
extdiel	1 or 78.5	0
intdiel	1	0
rgbmax	25.0	25.0
saltcon	0	0
cut	1000.0	8.0
dielc	1	1
rdt	0	0
igb	6 or input value	0
alpb	0	0
gbsa	0	0
lj1264	0	0
ipb	0	0
inp	0	0
vdwmeth	0	1
ew_type	0	0
ntb	0	1
ifqnt	0	0
jfastw	0	0
ntc	1	1
ntf	1	1

qmmm_input_options

This struct contains a set of options for controlling what portion of the system is treated using quantum mechanics (QM), which QM Hamiltonian is used to treat the QM portion of the system, how the boundary between the QM and MM portions of the system are handled, and how the QM and MM portions interact.

The variables in this data structure have the same name and function as the variables defined in the `&qmmm` namelist of the input file. You can find more information about QM/MM in Chatper 6 on page 45 and about the options specifically in Chapter 4 and Subsection 6.1.6.

There are three types of data types in this struct. Floating point, integer, and character array (i.e., string) values. Like with sander_input above, floating point numbers are represented in full double precision and integers as standard integers. The strings in this section are fixed-size arrays of characters. The type of each variable is indicated in parentheses after the variable is defined followed by the fixed-size length of the array if it is an array value. The standard value for the maximum number of QM atoms (`MAX_QUANTUM_ATOMS`) is 10,000.

Note that strings are treated by the API as Fortran strings, *not* C-style strings. The main difference is that Fortran strings do not have a null terminal character (`'\0'`), which means that every character after the final "letter" of the string must contain a space (or null character). As a result, the typical string routines defined in the C `string.h` header file (e.g., `strcpy` and `strncpy`) may not assign the strings correctly if they are not properly initialized entirely with spaces first. That is why the `qm_sander_input` function is provided as part of the API, so I suggest that you always initialize a `qmmm_input_options` data structure using this method when using the C or C++ APIs.

The defaults listed below are those assigned by the `qm_sander_input` function in the API (and are the same as the defaults defined in Subsection 6.1.6).

qmcut Nonbonded cutoff in Angstroms used for QM/MM nonbonded interactions (note there is no such thing as a cutoff *within* the QM region, since it is the wavefunction of the entire system we are optimizing). The default value is the MM cutoff being used (i.e., `cut` from `sander_input`, above). (float)

lnk_dis Distance in Angstroms of the QM atom to its link atom. Default is 1.09. (float)

scfconv Controls the convergence of the SCF calculation. The SCF terminates when the energy difference between the last two steps is smaller than the value given here. Default is 10^{-8} and the smallest value that can practically be used within the limits of double precision floating point arithmetic is 10^{-14}. (float)

errconv SCF tolerance on the maximum absolute value of the erro matrix (i.e., the commutator of the Fock matrix with the density matrix). The value is in units of Hartrees. The default value is large enough that scfconv will always be more strict. (float)

dftb_telec Electronic temperature, in K, used to accelerate SCC convergence in DFTB calculations. The electronic temperature affects the Fermi distribution promoting some HOMO/LUMO mixing, which can accelerate the convergence in difficult cases. In most cases, a low *telec* (around 100K) is enough. Should be used only when necessary, and the results checked carefully. Default: 0.0 (float)

dftb_telec_step The size of the step to take when reducing the electronic temperature in a DFTB calculation. The smaller the step, the longer it will take to get the electronic temperature to zero. (float)

fockp_d1 First prefactor for the Fock matrix prediction. Default is 2.4. Changing this is not recommended. (float)

fockp_d2 Second prefactor for the Fock matrix prediction. Default is -1.2. Changing this is not recommended. (float)

fockp_d3 Third prefactor for the Fock matrix prediction. Default is -0.8. Changing this is not recommended. (float)

fockp_d4 Fourth prefactor for the Fock matrix prediction. Default is 0.6. Changing this is not recommended. (float)

damp SCF damping factor. Default is 1.0. Changing this is not recommended. (float)

vshift Controls level shifting for NDDO methods (not DFTB). Virtual orbitals can be shifted up by vshift (in eV) to improve SCF convergence in cases with a small HOMO/LUMO gap. Default is 0.0. (float)

kappa Related to the Debye salt concentration for GB models. This is set automatically from saltcon in the `sander_input` data structure. (float)

pseudo_diag_criteria Controls whether a pseudo-diagonalization of the Fock matrix can be performed (not applicable for DFTB). Default is 0.05. (float)

min_heavy_mass The smallest value, in atomic mass units, that an atomic mass can have and still be considered a "heavy-atom" (i.e., anything besides Hydrogen). Default is 4.0. (float)

r_switch_hi If qmmm_switch (below) is turned on, this is the distance, in Angstroms, at which the switch goes to zero. By default, it is the same as qmcut. (float)

r_switch_lo If qmmm_switch (below) is turned on, this is the distance, in Angstroms, at which the switch turns on. By default, it is 2 Angstroms smaller than `r_switch_hi`. (float)

iqmatoms List of atom indexes, starting from 1, that will be treated using QM. This is one way, along with qmmask, of specifying the QM region. Default is an empty list. (integer array, `MAX_QUANTUM_ATOMS`).

qmgb Specifies how the QM region should be treated with Generalized Born. (integer)

- **= 2** (default) As described above, the electrostatic and "polarization" fields from the MM charges and the exterior dielectric, respectively, are included in the Fock matrix for the QM Hamiltonian.
- **= 3** This is intended for debugging and is only useful for single-point calculations. This computes the GB energy by treating every atom in the QM region as a point charge equal to its Mulliken charge. This can be compared to the result when qmgb is set to 2 to evaluate the "strain" energy from the GB solvation.

lnk_atomic_no The atomic number of the element you wish to use as the link atom. Default is 1 (Hydrogen). (integer)

ndiis_matrices The number of error vectors to use for the DIIS convergence algorithm. Default is 6. (integer)

ndiis_attempts The number of iterations that DIIS extrapolation will be attempted. Not available for DFTB. Default value is 0, maximum is 1000. (integer)

lnk_method The method used to define how classical valence terms across the QM/MM boundary will be treated. See Subsection 6.1.7 for more information. Default is 1. (integer)

qmcharge The net charge of the QM region. Default is 0. (integer)

corecharge The net charge of the core QM region. Default is 0. (integer)

buffercharge The net charge of the buffer QM region. Default is 0. (integer)

spin Spin multiplicity of the QM region. Default is 1 (singlet). (integer)

qmqmdx Controls whether QM-QM derivatives are computed analytically or pseudo-numerically. The default (and recommended) is to use analytical QM-QM derivatives. Set to 1 for analytical derivatives, 2 for pseudo-numerical derivatives. Default is 1. (integer)

verbosity This has no effect on the API, since output is suppressed. Keep the default value of 0. (integer)

printcharges This has no effect on the API since output is suppressed. Keep the default value of 0. (integer)

printdipole This has no effect on the API, since output is suppressed. Keep the default value of 0. (integer)

print_eigenvalues This has no effect on the API, since output is suppressed. Keep the default value of 0. (integer)

peptide_corr If set to 0, (default), do not apply a correction to peptide linkages. If set to 1, apply a MM correction to peptide linkforages. (integer)

itrmax Maximum number of SCF iterations to perform before deciding that the convergence has failed. Default is 1000. (integer)

printbondorders This has no effect on the API, since output is suppressed. Keep the default value of 0. (integer)

qmshake Controls whether SHAKE is applied to QM atoms. If 0, no SHAKE. If 1 (default), SHAKE QM atoms if MM SHAKE is turned on. By default, MM SHAKE is not turned on. This really has no effect, anyway, since the API does not currently support dynamics. (integer)

qmmmrij_incore If set to 1 (default), store QM-MM pairs and related equations in memory. If set to 0, do not. (integer)

qmqm_erep_incore If set to 1 (default), store QM-QM 1-electron repulsion integrals to memory. If set to 0, calculate them on-the-fly. (integer)

pseudo_diag If set to 1 (default), allow the use of pseudo-diagonalization of the Fock matrix as long as the `pseudo_diag_criteria` is met. (integer)

qm_ewald Specifies how the long-range electrostatics for the QM region should be treated. See the description in Subsection 6.1.6 for more information. (integer)

qm_pme If 0, use a regular Ewald sum for computing QM-QM and QM-MM long-range electrostatic interactions. If 1 (default), use PME instead. (integer)

kmaxqx Number of K-space vectors to use in the Ewald/PME calculations in the X-dimension. Default value is 8. (integer)

kmaxqy Same as above, but in the Y-dimension. (integer)

kmaxqz Same as above, but in the Z-dimension. (integer)

ksqmaxsq Specifies the maximum number of K^2 values for the spherical cutoff in reciprocal space when doing a QM-MM Ewald sum. The default value of 100 should be optimal for most systems. (integer)

qmmm_int Controls the way in which the QM-MM interaction is handled. See Subsection 6.1.6 for more information. Default is 1. (integer)

adjust_q Controls how charge is conserved during a QM/MM calculation with respect to link atoms. See Subsection 6.1.6 for more information. Default is 2. (integer)

tight_p_conv Controls the tightness of the convergence criteria on the density matrix in the SCF. If 0 (default), the convergence is loose. If set to 1, convergence is tight. See Chapter 4 for more information. (integer)

diag_routine The diagonalization routine to use to diagonalize the Fock matrix. By default (diag_routine = 0), the fastest routine is chosen. See the description in Chapter 4 for more details. (integer)

density_predict If 1, use the density matrix from the previous MD step. Since MD is not currently supported in the API, do not deviate from the default value of 0. (integer)

fock_predict If set to 0, do not attempt to predict the Fock matrix. (Default). If set to 1, try to. (integer)

vsolv If set to 1, use variable solvent QM/MM. If set to 0 (default), do not. This option is irrelevant to the API since it does not support QM/MM. (integer)

dftb_maxiter The maximum number of SCF iterations to be used in SCC-DFTB calculations. Default is 70. (integer)

dftb_disper If set to 1, use a dispersion correction for DFTB/SCC-DFTB. If set to 0 (default), do not. (integer)

dftb_chg Has no effect on the API, since printing is disabled. (integer)

abfqmmm Toggles the adaptive biased force QM/MM. Since the API does not support MD, this option has no effect. Default is 0. (integer)

hot_spot If set to 1, activates hot spot-like adaptive calculation in which the forces of atoms in the buffer region are linear combinations of the forces obtained from the extended and reduced calculations using a smoothing function. If set to 0 (default), disable this behavior. (integer)

qmmm_switch If set to 1, use a switching function defined by `r_switch_lo` and `r_switch_hi`. If set to 0 (default), do not. (integer)

core_iqmatoms A list of atom indices (starting at 1) that are selected for inclusion in the core QM/MM region in adaptive simulations. (integer array, `MAX_QUANTUM_ATOMS`)

buffer_iqmatoms A list of atom indices (starting at 1) that are selected for inclusion in the buffer QM/MM region in adaptive simulations. (integer array, `MAX_QUANTUM_ATOMS`)

qmmask An Amber selection mask that provides another way of defining the QM region instead of `iqmatoms`. (character array, 8192)

coremask An Amber selection mask that provides another way of defining the core QM region in adaptive simulations instead of `core_iqmatoms`. (character array, 8192)

buffermask An Amber selection mask that provides another way of defining the buffer QM region in adaptive simulations instead of `buffer_iqmatoms`. (character array, 8192)

centermask An Amber selection mask that defines the center region. If not set, it defaults to `coremask`. (character array, 8192)

dftb_3rd_order Specifies the 3rd-order DFTB correction. Default ('NONE') means no 3rd order correction is used. See Chapter 4 for more information. (character array, 256)

qm_theory String that defines which level of QM theory to use. There is no default and this must be supplied. Available options are defined in Chapter 4. (character array, 12)

pot_ene

This data structure is populated when the energy and forces are computed for the positions that are currently set. All elements of this data structure are double-precision floating point numbers and are given in kilocalories per mole.

tot The total potential energy

vdw The van der Waals contribution to the total energy (not including 1-4 interactions)

elec The electrostatic contribution to the total energy (not including 1-4 interactions)

gb Polar solvation free energy from GB calculations

bond The energy contribution from valence bonds.

angle The energy contribution from valence angles.

dihedral The energy contribution from valence torsions.

vdw_14 The energy contribution from 1-4 van der Waals interactions

elec_14 The energy contribution from 1-4 electrostatic interactions

constraint Really misnamed, this is the total restraint energy if NMR or positional restraints are used.

polar Polarization energy if you are using a polarizable force field.

hbond The 10-12 contribution to the total energy (not used in modern force fields)

surf The non-polar solvation free energy contribution from GB and PB calculations.

scf The QM energy contribution (includes charge-charge interactions between MM and QM atoms, but not dispersion interactions—those are added to the vdw component).

disp Dispersion energy contribution (?? not really sure what this is)

dvdl Not really applicable to the API, since it is used for constant pH MD calculations. This should always be 0.

angle_ub For CHARMM force field, this is the Urey-Bradley contribution to the total energy.

imp For CHARMM force field, this is the improper torsion contribution to the total energy.

cmap For CHARMM force field, this is the correction map energy contribution for coupled torsions.

emap When fitting to an electron density map, this is the restraint energy derived from violations to the map.

les The total energy contributed by the LES copies.

noe The energy penalty for NOE violations.

pb The total polar solvation free energy from PB calculations.

rism The total solvation free energy from 3D-RISM calculations.

ct Charge transfer energy (for crg_reloc)

amd_boost This is the AMD boosting energy. It is not applicable for the API since molecular dynamics is not currently supported.

8.13.1.2. Basic subroutines

This section describes the functions and subroutines that are defined by the API and explains what they do. Since their exact behavior (e.g., their arguments and return values) differ depending on which API you are using, the exact usage is deferred to later sections. However, what they *do* is described here.

There are very strong similarities between the C/C++ and Fortran function calls. While the Python function calls are also similar, the Python behavior often differs the most.

gas_sander_input This function will initialize a `sander_input` data structure with the appropriate defaults for carrying out either a gas-phase calculation or an implicit solvent GB calculation. It takes an integer argument defining the GB model to use. See the `igb` variable in the `sander_input` data structure above for allowable values.

It is recommended that you initialize your `sander_input` instance using either this routine or `pme_sander_input` to make sure that all variables are initialized. Uninitialized variables in any of the compiled languages (i.e., not Python) take on undefined behavior and could result in strange bugs.

This can be called regardless of whether or not a system is currently set up.

pme_sander_input This function will initialize a `sander_input` data structure with the appropriate defaults for carrying out a PME calculation on a periodic system. It is recommended that you initialize your `sander_input` instance using either this routine or `gas_sander_input` to make sure that all variables are initialized. Uninitialized variables in any of the compiled languages (i.e., not Python) take on undefined behavior and could result in strange bugs.

This can be called regardless of whether or not a system is currently set up.

qm_sander_input This function will initialize a `qmmm_input_options` data structure with the defaults listed in Subsection 8.13.1.1. This is the recommended method for initializing QM input options, particularly in the C and C++ interfaces where string handling is fragile.

sander_setup These functions take a topology file, coordinates, box dimensions, and a set of input options (`sander_input` and `qmmm_input_options`) and sets up the *sander* API so that energies and forces can be calculated.

These functions can only be called if no system is currently set up. You must call `sander_cleanup` before setting up a different system (or changing input parameters).

set_positions This function takes an array of double precision particle positions ($3 \times$ natom) and sets them as the active conformation.

This function can only be called if there is currently a system set up.

set_box This function takes three box lengths and the angles between them and sets the unit cell (and reciprocal unit cell) vectors from these values.

This function can only be called if there is currently a system set up.

sander_natom This function returns the number of atoms present in the system that is currently set up.

This function can only be called if there is currently a system set up.

get_positions This function returns the currently active atomic coordinates for the system that is currently set up.

This function can only be called if there is currently a system set up.

get_inpcrd_natom This function takes the name of an inpcrd file and reads the number of atoms that are defined in this file. If this file is not present or its format cannot be determined, the number of atoms is set to -1, which indicates an error.

This function can be called regardless of whether or not a system is currently set up.

read_inpcrd_file This function takes the name of an inpcrd file, an array of length $3 \times$ natom double precision floating point numbers, and an array of 6 double precision floating numbers and fills them with the atomic coordinates and box dimensions, respectively. The box dimensions are stored as a, b, c, α, β, and γ.

Since the two arrays must already be allocated, the typical workflow is to call `get_inpcrd_natom` to determine how large the coordinate array must be made. Then call `read_incprd_file` after allocating the coordinate array.

This function can be called regardless of whether or not a system is currently set up.

is_setup This function returns whether a system is currently set up or not.

This function can be called regardless of whether or not a system is currently set up.

energy_forces This function computes the energy and forces for the current coordinates of the system that is currently set up and returns them in the potential energy data structure and $(3 \times$ natom)-length double precision array that is passed to this routine.

This function can only be called if a system is currently set up.

sander_cleanup This function clears all of the internal memory initialized and allocated by the sander_setup routines. This function can only be called if a system is set up, but after this function completes, a system is no longer set up.

8.13.2. The Fortran API

The Fortran API is implemented with a Fortran module. The module is compiled when AmberTools is built and the modulefile is deposited in `$AMBERHOME/include`.

One of the limitations of Fortran modules is that you *must* use the same compiler to build your program as you used to compile AmberTools in the first place. If you wish to change compilers (in many cases, this also includes compiler versions as well), then you need to recompile AmberTools with that same compiler as well. The available API modules are `sander_api` for the standard *sander* functionality and `sanderles_api` for the LES capabilities.

8.13.2.1. Data structures

The Fortran data structures are all different sequence `types`. The sequence descriptor simply means that they are layed out sequentially in memory in exactly the same way that a struct is in C or C++. Variables within a type are accessed using the `%` operator.

The sander input options are available as `type(sander_input)`, the QM/MM input options are available as `type(qmmm_input_options)`, and the potential energy data structure is available as `type(potential_energy_rec)`. The names of the variables that make up each of these types are the same as those defined in Subsection 8.13.1.1.

An example of using the `sander_input` type is shown in the small code fragment below

```
use sander_api, only : sander_input
type(sander_input) :: inp
inp%cut = 9999.d0
inp%ifqnt = 0
inp%igb = 5
```

8.13.2.2. Function call syntax

This section details the function calls for the various subroutines available in the Fortran API. All of these subroutines and functions are public members of both `sanderapi_mod` and `sanderlesapi_mod`.

```
subroutine gas_sander_input(sander_input inp, int igb)
```

This subroutine takes a `sander_input` instance and optionally an integer corresponding to the GB model you want to use (see the description for igb above with regards to permissible values). If an illegal `igb` value is provided, a warning is printed to `stderr` and a value of 6 (corresponding to vacuum) is given to `inp%igb`. See table 8.1 for a list of the default values assigned to each variable.

```
subroutine pme_sander_input(sander_input inp)
```

This subroutine takes a `sander_input` instance and initializes every variable inside with the value listed in table 8.1.

```
subroutine qm_sander_input(qmmm_input_options inp)
```

This subroutine takes a `qmmm_input_options` instance and initializes all of the variables to the values given in Subsection 8.13.1.1. This is the recommended way to initialize the QM/MM options type.

```
subroutine sander_setup(character(len=*) top,
                        double precision, dimension(3*natom) crd,
                        double precision, dimension(6) box,
                        sander_input inp,
                        qmmm_input_options qm_inp,
                        integer ierr)
```

This subroutine sets up *sander* with the given topology file name, given coordinates, given box dimensions, input options, and QM/MM input options. The ierr variable is an error flag and will come back with a value of 0 if the setup succeeded or a value of 1 if it failed. Every other variable is input and guaranteed not to change.

No checking is done to make sure that the number of coordinates provided is correct compared to the number of atoms defined in the topology file. Note that answers will be ridiculous if the coordinate order does not match the atom order in the topology file. Segfaults and other memory violations are possible if the provided coordinate array or box array are too small.

The box array is given in the format $(a, b, c, \alpha, \beta, \gamma)$, which is the same as the format used at the bottom of the input coordinate and restart files. This argument is required even if the system is not periodic, but the values are not used (so they can be initialized to anything).

The `qmmm_input_options` variable is optional, but must be present if `inp%ifqnt` is 1. If `qm_inp` is not provided but QM/MM is requested, an error message will be printed to stderr and `ierr` will return with a value of 1.

The error flag `ierr` is required. If `qm_inp` is omitted, then ierr must be specified via keyword. See the examples at the end of this section. This function should never be called if a system is already set up.

```
subroutine set_positions(double precision, dimension(3*natom) crd)
```

This subroutine sets the current positions of the active system (and so can only be called if a system is currently set up). Note that the onus is on the programmer to make sure that the coordinate array is large enough. No error checking is done. The input parameter is guaranteed not to change.

```
subroutine set_box(double precision a, double precision b,
                   double precision c,
                   double precision alpha, double precision beta,
                   double precision gamma)
```

This subroutine sets the box dimensions and angles from the input parameters (which are guaranteed not to change).

```
subroutine get_positions(double precision, dimension(3*natom) positions)
```

This subroutine stores the currently active positions inside the passed array.

```
subroutine energy_forces(type(potential_energy_rec) ener,
                         double precision, dimension(3*natom) forces)
```

This subroutine will compute the energies and forces from the current conformation of the system that is currently set up and populate the `ener` type and `forces` array with the resulting values. Those parameters are purely output. The energies are all given in units of kilocalories per mole and forces are given in $kcal/mol/\AA$. This subroutine can only be called if a system is currently set up.

```
subroutine sander_cleanup()
```

This subroutine will deallocate all memory used by the *sander* API and return it to a state where no system is set up and a new one can be initialized.

```
logical function is_setup()
```

This function can be called to query whether there is currently a system set up for the *sander* API. It returns `.true.` if a system is set up and `.false.` otherwise.

```
subroutine sander_natom(integer natom)
```

This subroutine will query the currently set up system and return the number of atoms defined by the topology file used during setup. The input parameter will return with the number of atoms in the system or 0 if no system is currently set up.

```
subroutine get_inpcrd_natom(character(len=*) filename, integer natom)
```

This subroutine will open the specified file and try to read how many atoms are defined in that coordinate file. It supports both NetCDF and standard ASCII-formatted inpcrd and restart files. If there was an error in reading the file—either because the file does not exist, read permissions are not set, or the format is unrecognized—`natom` will return with a value of -1. Otherwise, `natom` returns with the number of atoms defined in the inpcrd file, `filename`.

```
subroutine read_inpcrd_file(character(len=*) filename,
                            double precision, dimension(3*natom) crd,
                            double precision, dimension(6) box,
                            integer ierr)
```

This subroutine will read the specified coordinate file and fill the `crd` and `box` arrays with the coordinates and box defined in the file. Both NetCDF restart files and ASCII restart files are supported. If no box is defined in the specified file, the `box` array is initialized to 0. The coordinate array is expected to be allocated with the appropriate amount of space. You can call `get_inpcrd_natom` (described above) to determine how large the coordinate array must be.

If there is a problem reading the file—either because the file does not exist, read permissions are not set, or the format is unrecognized—`ierr` will come back with a value of 1 and the `crd` array will be uninitialized (the `box` array will still be set to 0). If reading succeeded, `ierr` will come back as 0. This function can be called regardless of whether a system is currently set up or not.

8.13.2.3. Example uses of the Fortran API

In this section we show a series of example programs that use the *sander* Fortran API. At the end of this section, we show how to compile your program using the same Fortran compiler you used to build Amber. We will assume

that you created a file with the same name as the program name using the `.F90` suffix. You are recommended to use this suffix for your own programs.

We do not do any error checking in these programs since it adds considerably to the length of the example programs. However, you are encouraged to make use of the error reporting in your own programs to avoid program crashes. Syntax highlighting is applied to make the code easier to read.

The first example we provide below shows a sample program that computes purely MM energies for a non-periodic system using one of the GB models available in Amber.

```
program sample1
    use sander_api, only: sander_input, gas_sander_input, &.
                          sander_setup, energy_forces, &
                          sander_cleanup, potential_energy_rec, &
                          get_inpcrd_natom, read_inpcrd_file, &
                          sander_natom
    implicit none
    double precision, allocatable, dimension(:) :: crd, frc
    double precision, dimension(6) :: box
    type(sander_input) :: inp
    type(potential_energy_rec) :: ene
    integer :: natom, ierr
    ! Find how many atoms are in our inpcrd file
    call get_inpcrd_natom("inpcrd", natom)
    allocate(crd(natom*3), stat=ierr)
    ! Parse the inpcrd file
    call read_inpcrd_file("inpcrd", crd, box, ierr)
    ! Set up input options to use igb=5 with 0.2M salt
    call gas_sander_input(inp, 5)
    inp%saltcon = 2.0d-1
    ! Set up our system
    call sander_setup("prmtop", crd, box, inp, ierr=ierr)
    ! Coordinate array is no longer needed
    deallocate(crd)
    ! Find out how big our force array must be
    call sander_natom(natom)
    allocate(frc(natom*3), stat=ierr)
    call energy_forces(ene, frc)
    ! Do whatever you want with the energies and forces
    ! ...
    ! Free up our memory
    call sander_cleanup
    deallocate(frc)
    return
end program sample1
```

The second example we provide shows how to use the Fortran API to compute the energy for a periodic system using a multiscale QM/MM Hamiltonian. We will treat residues 10, 11, 12, and 20 using the PDDG-PM3 Hamiltonian.

```
program sample2
    use sander_api, only: sander_input, pme_sander_input, &
                          sander_setup, energy_forces, &
                          sander_cleanup, potential_energy_rec, &
                          get_inpcrd_natom, read_inpcrd_file, &
                          sander_natom, qmmm_input_options, &
```

qm_sande.r_input

```
    implicit none
    double precision, allocatable, dimension(:) :: crd, frc
    double precision, dimension(6) :: box
    type(sander_input) :: inp
    type(qmmm_input_options) :: qm_inp
    type(potential_energy_rec) :: ene
    integer :: natom, ierr
    ! Find how many atoms are in our inpcrd file
    call get_inpcrd_natom("inpcrd", natom)
    allocate(crd(natom*3), stat=ierr)
    ! Parse the inpcrd file
    call read_inpcrd_file("inpcrd", crd, box, ierr)
    ! Set up input options to use PME with a 10A cutoff
    call pme_sander_input(inp)
    inp%cut = 10.d0
    inp%ifqnt = 1
    call qm_sander_input(qm_inp)
    qm_inp%qmmask = ":10-12,20"
    qm_inp%qm_theory = "PDDG-PM3"
    ! Set up our system
    call sander_setup("prmtop", crd, box, inp, qm_inp, ierr)
    ! Coordinate array is no longer needed
    deallocate(crd)
    ! Find out how big our force array must be
    call sander_natom(natom)
    allocate(frc(natom*3), stat=ierr)
    call energy_forces(ene, frc)
    ! Do whatever you want with the energies and forces
    ! ...
    ! Free up our memory
    call sander_cleanup
    deallocate(frc)
    return
end program sample2
```

To compile Fortran programs using the *sander* API, the compiler must be able to find the `sander_api` (or `sanderles_api`) module files, which are deposited in `$AMBERHOME/include` when you build AmberTools. You must also link `libsander.so` (or `libsanderles.so`) when you link your program. On Mac OS X, these shared libraries are named `libsander.dylib` and `libsanderles.dylib` instead.

The programs we've written above are simple enough that they can be compiled and linked at the same time. The following command should compile the `sample1` program above, assuming it was saved to a file called `sample1.F90`. Note, make sure you use the same compiler you used to build AmberTools in the first place.

```
gfortran -I$AMBERHOME/include -L$AMBERHOME/lib -o sample1 sample1.F90 -lsander
```

This command will create a program called *sample1* that you can run from the command-line. Of course as it is written, the program will require that the files prmtop and inpcrd be present in the current directory. It will initialize the *sander* API, compute the energy, and quit without printing anything. Feel free to experiment with your own modifications to these programs.

8.13.2.4. Using the LES Fortran API

To use the LES functionality, you need to use the `sanderles_api` module instead of `sander_api` and you have to link to the sanderles library instead of the sander library (i.e., change `-lsander` to `-lsanderles` in the above compilation step). Since both libraries define most of the same symbols, you unfortunately cannot link both libraries to the same program. For example:

```
gfortran -I$AMBERHOME/include -L$AMBERHOME/lib -o sample1 sample1.F90 -lsanderles
```

8.13.3. The C and C++ APIs

This section describes how to use the C and C++ APIs. These two APIs are the same, and operate very much like a prototypical C API. This is because C and Fortran are both procedural languages (as opposed to object-oriented, like C++). Therefore, Fortran functionality maps more completely onto C than it does onto C++.

The function prototypes and data structures used for the C and C++ APIs are defined in the `sander.h` header file that is installed to `$AMBERHOME/include` when you build AmberTools.

8.13.3.1. Data Structures

The C and C++ data structures are all different `structs`. Variables within a `struct` are accessed using the `.` operator.

The sander input options are available as the type `sander_input`, the QM/MM input options are available as the type `qmmm_input_options`, and the potential energy data structure is available as the type `pot_ene`.

An example of using the `sander_input` type is shown in the small code fragment below.

```
#include "sander.h"
sander_input inp;
inp.cut = 9999.0;
inp.ifqnt = 0;
inp.igb = 5;
```

8.13.3.2. Function call syntax

This section details the function calls for the various functions defined in the `sander.h` header file. The syntax is almost identical to the Fortran syntax, except that errors codes are typically returned by the function rather than set in the final input parameter.

```
void gas_sander_input(sander_input *inp, int igb)
```

Unlike the Fortran API, the GB parameter is not optional. This subroutine takes a pointer to a `sander_input` instance and the GB model you wish to use (0 or 6 for vacuum). If an illegal igb value is provided, a warning is printed to `stderr` and a value of 6 is given to `inp->igb`. See table 8.1 for the default values assigned to each variable.

```
void pme_sander_input(sander_input *inp)
```

This subroutine takes a pointer to a `sander_input` instance and initializes every variable inside with the value listed in table 8.1.

```
void qm_sander_input(qmmm_input_options *inp)
```

This subroutine takes a `qmmm_input_options` instance and initializes all of the variables to the values given in Subsection 8.13.1.1. This is the recommended way to initialize the QM/MM options type.

```
int sander_setup_mm(const char* top, double *crd,
                    double *box, sander_input *inp)
```

This function sets up *sander* with the given topology file name, given coordinates, given box dimensions, and input options. Since overloading is not permitted in C, the QM/MM input `struct` cannot be made optional. Therefore, this function can only be used when `inp->ifqnt` is 0. This function returns 0 upon success or 1 upon failure. A system is only considered set up if this function returns 0.

No checking is done to make sure that the number of coordinates provided is correct compared to the number of atoms defined in the topology file. Note that answers will be ridiculous if the coordinate order does not match the atom order in the topology file. Segfaults and other memory violations are possible if the provided coordinate array or box array are too small.

The box array is given in the format $(a, b, c, \alpha, \beta, \gamma)$, which is the same as the format used at the bottom of the input coordinate and restart files. This argument is required even if the system is not periodic, but the values are not used (so they can be initialized to anything).

This function should never be called if a system is already set up.

```
int sander_setup(const char* top, double *crd,
                 double *box, sander_input *inp,
                 qmmm_input_options *qm_inp)
```

This function does the same thing as `sander_setup_mm` described above, but it also requires a pointer to a `qmmm_input_options` instance. If `inp->ifqnt` is set to 0, the contents of `qm_inp` are ignored and a standard MM system is set up. If successful, this function returns 0. Otherwise, it returns 1.

This function should never be called if a system is already set up.

```
void set_positions(double *crd)
```

This function sets the current positions of the active system (and so can only be called if a system is currently set up). Note that the onus is on the programmer to make sure that the coordinate array is large enough. No error checking is done. The input parameter is guaranteed not to change.

```
void set_box(double a, double b, double c,
             double alpha, double beta, double gamma)
```

This function sets the box dimensions and angles from the input parameters (which are guaranteed not to change).

```
void get_positions(double *positions)
```

This function gets the "active" positions for the system that is currently set up.

```
void energy_forces(pot_ene *ener, double *forces)
```

This function will compute the energies and forces from the current conformation of the system that is currently set up and populate the `ener` type and `forces` array with the resulting values. Those parameters are purely output. The energies are all given in units of kilocalories per mole and forces are given in $kcal/mol/\AA$. This subroutine can only be called if a system is currently set up.

```
void sander_cleanup(void)
```

This function will deallocate all memory used by the *sander* API and return it to a state where no system is set up and a new one can be initialized.

```
int is_setup(void)
```

This function can be called to query whether there is currently a system set up for the *sander* API. It returns 0 if no system is set up and 1 if a system is set up.

```
int sander_natom(void)
```

This funcion will query the currently set up system and return the number of atoms defined by the topology file used during setup. If no system is set up, this function returns 0.

```
int get_inpcrd_natom(const char *filename)
```

This function will open the specified file and try to read how many atoms are defined in that coordinate file. It supports both NetCDF and standard ASCII-formatted inpcrd and restart files. If there was an error in reading the file—either because the file does not exist, read permissions are not set, or the format is unrecognized—the return value will be -1. Otherwise, this function returns the number of atoms defined in the inpcrd file, `filename`.

```
int read_inpcrd_file(const char* filename, double *crd, double *box)
```

This subroutine will read the specified coordinate file and fill the `crd` and `box` arrays with the coordinates and box defined in the file. Both NetCDF restart files and ASCII restart files are supported. If no box is defined in the specified file, the `box` array is initialized to 0. The coordinate array is expected to be allocated with the appropriate amount of space. You can call `get_inpcrd_natom` (described above) to determine how large the coordinate array must be.

If there is a problem reading the file—either because the file does not exist, read permissions are not set, or the format is unrecognized—this function will return 1 and the `crd` array will be uninitialized (the `box` array will still be set to 0). If reading succeeded, this function will return 0. This function can be called regardless of whether a system is currently set up or not.

8.13.3.3. Examples and uses of the C and C++ APIs

In this section, we show examples of how to use the C and C++ API. These samples do exactly the same thing as the two examples in Subsection 8.13.2.3. At the end of this section, we show how to compile your C or C++ program.

We do not do any error checking in these programs since it adds considerably to the length of the example programs. However, you are encouraged to make use of the error reporting in your own programs to avoid program crashes. Syntax highlighting is applied to make the code easier to read.

The first example we provide below shows a sample C program that computes purely MM energies for a non-periodic system using one of the GB models available in Amber.

```
#include <stdlib.h>
#include "sander.h"
int main() {
    sander_input inp;
    double *crd, *frc;
    double box[6];
    pot_ene ene;
    int natom, ierr;
    // Find out how many atoms are in our inpcrd file
    natom = get_inpcrd_natom("inpcrd");
    crd = (double*) malloc(natom*3*sizeof(double));
    ierr = read_inpcrd_file("inpcrd", crd, box);
    // Set up input options to use igb=5 with 0.2M salt
    gas_sander_input(&inp, 5);
    inp.saltcon = 0.2;
    // Set up our system
    ierr = sander_setup_mm("prmtop", crd, box, &inp);
    // Coordinate array is no longer needed
    free(crd);
    // Find out how big our force array must be
    frc = (double*) malloc(sander_natom()*3*sizeof(double));
    energy_forces(&ene, frc);
    /* Do whatever you want with the energies and forces
     * ...
```

```
     * Free up our memory
     */
    sander_cleanup();
    free(frc);
    return 0;
}
```

The second example we provide shows how to use the C API to compute the energy for a periodic system using a multiscale QM/MM Hamiltonian (in a C++ program this time). We will treat residues 10, 11, 12, and 20 using the PDDG-PM3 Hamiltonian.

```
#include "sander.h"
#include <cstring>
int main() {
    sander_input inp;
    double *crd, *frc;
    double box[6];
    pot_ene ene;
    int natom, ierr;
    // Find out how many atoms are in our inpcrd file
    natom = get_inpcrd_natom("inpcrd");
    crd = new double[natom*3];
    ierr = read_inpcrd_file("inpcrd", crd, box);
    // Set up input options to use igb=5 with 0.2M salt
    pme_sander_input(&inp);
    inp.cut = 10.0;
    qm_sander_input(&qm_inp);
    strncpy(qm_inp.qmmask, ":10-12,20", 9);
    strncpy(qm_inp.qm_theory, "PDDG-PM3", 8);
    // Set up our system
    ierr = sander_setup("prmtop", crd, box, &inp, &qm_inp);
    // Coordinate array is no longer needed
    delete[] crd;
    // Find out how big our force array must be
    frc = new double[sander_natom()*3];
    energy_forces(&ene, frc);
    /* Do whatever you want with the energies and forces
     * ...
     * Free up our memory
     */
    sander_cleanup();
    delete[] frc;
    return 0;
}
```

To compile C or C++ programs using the *sander* API, the compiler must be able to find the `sander.h` header file, which are deposited in `$AMBERHOME/include` when you build AmberTools. You must also link `libsander.so` (or `libsanderles.so`) when you link your program. On Mac OS X, these shared libraries are named `libsander.dylib` and `libsanderles.dylib` instead.

The programs we've written above are simple enough that they can be compiled and linked at the same time. The following command should compile the `sample1` program above, assuming it was saved to a file called `sample1.F90`. Note, make sure you use the same compiler you used to build AmberTools in the first place.

```
gcc -I$AMBERHOME/include -L$AMBERHOME/lib -o sample1 sample1.c -lsander
```

This command will create a program called *sample1* that you can run from the command-line. Of course as it is written, the program will require that the files prmtop and inpcrd be present in the current directory. It will initialize the *sander* API, compute the energy, and quit without printing anything. Feel free to experiment with your own modifications to these programs. For the second sample, you need to use a C++ compiler instead of the C compiler.

8.13.3.4. Using the LES C/C++ API

There is only one header file for the *sander* C/C++ API. The LES and standard functionalities are differentiated using the LES preprocessor directive. To use the LES functionality, you need to define the LES macro. You can either do this in the source code (by putting "`#define LES 1`" before `#include "sander.h"`) or by compiling with the `-DLES` flag. If you use the `LES` symbol (either as a variable or a preprocessor macro), you will have to implement this in the source code and undefine the macro after `sander.h` is included. For example, on the command line this would look like:

```
gcc -DLES -I$AMBERHOME/include -L$AMBERHOME/lib -o sample1 sample1.c -lsanderles
```

8.13.4. The Python API

This section describes how to use the Python API so that you can use *sander* functionality inside your own Python scripts. Building the Python bindings requires that the Python development headers and libraries be installed. As long as you install the recommended packages listed on *https://ambermd.org/amber_install.html* for your Linux, Mac or Windows distribution, the necessary prerequisites will be installed.

The *sander* functionality is implemented in the `sander` Python module. The *sander* LES functionality is implemented in the `sanderles` module. While the Python API implements the functions described on page 127, the semantics of how these functions are used in Python differs more than the difference between the C/C++ and Fortran APIs.

The Python API has numerous advantages over the other options. First, processing strings is handled correctly by the boilerplate that interfaces Python with C, meaning that the programmer does not have to worry about how strings map to the underlying Fortran code. Second, data is always initialized, so the programmer does not have to worry about bugs arising from uninitialized variables. Finally, array sizes are determined automatically and no allocation or deallocation is required.

Furthermore, the Python API provided here interacts with other Python packages provided as part of AmberTools—specifically several of the classes provided by ParmEd. See Section **??** for more information (specifically Subsection **??** for the ParmEd Python API documentation).

8.13.4.1. Data Structures

The data structures in the Python API are all "restricted" classes, where restricted means setting new attributes is not supported and will raise an `AttributeError`. The data types for the *sander* input options, QM/MM input options, and potential energy terms are the classes `InputOptions`, `QmInputOptions`, and `EnergyTerms`, respectively. The last class is part of the private `sander._pys` namespace since it is only produced as output and never needed as input, whereas the first two are members of the `sander` package namespace.

Unlike C and Fortran, the Python classes have default constructors that will initialize all of the variables for the different classes. An example of using the `InputOptions` class is shown below.

```
import sander
inp = sander.InputOptions()
inp.cut = 9999.0
inp.extdiel = 78.5
inp.intdiel = 1
```

8.13.4.2. Function call syntax

This section details the function calls for the various functions defined in the `sander` package.

```
inp = sander.gas_input(6)
```

The `igb` argument is an optional integer that defaults to 6 (vacuum). This function returns an initialized `InputOptions` instance whose values are listed in table 8.1. If an illegal igb value is provided, a `ValueError` is raised.

```
inp = sander.pme_input()
```

This subroutine returns a `InputOptions` instance and initializes every member with the value listed in table 8.1.

```
qm_inp = sander.qm_input()
qm_inp = sander.QmInputOptions()
```

These two commands both return a `QmInputOptions` instance and initializes all of the variables to the values given in Subsection 8.13.1.1. The function (`sander.qm_input`) is redundant, since the `QmInputOptions` constructor does the same thing. The function was provided only for consistency with the Fortran and C/C++ APIs.

```
sander.setup(prmtop, coordinates, box, mm_options, qm_options=None)
```

This function sets up *sander* with the given topology file, coordinates, box dimensions, and input options. If `mm_options.ifqnt` is 1 and `qm_options` is not provided, a `ValueError` is raised. The topology file can be either an `AmberParm` instance (see Subsection **??** for more information) or a string filename pointing to a valid Amber topology file.

The coordinate array can either be a `numpy.ndarray` instance an `array.array` instance, or a `list`. The array must be 1-dimensional with a length equal to $3 \times$ natom. In particular, the coordinate array taken from a `Rst7` instance can be used. Alternatively, the `coordinates` argument can be a string that is the filename of a coordinate or restart file.

The box array, too, can be a `numpy.ndarray`, `array.array`, or `list` instance of length 6. If it is not one of those data types, a `TypeError` will be raised. If it does not have 6 elements, a `ValueError` will be raised. If no box is needed, the `box` argument can be set to `None`. Alternatively, if `box` is set to `None` and a filename was passed to the `coordinates` argument that contains box dimensions, the box will be set from the information in that file. However, any box dimensions passed using the `box` argument will take precedence.

The `mm_options` must be a `InputOptions` instance or a TypeError will be raised. The `qm_options` must be a `QmInputOptions` instance or `None`. Otherwise, a `TypeError` will be raised.

If there is any problem setting the system up, or if a system is already set up, a `RuntimeError` will be raised. This "function" is actually a class that implements the context manager protocol via the `with` statement (Python 2.5 or greater, only—Python 2.4 users must use the syntax above).

```
with sander.setup(prmtop, coordinates, box, mm_options, qm_options=None):
    ... do stuff
```

The return value of `sander.setup` is a reference to the class (which itself can be used in a context manager). Upon exiting the context manager, `sander.cleanup` is called (but only if `sander.setup` succeeded).

```
sander.set_positions(crd)
```

This function sets the current positions of the active system. The `crd` argument can be a `numpy.ndarray`, `array.array`, or `list` instance and must be either 1-dimensional with a length $3 \times$ natom or 2-dimensional with a shape of natom, 3. If the array is not the correct length, a `ValueError` will be raised. If it is not one of the aforementioned types, a `TypeError` will be raised. If a system is not currently set up, a `RuntimeError` will be raised. This function returns `None`.

```
sander.set_box(a, b, c, alpha, beta, gamma)
```

This function sets the box dimensions and angles from the input parameters. If the incorrect number of arguments are given, or if the arguments are not all numbers, a `TypeError` is raised. If no system is currently set up, a `RuntimeError` is raised. This function returns `None`.

```
positions = sander.get_positions()
```

This function returns the coordinates as a one-dimensional list for the currently active system. If no system is currently set up, a `RuntimeError` is raised.

```
ene, frc = sander.energy_forces()
```

This function will compute the energies and forces from the current conformation of the system that is currently set up and returns a two-element `tuple` in which the first element is an `EnergyTerms` instance with the attributes listed in Subsection 8.13.1.1 and the second attribute is a $3 \times$ natom-length `list` with the atomic forces. The energies are all given in units of kilocalories per mole and forces are given in $kcal/mol/\text{Å}$. A `RuntimeError` is raised if no system is currently set up.

```
sander.cleanup()
```

This function will deallocate all memory used by the *sander* API and return it to a state where no system is set up and a new one can be initialized. If no system is set up, a `RuntimeError` is raised. This function returns `None`.

```
bool = sander.is_setup()
```

This function can be called to query whether there is currently a system set up for the *sander* API. It returns `False` if no system is set up and `True` if a system is set up.

```
natom = sander.natom()
```

This function will query the currently set up system and return the number of atoms defined by the topology file used during setup. If no system is set up, this function raises a `RuntimeError`.

Coordinate file parsing No functions are provided to parse and query coordinate and restart files, since the `Rst7` class from the parmed.amber package already does that. Examples using this class are shown in the next section.

8.13.4.3. Examples and uses of the Python API

In this section, we show examples of how to use the Python API. These samples do exactly the same thing as the two examples in Subsection 8.13.2.3 and Subsection 8.13.3.3.

Unlike the previous APIs, the Python API has built-in error checking through the utilization of the Exception mechanism. The various exceptions that can be raised and the circumstances in which they will be raised are described in the previous section. You may wish to catch some of the exceptions in your own Python scripts to implement more elaborate error handling. Notice that the Python program here is much simpler than the equivalent Fortran and C programs presented earlier.

These programs also make use of the `AmberParm` class in the chemistry package that is part of the ParmEd program (see Section **??**).

```
import sander
from parmed.amber.readparm import AmberParm
# Initialize the topology object with coordinates
parm = AmberParm("prmtop", "inpcrd")
# Set up input options to use igb=5 with 0.2M salt
inp = sander.gas_input(5)
```

```
inp.saltcon = 0.2
sander.setup(parm, parm.coordinates,None, inp)
# Compute the energies and forces
ene, frc = sander.energy_forces()
# Do whatever you want with the energies and forces
# ...
# Free up our memory
sander.cleanup()
```

The second example we provide shows how to use the Python API to compute the energy for a periodic system using a multiscale QM/MM Hamiltonian. We will treat residues 10, 11, 12, and 20 using the PDDG-PM3 Hamiltonian. Also, rather than loading the inpcrd file directly into the AmberParm object, we use the `open` constructor of the `Rst7` class to read in the coordinate file. While this is exactly what the `AmberParm` class does under the hood, this approach is presented here to show how to use the `Rst7` class in your own programs.

```
import sander
from parmed.amber.readparm import AmberParm, Rst7
# Initialize the topology object with coordinates
parm = AmberParm("prmtop")
rst = Rst7.open("inpcrd")
# Set up input options to use PME with a 10A cutoff
inp = sander.gas_input(5)
inp.cut = 10.0
qm_inp = sander.QmInputOptions()
qm_inp.qmmask = ":10-12,20"
qm_inp.qm_theory = "PDDG-PM3"
sander.setup(parm, rst.coords, rst.box, inp, qm_inp)
# Compute the energies and forces
ene, frc = sander.energy_forces()
# Do whatever you want with the energies and forces
# ...
# Free up our memory
sander.cleanup()
```

One final thing we will mention is that the sander Python API supports the context manager protocol! The previous example can be rewritten as

```
import sander
from parmed.amber.readparm import AmberParm, Rst7
# Initialize the topology object with coordinates
parm = AmberParm("prmtop")
rst = Rst7.open("inpcrd")
# Set up input options to use PME with a 10A cutoff
inp = sander.gas_input(5)
inp.cut = 10.0
qm_inp = sander.QmInputOptions()
qm_inp.qmmask = ":10-12,20"
qm_inp.qm_theory = "PDDG-PM3"
with sander.setup(parm, rst.coords, rst.box, inp, qm_inp):
    # Compute the energies and forces
    ene, frc = sander.energy_forces()
# Do whatever you want with the energies and forces
# ...
# Free up our memory
```

When the context manager is exited (i.e., when program execution is no longer inside the `with` block), sander is automatically cleaned up. This occurs regardless of whether or not an error was raised during the execution of the code within the `with` block. Notice how `sander.cleanup()` is no longer necessary.

8.13.4.4. Using the LES Python API

To use the LES functionality in Python, you need to import the `sanderles` package instead of the `sander` package. Note that while nothing stops you from importing both the `sander` and `sanderles` packages in the same Python script, both packages will not work correctly in the same script.

9. Atom and Residue Selections

There are three ways to select atoms and residues in AMBER-related routines: the **AMBER "mask"** notation, used by most programs, the **NAB "atom expressions"**, which work only with NAB-compiled applications, and an older "GROUP" specification used in *sander* and *pmemd*. Information about these is collected in this chapter.

9.1. Amber Masks

A "mask" is a notation which selects atoms or residues for special treatment. A frequent usage is fixing or tethering selected atoms or residues during minimization or molecular dynamics.

The following lines are partially copied from the original AMBER documentation. For more details, refer to the entire section of that documentation describing the *ambmask* utility.

The "mask" selection expression is composed of "elementary selections". These start with "**:**" to select by residues, or "@" to select by atoms. Residues can be selected by numbers (given as numbers separated by commas, or as ranges separated by a dash) or by names (given as a list of residue names separated by commas). The same holds true for atom selections by atom numbers or atom names. In addition, atoms can be selected by AMBER atom type, in which case "@" must be immediately followed by "%". The notation ":*" means all residues and "@*" means all atoms. The following examples show the usage of this syntax.

Residue Number List Examples

```
:1-10      = "residues 1 to 10"
:1,3,5     = "residues 1, 3, and 5"
:1-3,5,7-9 = "residues 1 to 3 and residue 5 and residues 7 to 9"
```

Residue Name List Examples

```
:LYS          = "all lysine residues"
:ARG,ALA,GLY  = "all arginine and alanine and glycine residues"
```

Atom Number List Examples

Note that these masks use the **actual sequential numbers of atoms** in the file. This is tricky and a serious source of error. You must know these numbers correctly. Using the atom numbers of a PDB file written out by an AMBER tool is an appropriate way to avoid pitfalls. **Do not use the original atom numbers from the raw PDB file you started with.**

```
@12,17       = "atoms 12 and 17"
@54-85       = "all atoms from 54 to 85"
@12,54-85,90 = "atom 12 and all atoms from 54 to 85 and atom 90"
```

Atom Name List Examples

```
@CA         = all atoms with the name CA (i.e., all C-alpha atoms)
@CA,C,O,N,H = all atoms with names CA or C or O or N or H
                  (i.e., the entire protein backbone)
```

Atom Type List Examples

This last mask type is only used by specialists and mentioned here for completeness. It allows the selection of AMBER atom types and requires detailed knowledge of AMBER force fields.

```
@%CT         = all atoms with the force field type CT
               (the standard sp3 aliphatic carbon)
@%N*,N3      = all atoms with the force field type N* or N3
               (N* is a special sp2 nitrogen, N3 is an sp3 nitrogen)
```

Note that in the above example, `N*` is actually an atom type. The * is **not** a wild card meaning "all N-something types"!

Logical Combinations

The selections above can be combined by various logical operators, including selections like "all atoms within a certain distance from...". The use of such combinations goes beyond this introductory script. Interested users should refer to the next section for details.

9.1.1. ambmask

NAME

ambmask - test group input FIND mask (or mask string given in the &cntrl section) and dump the resulting atom selection in a given format

SYNOPSIS

```
ambmask -p prmtop -c inpcrd -prnlev [0-3] -out [short| pdb| amber] -find [maskstr]
```

DESCRIPTION

ambmask acts as a filter that inputs an Amber topology file and an Amber coordinate file and applies the "maskstr" selection string to select specific atoms or residues. (The "maskstr" selection string is similar syntactically to UCSF Chimera/Midas.) Residues can be selected by their numbers or names. Atoms can be selected by numbers, names, or Amber (forcefield) type. Selections are case insensitive. The selected atoms are printed to **stdout** (by default, in Amber-style PDB format). Atom and residue names and numbers are taken from the Amber topology. Beware that the selection string works on those names and not the ones from the original PDB file. If you are not sure how atoms or residues are named or numbered in the Amber topology, use **ambmask** with a selection string ":*" (which is the default) to dump the whole PDB file with corresponding Amber atom/residue names and numbers.

The "maskstr" selection expression is composed of "elementary selections". These start with ":" to select by residues, or "@" to select by atoms. Residues can be selected by numbers (given as numbers separated by commas, or as ranges separated by a dash) or by names (given as a list of residue names separated by commas). The same holds true for atom selections by atom numbers or atom names. In addition, atoms can be selected by Amber atom type, in which case "@" must be immediately followed by "%". ":*" means all residues and "@*" means all atoms. The following examples show the usage of this syntax. Square brackets should not be used in actual expressions, they are only used below to denote individual selection string examples:

```
:{residue numlist} [:1-10] [:1,3,5] [:1-3,5,7-9]
:{residue namelist} [:LYS] [:ARG,ALA,GLY]
@{atom numlist} [@12,17] [@54-85] [@12,54-85,90]
@{atom namelist} [@CA] [@CA,C,O,N,H]
@%{atom typelist} [@%CT] [@%N*,N3]
```

These "elementary selections" can be combined into more complex selections using binary operators "&" (and) and "|" (or), unary operator "!" (negation), distance binary operators "<:", ">:", "<@", ">@", and parentheses. Spaces around operators are irrelevant. Parentheses have the highest priority, followed by distance operators ("<:", ">:", "<@", ">@"), "!" (negation), "&" (and) and "|" (or) in order of descending priority. A wildcard "=" in an atom or residue name matches any name starting with a given character (or characters). For example, [:AS=] would match all aspartic acid residues (ASP), and asparagines (ASN); [@H=] would match all atom names starting with H (which are effectively all hydrogens). It cannot be used to match the end part of names (such as [:=A]). Some examples of more complex selections follow:

```
[@C= & !@CA,C]
```

.. all carbons except backbone alpha and carbonyl carbon

```
[(:1-3@CA | :5-7@CB)]
```

.. alpha carbons in residues 1-3 and beta carbons in residues 5-7

```
[:CYS,ARG & !(:1-10 | @CA,CB)]
```

.. all CYS and ARG atoms except those which are in residues 1-10 and which are CA or CB

```
[:* & !@H=] or [!@H=]
```

.. all heavy atoms (i.e. except hydrogens)

```
[:5 <@4.5]
```

.. all atoms within 4.5A from residue 5

```
[(:1-55 <:3.0) & :WAT]
```

.. all water molecules within 3A from residues 1-55

Compound expressions of the following type are also allowed:

```
:{residue numlist|namelist}@{atom numlist|namelist|typelist}
[:1-10@CA] is equivalent to [:1-10 & @CA]
[:LYS@H=] is equivalent to [:LYS & @H=]
```

OPTIONS

The program needs an Amber topology file and coordinates (restrt format). The filename specified with the *-p* option is Amber topology, while the filename given with the *-c* option is a coordinate file. If *-p* or *-c* options are not given, the program expects that files "prmtop" and/or "inpcrd" exist in the current directory, which will be taken as topology and coordinate files correspondingly. If no command line options are given, the program prints the usage statement.

The option *-prnlev* specifies how much (debugging) information is printed to **stdout**. If it is 0, only selected atoms are printed. More verbose output (which might be useful for debugging purposes) is achieved with higher values: 1 prints original "maskstr" in its tokenized (with operands enclosed in square brackets) and postfix (or Reverse Polish Notation) forms; number of atoms and residues in the topology file and number of selected atoms are also printed to **stdout**. 2 prints the resulting mask array, which is an array of integer values, with '1' representing a selected atom, and '0' an unselected one. Value of 3, in addition, prints mask arrays as they are pushed or popped from the stack (this is really only useful for tracing the problems occurring during stack operations). The *-prnlev* values of 0 or 1 should suffice for most uses.

The option *-out* specifies the format of printed atoms. "short" means a condensed output using residue (:) and atom (@) designators followed by residue ranges and atom names. "pdb" (default) prints atoms in Amber-style PDB format with the original "maskstr" printed as a REMARK at the top of the PDB file, and "amber" prints atom/residue ranges in the format suitable for copying into group input section of Amber input file.

The option *-find* is followed by "maskstr" expression. This is a string where some characters have a special meaning and thus express what parts (atoms/residues) of the molecule will get selected. The syntax of this string is explained in the section above (DESCRIPTION). If this option is left out, it defaults to ":*", which selects all atoms in the given topology file. The length of "maskstr" is limited to 80 characters. If the "maskstr" contains spaces or special characters (which would be expanded by the shell), it should be protected by single or double quotes (depending on the shell). In addition, for C-shells even a quoted exclamation character may be expanded for history substitution. Thus, it is recommended that the operand of the negation operator always be enclosed in parentheses so that "!" is always followed by a "(" to produce "!(" which disables the special history interpretation. For example, [@C= & !(@CA,C)] selects all carbons except backbone alpha and carbonyl carbon; the parentheses are redundant but shell safe. The man page indicates further ways to disable history substitution.

FILES

Assumes that *prmtop* and *inpcrd* files exists in the current directory if they are not specified with *-p* and *-c* options. Resulting (i.e. selected) atoms are written to **stdout**.

BUGS

Because all atom names are left justified in Amber topology and the selections are case insensitive, there is no way to distinguish some atom names: alpha carbon CA and a calcium ion Ca are a notorious example of that.

9.2. GROUP Specification

This section describes the format used to define groups of atoms in various Amber programs. In *sander*, a group can be specified as a movable "belly" while the other atoms are fixed absolutely in space (aside from scaling caused by constant pressure simulation), and/or a group of movable atoms can independently restrained (held by a potential) at their positions. In *anal*, groups can be defined for energy analysis. In sander and pmemd, GROUP input comes at the end of the mdin input file, as discussed in Section 8.5.

Except in the analysis module where different groups of atoms are considered with different group numbers for energy decomposition, in all other places the groups of atoms defined are considered as marked atoms to be included for certain types of calculations. In the case of constrained minimization or dynamics, the atoms to be constrained are read as groups with a different weight for each group.

Reading of groups is performed by the routine RGROUP, and you are advised to consult it if there is still some ambiguity in the documentation.

Input description:

```
- 1 - Title format(20a4)
ITITL Group title for identification.
Setting ITITL = 'END' ends group input.
-------------------------------------------------------------------
- 1A - Weight format(f)
This line is only provided/read when using GROUP input to
define restrained atoms.
WT The harmonic force constants in kcal/mol-A**2 for the group
of atoms for restraining to a reference position.
-------------------------------------------------------------------
- 1B - Control to define the group
KTYPG , (IGRP(I) , JGRP(I) , I = 1,7) format(a,14i)
KTYPG Type of atom selection performed. A molecule can be
defined by using only 'ATOM' or 'RES', or part of the
molecule can be defined by 'ATOM' and part by 'RES'.
```

'ATOM' The group is defined in terms of atom numbers. The atom number list is given in igrp and jgrp.
'RES' The group is defined in terms of residue numbers. The residue number list is given in igrp and jgrp.
'FIND' This control is used to make additional conditions (apart from the 'ATOM' and 'RES' controls) which a given atom must satisfy to be included in the current group. The conditions are read in the next section (1C) and are terminated by a SEARCH card.
Note that the conditions defined by FIND filter any set(s) of atoms defined by the following ATOM/RES instructions. For example,

```
-- group input: select main chain atoms --
FIND
* * M *
SEARCH
RES 1 999
END
END
```

'END' End input for the current group. Followed by either another group definition (starting again with line 1 above), or by a second 'END' "card", which terminates all group input.
IGRP(I) , JGRP(I)
The atom or residue pointers. If ktypg .eq. 'ATOM' all atoms numbered from igrp(i) to jgrp(i) will be put into the current group. If ktypg .eq. 'RES' all atoms in the residues numbered from igrp(i) to jgrp(i) will be put into the current group. If igrp(i) = 0 the next control card is read.
It is not necessary to fill groups according to the numerical order of the residues. In other words, Group 1 could contain residues 40-95 of a protein, Group 2 could contain residues 1-40 and Group 3 could contain residues 96-105.
If ktypg .eq. 'RES', then associating a minus sign with igrp(i) will cause all residues igrp(i) through jgrp(i) to be placed in separate groups.
In the analysis modules, all atoms not explicitly defined as members of a group will be combined as a unit in the (n + 1) group, where the (n) group in the last defined group.

- 1C - Section to read atom characteristics
******* Read only if KTYPG = 'FIND' *******
JGRAPH(I) , JSYMBL(I) , JTREE(I) , JRESNM(I) ***format(4a)***
A series of filter specifications are read. Each filter consists of four fields (JGRAPH,JSYMBL,JTREE,JRESNM), and each filter is placed on a separate line. Filter specification is terminated by a line with JGRAPH = 'SEARCH'. A maximum of 10 filters may be specified for a single 'FIND' command.
The union of the filter specifications is applied to the atoms defined by the following ATOM/RES cards. I.e. if an atom satisfies any of the filters, it will be included in the current group. Otherwise, it is not included. For example, to select all non main chain atoms from residues 1 through 999:

```
-- group input: select non main chain atoms --
FIND
* * S *
```

```
* * B *
* * 3 *
* * E *
SEARCH
RES 1 999
END
END
'END' End input for the current group. Followed by either another
The four fields for each filter line are:
JGRAPH(I) The atom name of atom to be included. If this and the
following three characteristics are satisfied the atom is
included in the group. The wild card '*' may be used to
to indicate that any atom name will satisfy the search.
JSYMBL(I) Amber atom type of atom to be included. The wild card
'*' may be used to indicate that any atom type will
satisfy the search.
JTREE(I) The tree name (M, S, B, 3, E) of the atom to be included.
The wild card '*' may be used to indicate that any tree
name will satisfy the search.
JRESNM(I) The residue name to which the atom has to belong to be
included in the group. The wild card '*' may be used to
indicate that any residue name will satisfy the search.
--------------------------------------------------------------------
```

Examples:

The molecule 18-crown-6 will be used to illustrate the group options. This molecule is composed of six repeating (-CH2-O-CH2-) units. Let us suppose that one created three residues in the PREP unit: CRA, CRB, CRC. Each of these is a (-CH2-O-CH2-) moiety and they differ by their dihedral angles. In order to construct 18-crown-6, the residues CRA, CRB, CRC, CRB, CRC, CRB are linked together during the LINK module with the ring closure being between CRA(residue 1) and CRB(residue 6).

Input 1:

```
Title one
RES 1 5
END
Title two
RES 6
END
END
```

Output 1: Group 1 will contain residues 1 through 5 (CRA, CRB, CRC, CRB, CRC) and Group 2 will contain residue 6 (CRB).

Input 2:

```
Title one
RES 1 5
END
Title two
ATOM 36 42
END
END
```

Output 2: Group 1 will contain residues 1 through 5 (CRA, CRB, CRC, CRB, CRC) and Group 2 will contain atoms 36 through 42. Coincidentally, atoms 36 through 42 are also all the atoms in residue 6.

Input 3:

```
Title one
```

```
RES -1 6
END
END
```

Output 3: Six groups will be created; Group 1: CRA, Group 2: CRB,..., Group 6: CRB.

Input 4:

```
Title one
FIND
O2 OS M CRA
SEARCH
RES 1 6
END
END
```

Output 4: Group 1 will contain those atoms with the atom name 'O2', atom type 'OS', tree name 'M' and residue name 'CRA'.

Input 5:

```
Title one
FIND
O2 OS * *
SEARCH
RES 1 6
END
END
```

Output 5: Group 1 will contain those atoms with the atom name 'O2', atom type 'OS', any tree name and any residue name.

Input 6:

```
Title one
RES 1 3 6 6
END
END
```

Output 6: One group is created containing residues 1 to 3 and 6. Up to seven ranges of contiguous residues can be specified per group. (In this case there are two ranges).

Input 7:

```
First restraint group
10.0
FIND
CA * * *
SEARCH
RES 1 17   25 36
END
Second restraint group, with a different restraint weight
1.0
FIND
CA * * *
SEARCH
RES 61 127
END
END
```

Output 7: CA atoms in residues 1-17 and 25-36 will be restrained to their initial positions with a strong weight of $10.0 \text{kcal} \cdot \text{mol}^{-1} \cdot \text{Å}^{-2}$; CA atoms in residues 61 to 127 will have a weaker restraint force constant.

10. Sampling configuration space

The "middle" scheme [Section 8.6.10] offers an efficient approach to accurately sample configuration space in standard molecular dynamics simulations. There are many instances when standard molecular dynamics simulations get “stuck” near the starting configuration, and fail to adequately sample the available low-energy configurational space. This chapter describes a variety of techniques that can partially overcome such problems. The following chapter (on Free Energies) continues many of these ideas, adapting them to the calculation of alchemical or configurational free energy differences. There is no good distinction between these two chapters, because good sampling of the canonical distribution and estimation of free energies go hand-in-hand. But the present chapter covers methods that are *primarily* devoted to enhanced or accelerated sampling, whereas the following chapter considers methods that explicitly estimate free energy differences.

10.1. Self-Guided Langevin dynamics

Self-guided Langevin dynamics (SGLD) is designed to enhance conformational search efficiency in either a molecular dynamics (MD) simulation (when *gamma_ln=0*) or a Langevin dynamics (LD) simulation (when *gamma_ln>0*). This method accelerates low frequency motion to enhance conformational sampling. [184–187]

Overview: The input parameter, *tsgavg*, defines the lower limit period of the low frequency motion. Typically, *tsgavg*=0.2 ps is recommended for motions like phase separation, secondary structure folding, and ligand docking, while *tsgavg*=1.0 ps is recommended for protein domain motion, and protein-protein docking. The input parameter, *sgft* or *tempsg*, defines the strength of the guiding effect. *sgft*=0~1 with 0 for regular LD or MD simulations. A smaller *sgft* will produce results closer to a normal MD or LD simulation. *tempsg* defines a conformational search ability that is comparable to a high temperature simulation at *temp*0=*tempsg*. Normally, *tempsg* or *sgft* is set to accelerates slow events to an affordable time scale while minimizing the pertubation to the conformational distribution. The guiding force can be applied to a part of a simulation system between atom *isgsta* and atom *isgend*.

The conformational distribution of SGLD can be reweighted to produce canonical ensemble averages[185, 188]. The reweighting information is in the simulation output file. The force-momentum based SGLD algorithm (SGLDfp) [189] (*isgld*=2) is available to allow conformational search to be accelerated while the canonical ensemble distribution is maintained. However, the conformational searching abilities of SGLDfp is reduced as compared with SGLD (*isgld*=1). Most recently, the generalized SGLD method [187](SGLDg) (*isgld*=3) is developed to enhance conformational search in both LD (when *gamma_ln>0*) and MD (when *gamma_ln=0*) simulations. SGLDg is more convenient and flexible and has better characterized conformational distribution and can be an replacement of normal LD to sample the canonical ensemble by setting *tempsg*=*temp*0.

- SGLDfp (*isgld*=2) allows low resolution structures, such as secondary structures and tertiary structures, and/or high resolution structures, such as bond lengths and bond angles, to be canonically sampled.

- SGLDg (*isgld*=3) utilizes independent low-frequency and high frequency Langevin equations to characterize the conformational searching and distribution. It provides options to use the force guiding factor, *sgff*, to control energy barriers in low frequency space and to use the guiding temperature, *tempsg*, and the momentum guiding factor, *sgft*, to enhance low frequency motion. When *tempsg* is set to the simulation temperature (*tempsg*=*temp*0), SGLDg samples exactly the canonical ensemble and is an excellent replacement for regular Langevin dynamics with enhanced conformational sampling ability, especially when the friction constant is high, for example, *gamma_ln*>10/ps. SGLDg is recommended when a canonical distribution needs be maintained (*tempsg*=*temp*0) or reweighted (*tempsg*>*temp*0).

SGLD can be used for replica exchange simulations (RXSGLD)[190] to achieve enhanced sampling with or without elevating temperature. *sgft* or *tempsg* can be used to define different replicas. See Section **??** for a detailed description of RXSGLD.

`isgld` SGLD algorithm index. Default *isgld*= 0, SGLD is diabled; *isgld*=1 will turn on SGMD/SGLD method for accelerated conformational search; *isgld*=2 will turn on the SGLDfp method [189] to maintain a canonical ensemble distribution. *isgld*=3 will enable SGLDg/SGMDg method.

`tsgavg` Local averaging time (*psec*) for the guiding force calculation. Default 0.2 *psec*. A larger value defines slower motion to be enhanced.

`sgft` Momentum guiding factor. Defines the strength of the guiding effect. Default 1.0 when *gamma_ln>0* (SGLD), 0.2 when *gamma_ln=0* (SGMD), or 0 for SGLDg or 1 for SGMDg (*isgld*=3). *When isgld=1 or 2, tempsg*>0 will override *sgft*. When *isgld*=3 and *gamma_ln*=0, i.e., SGMDg, *sgft* represents the guiding friction constant in the unit of 1/ps and should be set to a positive value.

`sgff` Force guiding factor for SGLDg or SGMDg (*isgld*=3). *sgff* is used to scale down low frequency energy surface by a factor, (1+*sgff*). *sgff* is suggested to take values between 0 and -0.1, with default value of 0. *sgff* is effective for SGMDg (*isgld*=3 and *gamma_ln*=0) simulations .

`tempsg` Target guiding temperature (K). This parameter is redefined since Amber 12 as a conformational search ability which is comparable to a high temperature simulation with *temp0=tempsg*. For example, by seting *tempsg*=500K, a SGMD/SGLD simulation will accelerate conformational search as much as rising the simulation temperature to 500K. When *isgld=1* or *2*, the default is *tempsg=0* K. When *tempsg=0*, the guiding effect will be defined by *sgft*. *tempsg*>*temp*0 will accelerate a conformational search and *tempsg*<*temp*0 will slow down a conformational search. *When isgld=1 or 2,* once *tempsg* is set, *sgft* will fluctuate to reach the target conformational search ability. When *isgld=3* (SGLDg), the default is *tempsg=temp*0, which allows the canonical ensemble will be sampled. When *tempsg*<>*temp0,* the reweighting factor can be found in the SGLD output lines described below.

`isgsta` The first atom index of SGLD region. Default is 1.

`isgend` The last atom index of SGLD region. Default is *natom*.

`fixcom` Option to remove the net translation of the center of mass. For finite systems it is often more convenient to have the center of mass fixed. Default 0 *when gamma_ln>0 or 1 when gamma_ln=0.* When *fixcom>0,* the center of mass is fixed.

`treflf` Reference low frequency temperature. Default 0.0. *treflf* is the low frequency temperature when no guiding force is applied to a simulation system (*sgft*=0). *treflf* is required for the weighting factor calculation in SGMD/SGLD simulation (*isgld*=1) or for the guiding force calculation in SGMDfp/SGLDfp simulations (*isgld*=2). *treflf* is not needed in SGLDg or SGMDg (*isgld*=3) simulations. When *treflf*=0, *treflf* will be estimated during a simulation. An accurate value of *treflf* can increase the accuracy in the weighting factor calculation in SGMDfp/SGLDfp simulations.

`sgfd` Optional high frequency force guiding factor for SGLDfp (*isgld*=2). Should not be set if conformational distribution in high resolution need be maintained. When not set, its instantaneous value (printed out in simulation output files) fluctuates to maintain high resolution conformational distribution.

The output of SGMD/SGLD simulations contains the following properties related to the enhancement in conformational search and reweighting of conformational distribution:

SGLF = SGFT TEMPSG TEMPLF TREFLF FRCLF EPOTLF SGWT
SGHF = SGFF SGFD TEMPHF TREFHF FRCHF EPOTHF VIRSG

These quantities are instantaneous values defined as below:

SGFT: Momentum guiding factor,
SGFF: Force guiding factor. Adjusted in SGLDfp simulations (*isgld*=2)
SGFD: Force dumping factor. Adjusted in SGLDfp simulations (*isgld*=2)
TEMPSG: Guiding temperature.
SGWT: Weighting free energy. exp(SGWT) is the weighting factor of current frame.
VIRSG: Virial of the guiding force.
TEMPLF: low frequency temperature
TEMPHF: high frequency temperature
TREFLF: reference low frequency temperature. It is the TEMPLF at SGFT=0 and TEMPSG=0.
TREFHF: reference high frequency temperature. It is the TEMPHF at SGFT=0 and TEMPSG=0.
FRCLF: low frequency force factor
FRCHF: high frequency force factor
EPOTLF: low frequency potential energy
EPOTHF: high frequency potential energy

The weight of a conformation is calculated by

```
Weight=exp(SGWT) =exp(((FRCLF*TREFLF/TEMPLF-1)*EPOTLF+
(FRCHF*TREFHF/TEMPHF-1)*EPOTHF+VIRSG)/(KBOLTZ*Temp))
```

or:

$$w_i = \exp\left((\lambda_{LF}\frac{T^0_{LF}}{T_{LF}} - 1)\frac{E_{LF}}{kT} + (\lambda_{HF}\frac{T^0_{HF}}{T_{HF}} - 1)\frac{E_{HF}}{kT} + \frac{W_{sg}}{kT}\right)$$

For convenience, two scripts, *sgldinfo.sh* and *sgldwt.sh*, are provided in the AMBERHOME/bin directory to extract SGLD properties and weighting factors from sander output files. For example, one can run:

```
sgldinfo.sh mdout
```

to examine the SGLD properties, and run:

```
sgldwt.sh mdout
```

to print weighting factors at each print time frame. One may specify TREFLF, e.g., 23.5 K, and/or TREFHF, e.g., 278.2 K, for more accurate weighting factors:

```
sgldwt.sh mdout 23.5 278.2
```

TREFLF and TREFHF can be obtained with *sgldinfo.sh* from a SGLD simulation at the same condition except SGFT=0 and TEMPSG=0. Without specifying TREFLF and TREFHF, they will be estimated for the calculation. Ensemble average properties are calculated through reweighting:

$$< P >= \frac{\sum_{i=N_0}^{N} w_i P_i}{\sum_{i=N_0}^{N} w_i}$$

For SGLDfp (*isgld*=2) and SGLDg (*isgld*=3) simulations, no reweighting is needed.

Here is an example of a SGLD simulation input file:

```
Sample SGLD simulation to reach a 500K conformational search ability
&cntrl
nstlim=1000, cut=99.0, igb=1, saltcon=0.1,
ntpr=100, ntwr=100000, ntt=3, gamma_ln=10.0,
ntx=5, irest=1, ig = 256251,
ntc=2, ntf=2, tol=0.000001,
dt=0.002, ntb=0, tempi=300., temp0=300.,
isgld=1,tsgavg=0.2,tempsg=500,
/
```

Below is an example of a SGLDg simulation input file to sample the canonical ensemble while accelerate conformational search:

```
Sample SGLDg simulation for efficient conformational sampling
&cntrl
nstlim=1000, cut=99.0, igb=1, saltcon=0.1,
ntpr=100, ntwr=100000, ntt=3, gamma_ln=10.0,
ntx=5, irest=1, ig = 256251,
ntc=2, ntf=2, tol=0.000001,
dt=0.002, ntb=0, tempi=300., temp0=300.,
isgld=3,tsgavg=0.2,tempsg=300,sgft=0.5,sgff=-0.1,
/
```

Below is an example of a SGMDg simulation input file for accelerated conformational search:

```
Sample SGLDg simulation for efficient conformational sampling
&cntrl
nstlim=1000, cut=99.0, igb=1, saltcon=0.1,
ntpr=100, ntwr=100000, ntt=1, gamma_ln=0.0,
ntx=5, irest=1, ig = 256251,
ntc=2, ntf=2, tol=0.000001,
dt=0.002, ntb=0, tempi=300., temp0=300.,
isgld=3,tsgavg=0.2,tempsg=500,sgft=1,sgff=-0.1,
/
```

10.2. Accelerated Molecular Dynamics

10.2.1. Introduction

Many systems of interest in chemistry, physics and biology are characterized by the presence of a number of metastable states separated by large barriers. Correctly sampling these systems is challenging for methods based on Molecular Dynamics, Monte Carlo sampling or any other type of dynamic simulation. For most biological systems of interest, the simulation time is limited to the nanosecond-microsecond time scale, so simple molecular dynamics cannot be used to adequately explore portions of the energy landscape separated by high barriers from the initial minimum. Furthermore, for most biological molecules, the energy landscape has multiple minima or potential energy wells with high free energy barriers, and during a molecular dynamics simulation the system is trapped in one or another local minimum for long periods of simulation time. Consequently, thermodynamics and many other properties of interest for large biological systems cannot be simulated directly because of the nonergodic nature of the present state of the molecular dynamics methodology for systems with high free energy barriers.

Accelerated Molecular Dynamics (aMD) is a bias potential introduced by the McCammon group at UCSD [191]. It is a modification to the potential that in practice reduces the height of local barriers, allowing the calculation to evolve much faster. A number of methods have been suggested to aid this problem, like replica exchange, metadynamics, etc. AMD represents an interesting option as it only requires the evolution of a single copy of the system, plus it doesn't require any previous knowledge of the shape of the potential, i.e. aMD doesn't require information of where are the barriers, saddle points or even what type of configuration changes are expected or necessary to traverse through a particular barrier. Moreover, an interesting feature of aMD is that the shape of the added potential conserves the underlying shape of the real one, such that minima are maintained as minima and barriers are preserved as barriers. In result, adding the aMD potential in practice simply modifies the relation between energy differences, so the distribution of sampling of different structures is still related to the original potential distribution and can be recovered exactly by reweighing.

The aMD modification of the potential is defined by the following equation:

$$\mathrm{V(r)}* = V(r) + \Delta V(r) \tag{10.1}$$

$$\Delta V(r) = \frac{(Ep - V(r))^2}{(\alpha P + Ep - V(r))} + \frac{(Ed - Vd(r))^2}{(\alpha D + Ed - Vd(r))} \tag{10.2}$$

where $V(r)$ is the normal potential and $Vd(r)$ is the normal torsion potential. Ep and Ed are average potential and dihedral energies that serve as a reference energy from which to compare the present position of the calculation and therefore the relationship to the boosting factor to be applied. The terms αP and αD are factors that determine inversely the strength with which the boost is applied. For large values of alpha, the potential felt at any point will essentially be the same as the true potential For values of alpha close to zero, the potential felt becomes constant, in this limit, the sampling becomes a random walk. The amount of boost felt at a particular point in the calculation, therefore, depends on the present value of the potential and dihedral energy, which is in direct correlation to how low in the energy surface the configuration is positioned at that moment. The boosting potential will be proportionally bigger for deeper regions of the potential energy surface, while it will be smaller for higher points, which in result conserves the underlying shape of the potential, as previously mentioned.

AMD has been applied to a vast diversity of interesting problems [192–196]. We have recently applied the implementation of aMD in Amber to the Bovine Pancreatic Trypsin Inhibitor and compared with an unbiased millisecond MD simulation, showing aMD is able to recover the right population distribution and shows excellent agreement with the MD simulation as with experimental data [195].

10.2.2. AMD implementation in Amber

AMD has been implemented in both *sander* and *pmemd* by Romelia Salomon-Ferrer. The implementation includes the possibility of boosting independently only the torsional terms of the potential (*iamd*=2) or the whole potential at once (*iamd*=1). It also allows the possibility to boost the whole potential with an extra boost to the torsions(*iamd*=3). All the information generated by aMD, necessary for reweighing is stored at each step into a vector which is flushed to a log file (*amd.log* by default) every time the coordinates are written to disk, i.e. every *ntwx* steps. This is done for performance reasons, since writing to disk is always time consuming and it is not advisable to do it every step. The name of the log file can be set to a user defined name by using the command line option -amdlog when running Amber. Our present implementation also allows the user to delay (or lag) the boosting a number of steps, i.e. only boost with a particular frequency defined by the variable *amdlag*. Additional parameters are specified by the following variables: *EthreshD* (Ed), *alphaD* (αD), *EthreshP* (Ep) and *alphaP* (αP).

AMD output is saved the *amd.log* this file contains all the information needed for reweighting the results obtained to recover the unperturbed distributions. The *amd.log* file gets written with the same frequency at which the configurations are saved to disk in the trajectory file (*mdcrd*). Each line corresponds to the information of a corresponding snapshot being saved on the mdcrd file. Regardless of what iamd value is used, the number of columns in the amd.log file are always the same, they just have 0 or 1 (correspondingly) if no boost is being added to dihedral or total energy

The amd.log file has the following header:

```
#All energy terms stored in units of kcal/mol
#ntwx,total_nstep,Unboosted-Potential-Energy,Unboosted-Dihedral-Energy,Total-Force-W
Dihedral-Force-Weight,Boost-Energy-Potential,Boost-Energy-Dihedral
```

The description for the main columns is as follows:

- **Unboosted-Potential-Energy:** Total Potential Energy without boost added, kcal/mol.
- **Unboosted-Dihedral-Energy:** dihedral energy without boost added, kcal/mol**.**
- **Total-Force-Weight:** The force scaling factor calculated from the boost to the Total Potential Energy
- **Dihedral-Force-Weight:** The dihedral force scaling factor from dihedral boost
- **Boost-Energy-Potential:** The boost energy in kcal/mol
- **Boost-Energy-Dihedral:** The dihedral boost energy in kcal/mol

IMPORTANT NOTE: Before Amber 14 the boost energy for the dihedral and total potential energy (last two columns) was given in units of *kT*. This decision was made at the beginning with the idea that the user could read and use these values directly for reweighting without any further work, but later it we decided it was much better and more consistent to have all energy output in kcal/mol as the rest of AMBER's energy output. As of AMBER 14, the last two columns of the amd.log file as given in units of kcal/mol.

Reweighting

For reweighting aMD results we would like to add the link to a great tutorial (http://mccammon.ucsd.edu/computing/amdReweighting/) which also provides a small python script to perfom the reweighting. The script is compatible with the newer versions of AMBER, and can be used to reweight 1D and 2D distributions. A simple C code is also provided in the aMD tutorial that performs reweighting based on the Kernel Density Estimation algorithm. This algorithm also performs very well and reduces the amount of noise without using a truncated expression for the exponential and can be used as an alternative for reweighting. To extract the energies from the amd.log file into a file, weights.dat, to use with this script, something like the following could be done:

```
# Column 1: dV in units of kbT; column 2: timestep; column 3: dV in units of kcal/mol
# For AMBER14: # awk 'NR%1==0' amd.log | awk '{print ($8+$7)/(0.001987*300)
    " " $2 " " ($8+$7)}' > weights.dat
# For AMBER12: # awk 'NR%1==0' amd.log | awk '{print ($8+$7)" " $3
    " " ($8+$7)*(0.001987*300)}' > weights.dat
```

For reweighting a 2D distribution, for instance a Phi Psi distribution, you would need to extract the values for Phi and Psi for each frame in the mdcrd file using AmberTools and generate the file Phi_Psi file and then use the python tool provided in the website to get the reweighted surface.

```
python PyReweighting-2D.py -input Phi_Psi -Emax 100 -discX 6 -discY 6

    -job amdweight_MC -order 10 -weight weights.dat | tee -a reweight_variable.log
```

For reweighting using a Maclaurin series expansion as an approximation for the exponential weight.

10.2.3. Preparing a system for aMD

As mentioned before, running aMD requires the definition of few parameters. AMD parameters are determined based on previous knowledge of the system, which is easily acquirable by a short regular MD simulation, from which the average values of the potential and torsion energy can be estimated. From there, a given amount of energy per degree of freedom is added those values, in the form of multiples of alpha, setting the values of Ep and Ed to be used. The following example should help clarify this procedure.

```
Average Dihedral : 611.5376 (based on MD simulations)
Average EPtot : -53155.3104 (based on MD simulations)
total ATOMS=16950
protein residues=64
```

For the dihedral potential:

```
Approximate energy contribution per degree of freedom.
3.5*64= 224          The value of 3.5 kcal/mol/residue seems to work well
alphaD = (1/5)*224 = 45 The value of .2 seems to work well
```

```
EthreshD = 224+611 = 835
For the total potential
alphaP = 16950*(1/5)=3390
For a lower boost you can also use a value between 0.15-0.19 instead of 0.20 (0.16 w
EthreshP = -53155.3104 + 3390 = -49765.3104
With these parameters, the aMD parameters in the input file should then be set to
iamd=3,EthreshD=835,alphaD=45,EthreshP=-49765,alphaP=3390,
For a higher acceleration it is common to simply add to Eb(dih) multiples of alpha.
iamd=3,EthreshD=880,alphaD=45,EthreshP=-49765,alphaP=3390,
Two levels higher would be then defined by:
iamd=3,EthreshD=925,alphaD=45,EthreshP=-49765,
```

After the aMD parameters to be used are defined, an MD run with aMD can be set using those parameters. Depending on the progress of the simulation, a higher boost can be applied as specified in the above example.

10.2.4. Sample input file for aMD

An example of an input file would be the following:

```
AVP dt=2.0fs with SHAKE, NPT aMD boost pot and dih
 &cntrl
  imin=0, irest=1, ntx=5,
  dt=0.002, ntc=2, ntf=2, tol=0.000001,iwrap=1,
  ntb=2, cut=12.0, ntp=1,igb=0,ntwprt = 3381,ioutfm = 1,
  ntt=3, temp0=310.0, gamma_ln=1.0, ig=-1,
  ntpr=1000, ntwx=1000, ntwr=2000000, nstlim=2000000,
  iamd=3,EthreshD=835,
  alphaD=45,EthreshP=-49765,
  alphaP=3390,
 /
&ewald
dsum_tol=0.000001,
 /
```

10.2.5. Further information

Test cases have been included into the distribution of Amber, also a tutorial based on a study we performed on BPTI [195], showing the power of aMD and its validation versus a millisecond run on the same system performed on Anton is now present on the Amber website. We encourage the user to read the paper, as well as follow the tutorial for more information.

10.3. Gaussian Accelerated Molecular Dynamics

10.3.1. Introduction

Gaussian Accelerated Molecular Dynamics (GaMD) is a biomolecular enhanced sampling method that works by adding a harmonic boost potential to smooth the system potential energy surface. The boost potential follows Gaussian distribution, which allows for accurate reweighting using cumulant expansion to the second order. GaMD has been demonstrated on simulations of alanine dipeptide, chignolin folding and ligand binding to the T4-lysozyme [197]. GaMD enables unconstrained enhanced sampling of these biomolecules without the need to set predefined reaction coordinates. Furthermore, the free energy profiles obtained from reweighting of the GaMD simulations help identify distinct low energy states of the biomolecules and characterize the protein folding and ligand binding pathways quantitatively.

The basic theory of GaMD can be found in References [197, 198], or at http://miao.compbio.ku.edu/GaMD.

10.3.2. Ligand Gaussian Accelerated Molecular Dynamics (LiGaMD)

A new algorithm called ligand GaMD or "LiGaMD" has been developed to simulate ligand binding and unbinding[199]. It works by selectively boosting the ligand non-bonded interaction potential energy. Another boost potential could be applied to the remaining potential energy of the entire system in a dual-boost algorithm (LiGaMD_Dual) to facilitate ligand binding. LiGaMD has been demonstrated on host-guest and protein-ligand binding model systems. Repetitive guest binding and unbinding in the β-cyclodextrin host were observed in hundreds-of-nanosecond LiGaMD simulations. The calculated binding free energies of guest molecules with sufficient sampling agreed excellently with experimental data (< 1.0 kcal/mol error). In comparison with previous microsecond-timescale conventional molecular dynamics simulations, accelerations of ligand kinetic rate constants in LiGaMD simulations were properly estimated using Kramers' rate theory. Furthermore, LiGaMD allowed us to capture repetitive dissociation and binding of the benzamidine inhibitor in trypsin within 1 μs simulations. The calculated ligand binding free energy and kinetic rate constants compared well with the experimental data. Therefore, LiGaMD provides a promising approach for characterizing ligand binding thermodynamics and kinetics simultaneously.

Next, one can add multiple ligand molecules in the solvent to facilitate ligand binding to proteins in MD simulations. This is based on the fact that the ligand binding rate constant kon is inversely proportional to the ligand concentration. The higher the ligand concentration, the faster the ligand binds, provided that the ligand concentration is still within its solubility limit. In addition to selectively boosting the bound ligand, another boost potential could thus be applied on the unbound ligand molecules, protein and solvent to facilitate both ligand dissociation and rebinding.

10.3.3. Peptide Gaussian Accelerated Molecular Dynamics (Pep-GaMD)

Peptides often undergo large conformational changes during binding to the target proteins, being distinct from small-molecule ligand binding or protein-protein interactions. We have developed another algorithm called peptide GaMD or "Pep-GaMD" that enhances sampling of peptide-protein interactions (manuscript in preparation). See http://miao.compbio.ku.edu/GaMD for more information.

10.3.4. Implementations of GaMD, LiGaMD and Pep-GaMD algorithms in Amber

GaMD has been implemented in pmemd, both the serial and parallel versions on CPU (*pmemd* and *pmemd.MPI*) and GPU (*pmemd.cuda* and *pmemd.cuda.MPI*) by Yinglong Miao. Note that GaMD is not available in Sander. Similar to aMD, GaMD provides options about what energies to boost (see the *igamd* variable.) The dual-boost simulation generally provides higher acceleration than the other single-boost simulations for enhanced sampling.

LiGaMD has been implemented by Yinglong Miao in only the serial GPU version of pmemd (*pmemd.cuda*). It provides options to boost only non-bonded potential energy of the bound ligand (LiGaMD, *igamd=10*) and in addition the total system potential energy other than the non-bonded potential energy of bound ligand (LiGaMD_Dual, *igamd=11*). LiGaMD_Dual generally provides higher acceleration than LiGaMD for enhanced sampling. The simulation parameters comprise of settings for calculating the threshold energy values and the effective harmonic force constants of the boost potentials.

Pep-GaMD has been implemented by Jinan Wang in only the serial GPU version of pmemd (*pmemd.cuda*). It provides options to boost only the peptide potential energy (Pep-GaMD, *igamd=14*) and in addition the total system potential energy other than the peptide potential energy (Pep-GaMD_Dual, *igamd=15*). Pep-GaMD_Dual generally provides higher acceleration than Pep-GaMD for enhanced sampling. The simulation parameters comprise of settings for calculating the threshold energy values and the effective harmonic force constants of the boost potentials.

All the information generated by GaMD, necessary for reweighing is stored at each step into a vector which is flushed to a log file (*gamd.log* by default) every time the coordinates are written to disk, i.e. every *ntwx* steps. The name of the log file can be set to a user defined name by using the command line option *-gamd* when running Amber. Additional parameters are specified by the following variables:

`igamd` Flag to apply boost potential

= 0 (default) no boost is applied

= 1 boost on the total potential energy only

= 2 boost on the dihedral energy only

= 3 dual boost on both dihedral and total potential energy

=4 boost on the non-bonded potential energy only

=5 dual boost on both dihedral and non-bonded potential energy

=10 boost on non-bonded potential energy of selected region (defined by timask1 and scmask1) as for a ligand (LiGaMD)

=11 dual boost on both non-bonded potential energy of the bound ligand and remaining potential energy of the rest of the system (LiGaMD_Dual)

=14 boost on the total potential energy of selected region (defined by timask1 and scmask1) as for a peptide (Pep-GaMD)

=15 dual boost on both the peptide potential energy and the total system potential energy other than the peptide potential energy (Pep-GaMD_Dual)

`iE` Flag to set the threshold energy E

= 1 (default) set the threshold energy to the lower bound $E = V_{max}$

= 2 set the threshold energy to the upper bound $E = V_{min} + (V_{max} - V_{min})/k_0$

`iEP` Flag to overwrite iE and set the threshold energy E for applying the first boost potential in dual-boost schemes

=1 (default) set the threshold energy to the lower bound E = Vmax

=2 set the threshold energy to the upper bound E = Vmin + (Vmax - Vmin)/k0

`iED` Flag to overwrite iE and set the threshold energy E for applying the second boost potential in dual-boost schemes

= 1 (default) set the threshold energy to the lower bound $E = V_{max}$

= 2 set the threshold energy to the upper bound $E = V_{min} + (V_{max} - V_{min})/k_0$

`ntcmdprep` The number of preparation conventional molecular dynamics steps. This is used for system equilibration and the potential energies are not collected for calculating their statistics. The default is 200,000 for a simulation with 2 fs timestep.

`ntcmd` The number of initial conventional molecular dynamics simulation steps. Potential energies are collected between *ntcmdprep* and *ntcmd* to calculate their maximum, minimum, average and standard deviation ($V_{max}, V_{min}, V_{avg}, \sigma_V$). The default is 1,000,000 for a simulation with 2 fs timestep.

`ntebprep` The number of preparation biasing molecular dynamics simulation steps. This is used for system equilibration after adding the boost potential and the potential statistics ($V_{max}, V_{min}, V_{avg}, \sigma_V$) are not updated during these steps. The default is 200,000 for a simulation with 2 fs timestep.

`nteb` The number of biasing molecular dynamics simulation steps. Potential statistics (Vmax, Vmin, Vavg, σV) are updated between the ntebprep and nteb steps and used to calculate the GaMD acceleration parameters, particularly E and k0. The default is 1,000,000 for a simulation with 2 fs timestep. A greater value may be needed to ensure that the potential statistics and GaMD acceleration parameters level off before running production simulation between the *nteb* and *nstlim* (total simulation length) steps. Moreover, *nstlim* can be set to *ntcmd+nteb*, by which the potential statistics and GaMD acceleration parameters are updated adaptively throughout the simulation. This in some cases provides more appropriate acceleration.

`ntave` The number of simulation steps used to calculate the average and standard deviation of potential energies. This variable has already been used in Amber. The default is set to 50,000 for GaMD simulations. It is recommended to be updated as about 4 times of the total number of atoms in the system. Note that ntcmdprep, ntcmd, ntebprep and nteb need to be multiples of ntave.

`irest_gamd` Flag to restart GaMD simulation

= 0 (default) new simulation. A file "gamd-restart.dat" that stores the maximum, minimum, average and standard deviation of the potential energies neeed to calculate the boost potentials (depending on the *igamd* flag) will be saved automatically after GaMD equilibration stage.

= 1 restart simulation (*ntcmd* and *nteb* are set to 0 in this case). The "gamd-restart.dat" file will be read for restart.

`sigma0P` The upper limit of the standard deviation of the first potential boost that allows for accurate reweighting. The default is 6.0 (unit: kcal/mol).

`sigma0D` The upper limit of the standard deviation of the second potential boost that allows for accurate reweighting in dual-boost simulations (e.g., igamd = 2, 3, 5, 11 and 15). The default is 6.0 (unit: kcal/mol).

`timask1` Specifies atoms of the bound ligand or peptide in ambmask format. The default is an empty string.

`scmask1` Specifies atoms of the bound ligand that will be described using soft core in ambmask format in LiGaMD. In Pep-GaMD, this flag was only used to specify atoms of peptide in ambmask format, but the peptide atoms will be not described using soft core. The default is an empty string.

`nlig` The total number of ligand molecules in the system. The default is 0.

`ibblig` The flag to boost the bound ligand selectively with nlig > 1

=0 (default) no selective boost

=1 boost the bound ligand selectively out of nlig ligand molecules in the system

`atom_p` Serial number of a protein atom (starting from 1 for the first protein atom) used to calculate the ligand distance. It is used only when ibblig = 1. The default is 0.

`atom_l` Serial number of a ligand atom (starting from 1 for the first ligand atom) used to calculate the ligand distance to the protein. It is used only when ibblig = 1. The default is 0.

`dblig` The cutoff distance between atoms atom_p and atom_l for determining whether the ligand is bound in the protein. It is used only when ibblig = 1. The default is 4.0 Å.

10.3.5. Algorithms used

The GaMD algorithm is summarized as follows:

```
GaMD {
  If (irest_gamd == 0) then
    For i = 1, ..., ntcmd // run initial conventional molecular dynamics
          If (i >= ntcmdprep) Update Vmax, Vmin
          If (i >= ntcmdprep && i%ntave ==0) Update Vavg, sigmaV
    End
    Save Vmax,Vmin,Vavg,sigmaV to gamd_restart.dat file
    Calc_E_k0(iE,sigma0,Vmax,Vmin,Vavg,sigmaV)

    For i = ntcmd+1, ..., ntcmd+nteb // Run biasing molecular dynamics
                                     // simulation steps
          deltaV = 0.5*k0*(E-V)**2/(Vmax-Vmin)
```

```
          V = V + deltaV
          If (i >= ntcmd+ntebprep) Update Vmax, Vmin
          If (i >= ntcmd+ntebprep && i%ntave ==0) Update Vavg, sigmaV
      Calc_E_k0(iE,sigma0,Vmax,Vmin,Vavg,sigmaV)
    End
    Save Vmax,Vmin,Vavg,sigmaV to gamd_restart.dat file
  else if (irest_gamd == 1) then
    Read Vmax,Vmin,Vavg,sigmaV from gamd_restart.dat file
  End if

  For i = ntcmd+nteb+1, ..., nstlim // run production simulation
    deltaV = 0.5*k0*(E-V)**2/(Vmax-Vmin)
    V = V + deltaV
  End
}

Subroutine Calc_E_k0(iE,sigma0,Vmax,Vmin,Vavg,sigmaV) {
  if iE = 1 :
        E = Vmax
        k0' = (sigma0/sigmaV) * (Vmax-Vmin)/(Vmax-Vavg)
        k0 = min(1.0, k0')
  else if iE = 2 :
        k0'' = (1-sigma0/sigmaV) * (Vmax-Vmin)/(Vavg-Vmin)
        if 0 < k0'' <= 1 :
               k0 = k0''
               E = Vmin + (Vmax-Vmin)/k0
        else
               E = Vmax
               k0' = (sigma0/sigmaV) * (Vmax-Vmin)/(Vmax-Vavg)
               k0 = min(1.0, k0')
        end
  end
}
```

The LiGaMD algorithm is summarized as the following:

```
LiGaMD {
     If (irest_gamd == 0) then
        For i = 1, ..., ntcmd // run initial conventional molecular dynamics
           If (i >= ntcmdprep) Update Vmax, Vmin
           If (i >= ntcmdprep && i%ntave ==0) Update Vavg, sigmaV
        End
       Save Vmax,Vmin,Vavg,sigmaV to gamd_restart.dat file
       Calc_E_k0(iE,sigma0,Vmax,Vmin,Vavg,sigmaV)
       For i = ntcmd+1, ..., ntcmd+nteb // Run biasing molecular dynamics simulation steps
             deltaV = 0.5*k0*(E-V)**2/(Vmax-Vmin)
             V = V + deltaV
             If (i >= ntcmd+ntebprep) Update Vmax, Vmin
             If (i >= ntcmd+ntebprep && i%ntave ==0) Update Vavg, sigmaV
             Calc_E_k0(iE,sigma0,Vmax,Vmin,Vavg,sigmaV)
       End
       Save Vmax,Vmin,Vavg,sigmaV to gamd_restart.dat file
     else if (irest_gamd == 1) then
       Read Vmax,Vmin,Vavg, sigmaV from gamd_restart.dat file
     End if

     lig0=1 // ID of the bound ligand
```

```
     For i = ntcmd+nteb+1, ..., nstlim // run production simulation
       If (ibblig>0 && i%ntave ==0) then // swap the bound ligand with lig0 for selective boost
         For ilig = 1, ..., nlig
            dlig = distance(atom_p, atom_l)
            If (dlig <= dblig) blig=ilig
         End
         If (blig != lig0) Swap atomic coordinates, forces and velocities of ligands blig with lig0
       End if

       deltaV = 0.5*k0*(E-V)**2/(Vmax-Vmin)
       V = V + deltaV
     End
}
```

The Pep-GaMD algorithm is summarized as the following:

```
Pep-GaMD {
     If (irest_gamd == 0) then
        For i = 1, ..., ntcmd // run initial conventional molecular dynamics
           If (i >= ntcmdprep) Update Vmax, Vmin
           If (i >= ntcmdprep && i%ntave ==0) Update Vavg, sigmaV
        End
       Save Vmax,Vmin,Vavg,sigmaV to gamd_restart.dat file
       Calc_E_k0(iE,sigma0,Vmax,Vmin,Vavg,sigmaV)
       For i = ntcmd+1, ..., ntcmd+nteb // Run biasing molecular dynamics simulation steps
             deltaV = 0.5*k0*(E-V)**2/(Vmax-Vmin)
             V = V + deltaV
             If (i >= ntcmd+ntebprep) Update Vmax, Vmin
             If (i >= ntcmd+ntebprep && i%ntave ==0) Update Vavg, sigmaV
             Calc_E_k0(iE,sigma0,Vmax,Vmin,Vavg,sigmaV)
       End
       Save Vmax,Vmin,Vavg,sigmaV to gamd_restart.dat file
     else if (irest_gamd == 1) then
       Read Vmax,Vmin,Vavg, sigmaV from gamd_restart.dat file
     End if


     For i = ntcmd+nteb+1, ..., nstlim // run production simulation
       deltaV = 0.5*k0*(E-V)**2/(Vmax-Vmin)
       V = V + deltaV
     End
}
```

10.3.6. Sample input files

Here is a sample input file for GaMD:

```
&cntrl
  imin = 0, irest = 0, ntx = 1,
  nstlim = 17000000, dt = 0.002,
  ntc = 2, ntf = 2, tol = 0.000001,
  iwrap = 1, ntb = 1, cut = 8.0,
  ntt = 3, temp0 = 300.0, tempi = 300.0,
  ntpr = 50, ntwx = 50, ntwr = 500,
  ntxo = 1, ioutfm = 1, ig = -1, ntwprt = 22,

  igamd = 3, iE = 1, irest_gamd = 0,
```

```
  ntcmd = 1000000, nteb = 1000000, ntave = 50000,
  ntcmdprep = 200000, ntebprep = 200000,
  sigma0P = 6.0, sigma0D = 6.0,
&end
```

Add the following for LiGaMD_Dual simulations:

```
   igamd = 11, irest_gamd = 0,
   ntcmd = 700000, nteb = 27300000, ntave = 140000,
   ntcmdprep = 280000, ntebprep = 280000,
   sigma0P = 4.0, sigma0D = 6.0, iEP = 2, iED=1,

   icfe = 1, ifsc = 1, gti_cpu_output = 0, gti_add_sc = 1,
   timask1 = ':225', scmask1 = ':225',
   timask2 = '', scmask2 = '',

   ibblig = 1, nlig = 10, atom_p = 2472, atom_l = 4, dblig = 3.7
```

Add the following parameters for Pep-GaMD simulations:

```
  icfe = 1, ifsc = 1,gti_cpu_output = 0,gti_add_sc = 1,
  timask1 = ':1-3', scmask1 = ':1-3',
  timask2 = '', scmask2 = '',

  igamd = 15, iE = 1, iEP = 1, iED = 1, irest_gamd = 0,
  ntcmd = 1000000, nteb = 1000000, ntave = 50000,
  ntcmdprep = 200000, ntebprep = 200000,
  sigma0P = 6.0, sigma0D = 6.0,
```

10.3.7. Further information

Reweighting analysis of GaMD simulations is similar to that of previous aMD simulations, except that the *amd.log* file name needs to be replaced by *gamd.log*. Test cases have been included into the distribution of Amber, also a tutorial based on a study we performed on alanine dipeptide [197], demonstrating the usage of GaMD on unconstrained enhanced sampling and free energy calculation of biomolecules is available at http://miao.compbio.ku.edu/GaMD. We encourage the user to read the paper, as well as follow the tutorial for more information. The PyReweighting scripts are available at: http://miao.compbio.ku.edu/PyReweighting/.

10.4. Targeted MD

The targeted MD option adds an additional term to the energy function based on the mass-weighted root mean square deviation of a set of atoms in the current structure compared to a reference structure. The reference structure is specified using the *-ref* flag in the same manner as is used for Cartesian coordinate restraints (NTR=1). Targeted MD can be used with or without positional restraints. If positional restraints are not applied (ntr=0), *sander* performs a best-fit of the reference structure to the simulation structure based on selection in *tgtfitmask* and calculates the RMSD for the atoms selected by *tgtrmsmask*. The two masks can be identical or different. This way, fitting to one part of the structure but calculating the RMSD (and thus restraint force) for another part of the structure is possible. If targeted MD is used in conjunction with positional restraints (ntr=1), only *tgtrmsmask* should be given in the control input because the molecule is 'fitted' implicitly by applying positional restraints to atoms specified in *restraintmask*.
The energy term has the form:

```
E = 0.5 * TGTMDFRC * NATTGTRMS * (RMSD-TGTRMSD)**2
```

The energy will be added to the RESTRAINT term. Note that the energy is weighted by the number of atoms that were specified in the *tgtrmsmask* (NATTGTRMS). The RMSD is the root mean square deviation and is mass weighted. The force constant is defined using the *tgtmdfrc* variable (see below). This option can be used with molecular dynamics or minimization. When targeted MD is used, *sander* will print the current values for the actual and target RMSD to the energy summary in the output file.

`itgtmd` **= 0** no targeted MD (default)

= 1 use targeted MD

= 2 use targeted MD to multiple targets (Multiply-targeted MD, or MTMD, see next section below)

`tgtrmsd` Value of the target RMSD. The default value is 0. This value can be changed during the simulation by using the weight change option.

`tgtmdfrc` This is the force constant for targeted MD. The default value is 0, which will result in no penalty for structure deviations regardless of the RMSD value. Note that this value can be negative, which would force the coordinates AWAY from the reference structure.

`tgtfitmask` Define the atoms that will be used for the rms superposition between the current structure and the reference structure. Syntax is in Chapter 9.1.1.

`tgtrmsmask` Define the atoms that will be used for the rms difference calculation (and hence the restraint force), as outlined above. Syntax is in Chapter 9.1.1.

One can imagine many uses for this option, but a few things should be kept in mind. In this implementation of targeted MD, there is currently only one reference coordinate set, so there is no way to force the coordinates to any specific structure other than the one reference. To move a structure toward a reference coordinate set, one might use an initial *tgtrmsd* value corresponding to the actual RMSD between the input and reference (*inpcrd* and *refc*). Then the weight change option could be used to decrease this value to 0 during the simulation. To move a structure away from the reference, one can increase *tgtrmsd* to values larger than zero. The minimum for this energy term will then be at structures with an RMSD value that matches *tgtrmsd*. Keep in mind that many different structures may have similar RMSD values to the reference, and therefore one cannot be sure that increasing *tgtrmsd* to a given value will result in a particular structure that has that RMSD value. In this case it is probably wiser to use the final structure, rather than the initial structure, as the reference coordinate set, and decrease *tgtrmsd* during the simulation. To address this, multiply-targeted MD is now available in Amber (*sander only*), and is described in the next section. As an additional note, a negative force constant *tgtmdfrc* can be used, but this can cause problems since the energy will continue to decrease as the RMSD to the reference increases.

Also keep in mind that phase space for molecular systems can be quite complex, and this method does not guarantee that a low energy path between initial and target structures will be followed. It is possible for the simulation to become unstable if the restraint energies become too large if a low-energy path between a simulated structure and the reference is not accessible.

Note also that the input and reference coordinates are expected to match the *prmtop* file and have atoms in the same sequence. No provision is made for symmetry; rotation of a methyl group by 120° would result in a nonzero RMSD value.

10.5. Multiply-Targeted MD (MTMD)

In Amber (*sander only*), the user may perform targeted MD calculations using multiple reference structures. Each reference may have its own associated target RMSD value and force constant, each of which can evolve independently in time. Additionally, the masks for each defined target may differ, and targeting to any given reference structure can be activated for some or part of the simulation. The energy term for MTMD is simply the sum of the energies that would be calculated for the molecule calculated relative to each target given the target RMSD and force constant for that target. The energy will then be added to the RESTRAINT term.

To use MTMD, the MTMD input file is specified using the *-mtmd* flag in the command line arguments for *sander*. The MTMD input file will contain one instance of the tgt namelist ("&tgt") for each reference structure used. The user may specify any number of reference structures.

10.5.1. Variables in the &tgt namelist:

`refin` The file name of the reference structure used. The input and reference coordinates are expected to match the *prmtop* file and have atoms in the same sequence. *Default for refin is '', no reference structure given.*

`mtmdform` If MTMDFORM > 0, then the reference coordinate file is formatted. Otherwise, the reference coordinate file is an unformatted (binary) file. *Default for MTMDFORM is the value assigned to MTMDFORM in the most recent namelist where MTMDFORM was specified. If MTMDFORM has not been specified in any namelist, it defaults to 1.*

`mtmdstep1, mtmdstep2` Targeted MD for this structure is run for steps/iterations MTMDSTEP1 through MTMDSTEP2. If MTMDSTEP2 = 0, then TMD will be run through the end of the run, and the values of the target RMSD and the force constant will not change with time. Note that the first step/iteration is considered step 0. *Defaults for MTMDSTEP1 and MTMDSTEP2 are the values assigned to them in the most recent namelist where MTMDSTEP1 and MTMDSTEP2 were specified. If MTMDSTEP1 and MTMDSTEP2 have not been specified in any namelist, they default to 0.*

`mtmdvari` If MTMDVARI > 0, then the force constant and target RMSD will vary with step number. Otherwise, they are constant throughout the run. If MTMDVARI >0, then the values MTMDSTEP2, MTMDRMSD2, and MTMDFORCE2 must be specified (see below). *Default for* MTMDVARI *is the value assigned to* MTMDVARI *in the most recent namelist where* MTMDVARI *was specified. If* MTMDVARI *has not been specified in any namelist, it defaults to 0.*

`mtmdrmsd, mtmdrmsd2` The target RMSD for this reference. If MTMDVARI >0, then the value of MTMDRMSD will vary between MTMDSTEP1 and MTMDSTEP2, so that, e.g. MTMDRMSD(MTMDSTEP1) = MTMDRMSD and MTMDRMSD(MTMDSTEP2) = MTMDRMSD2. *Defaults for MTMDRMSD and MTMDRMSD2 are the values assigned to them in the most recent namelist where MTMDRMSD and MTMDRMSD2 were specified. If MTMDRMSD and MTMDRMSD2 have not been specified in any namelist, they default to 0.0.*

`mtmdforce, mtmdforce2` The force constant for this reference. If MTMDVARI >0, then the value of MTMDFORCE will vary between MTMDSTEP1 and MTMDSTEP2, so that, e.g. MTMDFORCE(MTMDSTEP1) = MTMDFORCE and MTMDFORCE(MTMDSTEP2) = MTMDFORCE2. *Defaults for* MTMDFORCE *and* MTMDFORCE2 *are the values assigned to them in the most recent namelist where* MTMDFORCE *and* MTMDFORCE2 *were specified. If* MTMDFORCE *and* MTMDFORCE2 *have not been specified in any namelist, they default to 0.0.*

`mtmdninc` If MTMDVARI >0 and MTMDNINC > 0, then the changes in the values of of MTMDRMSD and MTMDFORCE are applied as a step function, with NINC steps/iterations between each change in the target values. If MTMDNINC = 0, the change is effected continuously (at every step). *Default for MTMDNINC is the value assigned to MTMDNINC in the most recent namelist where MTMDNINC was specified. If MTMDNINC has not been specified in any namelist, it defaults to 0.*

`mtmdmult` If MTMDMULT=0, and the values of MTMDFORCE changes with step number, then the changes in the force constant will be linearly interpolated from MTMDFORCE→MTMDFORCE2 as the step number changes. If MTMDMULT=1 and the force constant is changing with step number, then the changes in the force constant will be effected by a series of multiplicative scalings, using a single factor, R, for all scalings. *i.e.*

MTMDFORCE2 = RINCREMENTS * MTMDFORCE**

INCREMENTS is the number of times the target value changes, which is determined by MTMDSTEP1, MTMDSTEP2, and MTMDNINC. *Default for MTMDMULT is the value assigned to MTMDMULT in the most recent namelist where MTMDMULT was specified. If MTMDMULT has not been specified in any namelist, it defaults to 0.*

`mtmdmask` Define the atoms that will be used for both the rms superposition between the current structure and the reference structure and the rms difference calculation (and hence the restraint force), as outlined above. Syntax is in Chapter 9.1.1.*Default for MTMDMASK is the value assigned to MTMDMASK in the most recent namelist where MTMDMASK was specified. If MTMDMASK has not been specified in any namelist, it defaults to '*', use all atoms in the fit and force calculations.* \

Namelist &tgt is read for each reference structure. Input ends when a namelist statement with refin = '' (or refin not specified) is found. Note that comments can precede or follow any namelist statement, allowing comments and reference definitions to be freely mixed.

10.6. Low-MODe (LMOD) methods

István Kolossváry's LMOD methods for minimization, conformational searching, and flexible docking[200–203] are fully implemented in Amber. The centerpiece of LMOD is a conformational search algorithm based on eigenvector following of low frequency vibrational modes. It has been applied to a spectrum of computational chemistry domains including protein loop optimization and flexible active site docking.

In the Amber 2020 release, the LMOD optimization code has been updated with major improvements and new features including more accurate flexible docking, the option to visualize normal modes, utilization of random mixtures of low-frequency modes, and the option to work with a range of modes anywhere in the spectrum and not just the lowest frequency modes. The latter is particularly useful for docking where the modes relevant to binding a ligand molecule are usually not the lowest frequency modes. The interface of the new LMOD has not changed, everything works exactly the same way as in Amber18 and earlier versions, a few parameters simply have additional options as documented below. The new features are demonstrated with production quality examples.

Details of the LMOD procedure, and hints on getting good performance, are given Section **??**, which should be consulted before trying the procedures in *sander*. The only difference between the *sander* and *NAB* implementations is the input specification; the same LMOD code is linked into both. The sections below give input details for *sander*.

There are **four "real-life" examples** of performing LMOD searches and in *lmod_vib_anim* **three examples** of updates in Amber20 including generating LMOD-vibration visualization. These are available at *https://ambermd.org/Manuals.php*. Each directory in the tar file has a README file with more information.

10.6.1. XMIN

The XMIN methods for minimization are traditional and manifold in the field of unconstrained optimization: PRCG is a Polak-Ribiere nonlinear Conjugate Gradient algorithm,[204] LBFGS is a Limited-memory Broyden-Fletcher-Goldfarb-Shanno quasi-Newton algorithm,[205] and TNCG is a Truncated Newton linear Conjugate Gradient method with optional LBFGS preconditioning.[206]

Some of the &cntrl namelist variables that control Amber's other minimization facilities also control XMIN. Consequently, non-experts can employ the default XMIN method merely by specifying ntmin = 3.

`maxcyc` The maximum number of cycles of minimization. Default is 1 to be consistent with Amber's other minimization facilities although it may be unrealistically short.

`ntmin` The flag for the method of minimization.

= 3 The XMIN method is used.

= 4 The LMOD method is used. The LMOD procedure employs XMIN for energy relaxation and minimization.

drms The convergence criterion for the energy gradient: minimization will halt when the root-mean-square of the Cartesian elements of the gradient is less than this. Default is 10^{-4}kcal$\cdot$mol$^{-1}\cdot$Å^{-1}. This is consistent with Amber's other minimization facilities. In Amber18 and earlier this default may have been unrealistically strict. In Amber20 this criterion refers to the minimization of the input structure for which the normal modes are computed, and to avoid unnatural vibrational modes it should be set to even stricter values, e.g., 10^{-8}. Compare with input parameter lmod_minimize_grms below.

Other options that control XMIN are in the scope of the &lmod namelist. These parameters enable expert control of XMIN.

lbfgs_memory_depth The depth of the LBFGS memory for LBFGS minimization, or LBFGS preconditioning in TNCG minimization. Default is 3. Suggested alternate value is 5. The value 0 turns off LBFGS preconditioning in TNCG minimization.

matrix_vector_product_method The finite difference Hv matrix-vector product method: "forward" = forward difference, "central" = central difference. Default is forward difference.

xmin_method The minimization method: "PRCG" = Polak-Ribiere Conjugate Gradient, "LBFGS" = Limited-memory Broyden-Fletcher-Goldfarb-Shanno, and "TNCG" = Optionally LBFGS-preconditioned Truncated Newton Conjugate Gradient. Default is LBFGS.

xmin_verbosity The verbosity of the internal status output from the XMIN package: 0 = none, 1 = minimization details, and 2 = minimization and line search details plus CG details in TNCG. Currently, the XMIN status output may be disordered with respect to Amber's output. Default is 0, no output of the XMIN package internal status. Note that XMIN is also available in AmberTools, in the NAB package. An annotated example output corresponding to XMIN_VERBOSITY=2 can be found in the NAB documentation.

10.6.2. LMOD

Some of the options that control LMOD have the same names as Amber's other minimization facilities. See the XMIN section immediately above. Other options that control LMOD are in the scope of the &lmod namelist. These parameters enable expert control of LMOD.

arnoldi_dimension The dimension of the ARPACK Arnoldi factorization. Zero specifies the whole space, that is, three times the number of atoms. Default is 0, the whole space. Basically, the ARPACK package used for the eigenvector calculations solves multiple "small" eigenvalue problems instead of a single "large" problem, which is the diagonalization of the three times the number of atoms by three times the number of atoms Hessian matrix. This parameter is the user specified dimension of the "small" problem. The allowed range is total_low_modes + 1 <= arnoldi_dimension <= three times the number of atoms. The default means that the "small" problem and the "large" problem are identical. This is the preferred, i.e., fastest, calculation for small to medium size systems, because ARPACK is guaranteed to converge in a single iteration. The ARPACK calculation scales with three times the number of atoms times the arnoldi_dimension squared and, therefore, for larger molecules there is an optimal arnoldi_dimension much less than three times the number of atoms that converges much faster in multiple iterations (possibly thousands or tens of thousands of iterations). The key to good performance is to select an arnoldi_dimension such that all the ARPACK storage fits in memory. For proteins, arnoldi_dimension=1000 is generally a good value, but often a very small 50-100 Arnoldi dimension provides the fastest net computational cost with very many iterations.

`conflib_filename` The user-given filename of the LMOD conformational library. The file format is Amber standard formatted trajectory output regardless of the value of &cntrl namelist variable ioutfm. The conformations are stored in energetic order (global minimum energy structure first), the number of conformations<= conflib_size. The default filename is *conflib*.

`conflib_size` The number of conformations to store in conflib. Default is 3.

`energy_window` The energy window for conformation storage; the energy of a stored structure will be in the interval [global_min, global_min + energy_window]. Default is 0, only storage of the global minimum structure.

`explored_low_modes` The number of low frequency vibrational modes used per LMOD iteration. Default is 3.

`frequency_eigenvector_recalc` The frequency, measured in LMOD iterations, of the recalculation of eigenvectors. Default is 3.

`frequency_ligand_rotrans` The frequency, measured in LMOD iterations, of the application of rigid-body rotational and translational motions to the ligand(s). At each frequency_ligand_rotrans-th LMOD iteration number_ligand_rotrans rotations and translations are applied to the ligand(s). Default is 1, ligand(s) are rotated and translated at every LMOD iteration.

`lmod_job_title` The user-given title for the job that goes in the first line of the conflib and lmod_trajectory files. The default job title is "job_title_goes_here".

`lmod_minimize_grms` In Amber18 and earlier the gradient root-mean-square convergence criterion of structure minimization. In Amber 20 this was specified to be the criterion to minimize low-energy conformations; such conformations do not require as strict a convergence criterion as does the first minimization whose convergence is now controlled with input parameter drms, see above. Default is 0.1.

`lmod_relax_grms` The gradient RMS convergence criterion of structure relaxation. Default is 1.0.

`lmod_restart_frequency` The frequency, in LMOD iterations, of conflib updating and LMOD restarting with a randomly chosen structure from the pool. Default is 5.

`lmod_step_size_max` The maximum length of a single LMOD ZIG move. Default is 5.0 Å.

`lmod_step_size_min` The minimum length of a single LMOD ZIG move. Default is 2.0 Å.

`lmod_trajectory_filename` The filename of the LMOD pseudo trajectory. The file format is standard Amber trajectory file. The conformations in this file show the progress of the LMOD search. The number of conformations = number_lmod_iterations + 1. The default filename is *lmod_trajectory*.

`lmod_verbosity` The verbosity of the internal status output from the LMOD package: 0 = none, 1 = some details, 2 = more details, 3 = everything including ARPACK information, 4 = ARPACK only, 5 = visualize normal modes. Currently, the LMOD status output may be disordered with respect to Amber's output. Default is 0, no output of the LMOD package internal status. Note that LMOD is also available in AmberTools, in the NAB package. An annotated example output corresponding to LMOD_VERBOSITY=2 can be found in the NAB documentation.

`monte_carlo_method` The Monte Carlo method: "Metropolis" = Metropolis Monte Carlo, "Total_Quench" = the LMOD trajectory always proceeds towards the lowest lying neighbor of a particular energy well found after exhaustive search along all of the low modes, and "Quick_Quench" = the LMOD trajectory proceeds towards the first neighbor found, which is lower in energy than the current point on the path, without exploring the remaining modes. Default is Metropolis Monte Carlo.

number_free_rotrans_modes In Amber18 and earlier this was solely the number of rotational and translational degrees of freedom (dof) which is related to the number of frozen or tethered atoms in the system: 0 atoms dof=6, 1 atom dof=3, 2 atoms dof=1, >=3 atoms dof=0. In Amber20 the input domain was extended to any non-negative integer, and it represents the number of modes for LMOD to skip. In this way LMOD can now explore a range of modes instead of simply modes starting with the lowest frequency. Note that it is recommended to set this to 0 once in order to examine the ro-translational modes. Default is 6.

number_ligand_rotrans The number of rigid-body rotational and translational motions applied to the ligand(s). Such applications occur at each frequency_ligand_rotrans-th LMOD iteration. Default is 0, no rigid-body motions applied to the ligand(s).

number_ligands The number of ligands for flexible docking. Default is 0, no ligand(s).

number_lmod_iterations The number of LMOD iterations. Default is 10. Note that setting number_lmod_iterations = 0 will result in a single energy minimization.

number_lmod_moves The number of LMOD ZIG-ZAG moves. Zero means that the number of ZIG-ZAG moves is not pre-defined, instead LMOD will attempt to cross the barrier in as many ZIG-ZAG moves as it is necessary. The criterion of crossing an energy barrier is stated above in the "LMOD Procedure" background section. number_lmod_moves $>$ 0 means that multiple barriers may be crossed and LMOD can carry the molecule to a large distance on the potential energy surface without severely distorting the geometry. Default is 0, LMOD will determine automatically where to stop the ZIG-ZAG sequence.

random_seed The seed of the random number generator. Default is 314159.

restart_pool_size The size of the pool of lowest-energy structures to be used for restarting. Default is 3.

rtemperature The value of RT in Amber energy units. This is utilized in the Metropolis criterion. Default is 1.5.

total_low_modes The total number of low frequency vibrational modes to be used. Default is the minimum of 10 and three times the number of atoms minus the number of rotational and translational degrees of freedom (number_free_rotrans_modes).

The following commands are part of the &lmod namelist. These commands control the way LMOD applies explicit translations and rotations to one or more ligands and take effect only if number_ligands >= 1. All commands are lists in square brackets, separated by commas such as [1, 33, 198], however, the list is read by Sander as a string and, therefore, it should be enclosed in single quotes.

ligstart_list, ligend_list The serial number(s) of the first/last atom(s) of the ligand(s). Type integer. The number(s) should correspond to the numbering in the Amber input files prmtop and inpcrd/restart. For example, if there is only one ligand and it starts at atom 193, the command should be ligstart_list = '[193]'. If there are three ligands, the command should be, e.g., '[193, 244, 1435]'. The same format holds for all of the following commands. Note that the ligand(s) can be anywhere in the atom list, however, a single ligand must have continuous numbering between the corresponding ligstart_list and ligend_list values. For example, ligstar_list = '[193, 244, 1435]' and ligend_list = '[217, 302, 1473]'.

ligcent_list The serial number(s) of the atom(s) of the ligand(s), which serves as the center of rotation. Type integer. The value zero means that the center of rotation will be the geometric center of gravity of the ligand.

rotmin_list, rotmax_list The range of random rotation of a particular ligand about the origin defined by the corresponding ligcent_list value is specified by the commands rotmin_list and rotmax_list. The angle is given in +/- degrees. Type float. For example, in case of a single ligand and ligcent_list

= '[0]', rotmin_list = '[30.0]' and rotmax_list = '[180.0]' means that random rotations by an angle +/- 30-180 degrees about the center of gravity of the ligand, will be applied. Similarly, with number_ligands = 2, ligcent_list= 120.0]' means that the first ligand will be rotated like in the single ligand example in this paragraph, but a second ligand will be rotated about its atom number 201, by an angle +/- 60-120 degrees.

`trmin_list, trmax_list` The range of random translation(s) of ligand(s) is defined by the same way as rotation. For example, with number_ligand = 1, trmin_list = '[0.1]' and trmax_list = '[1.0]' means that a single ligand is translated in a random direction by a random distance between 0.1 and 1.0 Angstroms.

11. Free energies

11.1. Thermodynamic integration

In a free energy calculation, the system evolves according to a mixed potential (such as in Eqs. 11.3 or 11.4, below). The essence of free energy calculations is to record and analyze the fluctuations in the values of V_0 and V_1 (that is, what the energies *would have been* with the endpoint potentials) as the simulation progresses. For thermodynamic integration (which is a very straightforward form of analysis) the required averages can be computed "on-the-fly" (as the simulation progresses), and printed at the end of a run. For more complex analyses (such as the Bennett acceptance ratio scheme), one needs to write the history of the values of V_0 and V_1 to a file, and later post-process this file to obtain the final free energy estimates.

There is not room here to discuss the theory of free energy simulations, and there are many excellent discussions elsewhere.[207–209] There are also plenty of recent examples to consult.[210, 211] Such calculations are demanding, both in terms of computer time, and in a level of sophistication to avoid pitfalls that can lead to poor convergence. Since there is no one "best way" to estimate free energies, *sander* and *pmemd* primarily provide the tools to collect the statistics that are needed. Assembling these into a final answer, and assessing the accuracy and significance of the results, generally requires some calculations outside of what Amber provides, *per se*. The discussion here will assume a certain level of familiarity with the basis of free energy calculations.

Both *sander* and *pmemd* have the capability of doing simple thermodynamic free energy calculations, using either PME or generalized Born potentials. When *icfe* is set to 1, information useful for doing thermodynamic integration estimates of free energy changes will be computed. The implementation is different between *sander* and *pmemd*. For *sander*, you must use the *multisander* capability to create two groups, one corresponding to the starting state, and a second corresponding to the ending state (see Section 8.12 for information); you will need a *prmtop* file for each of these two endpoints. For *pmemd*, you use a single *prmtop* file which contains both the starting and ending states. For both *sander* and *pmemd* a mixing parameter λ is used to interpolate between the "unperturbed" and "perturbed" potential functions.

11.1.1. Thermodynamic integration using Sander

There are now two different ways to prepare a thermodynamic integration free energy calculation in Sander. The first is unchanged from previous versions of Amber: Here, the two prmtop files that you create must have the same number of atoms, and the atoms must appear *in the same order* in the two files. This is because there is only one set of coordinates that are propagated in the molecular dynamics algorithm. If there are more atoms in the initial state than in the final, "dummy" atoms must be introduced into the final state to make up the difference. Although there is quite a bit of flexibility in choosing the initial and final states, it is important in general that the system be able to morph "smoothly" from the initial to the final state. Alternatively, you can set up your system to use the softcore potential algorithm described below. This will remove the requirement to prepare "dummy" atoms and allows the two *prmtop* files to have different numbers of atoms.

The basics of the *multisander* functionality are given in Section 8.12, but the mechanics are really quite simple. You start a free energy calculation as follows:

```
mpirun -np 4 sander.MPI -ng 2 -groupfile <filename>
```

Since there are 4 total cpu's in this example, each of the two groups will run in parallel with 2 cpu's each. The number of processors must be a multiple of two. The *groups* file might look like this:

```
-O -i mdin -p prmtop.0 -c eq1.x -o md1.o -r md1.x -inf mdinfo
-O -i mdin -p prmtop.1 -c eq1.x -o md1b.o -r md1b.x -inf mdinfob
```

The input (*mdin*) and starting coordinate files must be the same for the two groups. Furthermore, the two *prmtop* files must have the same number number of atoms, in the same order (since one common set of coordinates will be used for both.) The simulation will use the masses found in the first *prmtop* file; in classical statistical mechanics, the Boltzmann distribution in coordinates is independent of the masses so this should not represent any real restriction.

On output, the two restart files should be identical, and the two output files should differ only in trivial ways such as timings; there should be no differences in any energy-related quantities, except if energy decomposition is turned on (*idecomp* > 0); then only the output file of the first group contains the per residue contributions to $\langle \partial V/\partial \lambda \rangle$. For our example, this means that one could delete the *md1b.o* and *md1b.x* files, since the information they contain is also in *md1.o* and *md1.x*. (It is a good practice, however, to check these file identities, to make sure that nothing has gone wrong.)

11.1.2. Basic inputs for thermodynamic integration

icfe The basic flag for free energy calculations. The default value of 0 skips such calculations. Setting this flag to 1 turns them on, using the mixing rules in Eq. 11.3, below.

clambda The value of λ for this run, as in Eqs. 11.3 and 11.4, below. Zero corresponds to the unperturbed Hamiltonian V_0. λ=1 corresponds to the perturbed Hamiltonian V_1.

klambda The exponent in Eq. 11.4, below.

tishake Flag that determines how SHAKE is handled:

- **= 0** Coordinates are synchronized after SHAKE, no constraints removed (default).
- **= 1** SHAKE is removed between bonds containing one common and one unique atom. This was the default in previous versions of *sander*. Note that disabling SHAKE requires the use of a 1 fs timestep.

11.1.2.1. Input flags specific to Sander

idecomp Flag that turns on/off decomposition of $\langle \partial V/\partial \lambda \rangle$ on a per-residue level. The default value of 0 turns off energy decomposition. A value of 1 turns the decomposition on, and 1-4 nonbonded energies are added to internal energies (bond, angle, torsional). A value of 2 turns the decomposition on, and 1-4 nonbonded energies are added to EEL and VDW energies, respectively. The frequency by which values of $\langle \partial V/\partial \lambda \rangle$ are included into the decomposition is determined by the NTPR flag. This ensures that the sum of all contributions equals the average of all total $\langle \partial V/\partial \lambda \rangle$ values output every NTPR steps. All residues, including solvent molecules, have to be chosen by the RRES card to be considered for decomposition. The RES card determines which residue information is finally output. The output comes at the end of the *mdout* file. For each residue contributions of internal -, VdW-, and electrostatic energies to $\langle \partial V/\partial \lambda \rangle$ are given as an average over all (NSTLIM/NTPR) steps. In a first section total per residue values are output followed below by further decomposed values from backbone and sidechain atoms.

11.1.3. Background theory of thermodynamic integration

The *sander and pmemd* programs do not compute free energies; it is up to the user to combine the output of several runs (at different values of λ) and to numerically estimate the integral:

$$\Delta A = A(\lambda = 1) - A(\lambda = 0) = \int_0^1 \langle \partial V/\partial \lambda \rangle_\lambda \, d\lambda \tag{11.1}$$

If you understand how free energies work, this should not be at all difficult. However, since the actual values of λ that are needed, and the exact method of numerical integration, depend upon the problem and upon the precision desired, we have not tried to pre-code these into the program.

The simplest numerical integration is to evaluate the integrand at the midpoint:

$$\Delta A \simeq \langle \partial V/\partial \lambda \rangle_{1/2}$$

This might be a good first thing to do to get some picture of what is going on, but is only expected to be accurate for very smooth or small changes, such as changing just the charges on some atoms. Gaussian quadrature formulas of higher order are generally more useful:

$$\Delta A = \sum_i w_i \langle \partial V/\partial \lambda \rangle_i \tag{11.2}$$

Some weights and quadrature points are given in the accompanying table; other formulas are possible,[212] but the Gaussian ones listed there are probably the most useful. The formulas are always symmetrical about λ= 0.5, so that λ and $(1-\lambda)$ both have the same weight. For example, if you wanted to use 5-point quadrature, you would need to run five jobs, setting $\lambda$to 0.04691, 0.23076, 0.5, 0.76923, and 0.95308 in turn. (Each value of λ should have an equilibration period as well as a sampling period; this can be achieved by setting the *ntave* parameter.) You would then multiply the values of $\langle \partial V/\partial \lambda \rangle_i$ by the weights listed in the Table, and compute the sum.

When *icfe=1* and *klambda* has its default value of 1, the simulation uses the mixed potential function:

$$V(\lambda) = (1-\lambda)V_0 + \lambda V_1 \tag{11.3}$$

where V_0 is the potential with the original Hamiltonian, and V_1 is the potential with the perturbed Hamiltonian. The program also computes and prints $\langle \partial V/\partial \lambda \rangle$ and its averages; note that in this case, $\langle \partial V/\partial \lambda \rangle = V_1 - V_0$. This is referred to as linear mixing, and is often what you want unless you are making atoms appear or disappear. If some of the perturbed atoms are "dummy" atoms (with no van der Waals terms, so that you are making these atoms "disappear" in the perturbed state), the integrand in Eq. 11.1 diverges at λ= 1; this is a mild enough divergence that the overall integral remains finite, but it still requires special numerical integration techniques to obtain a good estimate of the integral.[208] *Sander and pmemd* implement one simple way of handling this problem: if you set *klambda* > 1, the mixing rules are

$$V(\lambda) = (1-\lambda)^k V_0 + [1-(1-\lambda)^k]V_1 \tag{11.4}$$

where k is given by *klambda*. Note that this reduces to Eq. 11.3 when k = 1, which is the default. If $k \geq$4, the integrand remains finite as $\lambda \rightarrow 1$.[208] We have found that setting k= 6 with disappearing groups as large as tryptophan works, but using the softcore option (*ifsc*>0) instead is generally preferred.[213] Note that the behavior of $\langle \partial V/\partial \lambda \rangle$ as a function of λ is not monotonic when *klambda* > 1. You may need a fairly fine quadrature to get converged results for the integral, and you may want to sample more carefully in regions where $\langle \partial V/\partial \lambda \rangle$ is changing rapidly.

Notes:

1. This is implemented in *sander* by calling the force() routine independently for each *multisander* group and then combining the forces on each step. For a fixed number of processors this increases the cost of the calculation compared with the *pmemd* code, which only calculates the differences between V_0 and V_1.

2. It is rather easy to make mistakes when running TI calculations. It is generally good to carry out a short run (say 50 steps) setting ntpr=1. Then check the following; if either test fails, be sure to fix the problem before proceeding.

   a) The restart files from V_0 and V_1 should be identical for *sander* (for *pmemd* there will only be a single restart file).

   b) If you diff the output files for *sander*, there should only be simple differences (for *pmemd* there will only be a single combined output file). All energies, temperatures, pressures, etc. should be the same in the two files. Simulations with *sander* using the QM/MM facility may show differences in the SCF energies, but be sure that the total energies, and all the MM components, are the same.

3. Eq. 11.4 is designed for having dummy atoms in the perturbed Hamiltonian, and "real" atoms in the regular Hamiltonian. You must ensure that this is the case when you set up the system in LEaP. (See the softcore

section, below, for a more general way to handle disappearing atoms, which does not require dummy atoms at all.)

4. One common application of this model is to pKa calculations, where the charges are mutated from the protonated to the deprotonated form. Since H atoms bonded to oxygen already have zero van der Waals radii (in the Amber force fields and in TIP3P water), once their charge is removed (in the deprotonated form) they are really then like dummy atoms. For this special situation, there is no need to use *klambda* > 1: since the van der Waals terms are missing from both the perturbed and unperturbed states, the proton's position can never lead to the large contributions to $\langle \partial V / \partial \lambda \rangle$ that can occur when one is changing from a zero van der Waals term to a finite one.

5. The implementation requires that the masses of all atoms be the same on all threads. To enforce this, the masses found for V_0 are used for V_1 as well. In classical statistical mechanics, the canonical distribution of configurations (and hence of potential energies) is unaffected by changes in the masses, so this should not pose a limitation. Since the masses for V_1 are ignored, they do not have to match those found for V_0.

6. Special care needs to be taken when using SHAKE for atoms whose force field parameters differ in the two end points. The same bonds must be SHAKEN in both cases, and the equilibrium bond lengths must also be the same. By default, the coordinates from V_0 are synchronized with those from V_1 after SHAKE. This will work for small perturbations, but if there is a significant change in bond length, it may be necessary to use the *noshakemask* input to remove SHAKE from the regions that are being perturbed. If this is done, be sure to set *tishake*=1 and to use a 1 fs timestep. Special care needs to be taken when water molecules are part of the region that is changing. You need to make sure that the "number of 3-point waters" is the same in both V_0 and V_1. This may require setting *jfastw* and/or building the structure so that *sander or pmemd* do not think that the water molecules involved are actually rigid waters. Also, just setting *noshakemask* might not be enough, since this flag does not affect the *settle* routine that handles rigid waters.

n	λ_i	$1-\lambda_i$	w_i
1	0.5		1.0
2	0.21132	0.78867	0.5
3	0.1127	0.88729	0.27777
	0.5		0.44444
5	0.04691	0.95308	0.11846
	0.23076	0.76923	0.23931
	0.5		0.28444
7	0.02544	0.97455	0.06474
	0.12923	0.87076	0.13985
	0.29707	0.70292	0.19091
	0.5		0.20897
9	0.01592	0.98408	0.04064
	0.08198	0.91802	0.09032
	0.19331	0.80669	0.13031
	0.33787	0.66213	0.15617
	0.5		0.16512
12	0.00922	0.99078	0.02359
	0.04794	0.95206	0.05347
	0.11505	0.88495	0.08004
	0.20634	0.79366	0.10158
	0.31608	0.68392	0.11675
	0.43738	0.56262	0.12457

Table 11.1.: *Abscissas and weights for Gaussian integration.*

11.1.4. Softcore Potentials in Thermodynamic Integration

Softcore potentials provide an additional way to perform thermodynamic integration calculations in Amber. The system setup has been simplified so that appearing and disappearing atoms can be present at the same time and no dummy atoms need to be introduced. For *sander*, two prmtop files, corresponding to the start and end states (V_0 and V_1) of the desired transformation need to be used. The common atoms that are present in both states need to appear in the same order in both prmtop files and must have identical starting positions. In addition to the common atoms, each process can have any number of unique soft core atoms, as specified by scmask. For *pmemd*, a single prmtop file is used, containing the unique atoms for both the start and end states. The soft core atoms are specified by scmask1 and scmask2 for V_0 and V_1 respectively.

A modified version of the vdW equation is used to smoothly switch off non-bonded interactions of these atoms with their common atom neighbors:

$$V_{V_0,disappearing} = 4\varepsilon(1-\lambda)\left[\frac{1}{\left[\alpha\lambda+\left(\frac{r_{ij}}{\sigma}\right)^6\right]^2}-\frac{1}{\alpha\lambda+\left(\frac{r_{ij}}{\sigma}\right)^6}\right] \tag{11.5}$$

$$V_{V_1,appearing} = 4\varepsilon\lambda\left[\frac{1}{\left[\alpha(1-\lambda)+\left(\frac{r_{ij}}{\sigma}\right)^6\right]^2}-\frac{1}{\alpha(1-\lambda)+\left(\frac{r_{ij}}{\sigma}\right)^6}\right] \tag{11.6}$$

Please refer to Ref [213] for a description of the implementation and performance testing when compared to the TI methods described above using *sander*. For similar information pertaining to pmemd please see Ref [214]. Note that the term "disappearing" is used here, but it would probably be better to say that atoms present in V_0 but not in V_1 are "decoupled" from their environment: the interactions among the "disappearing" atoms are not changed, and do not contribute to $\langle \partial V/\partial\lambda \rangle$. If the disappearing atoms are a separate molecule (say a non-covalently-bound ligand), this can be viewed as a transfer to the gas-phase.

Note that a slightly different setup is required for using soft core potentials compared to older TI-implementations. Specifically, the difference is that to add or remove atoms without soft core potentials, they are transformed into interactionless dummy particles, so both end state prmtop files have the same number of atoms. When using soft core potentials instead, no dummy atoms are needed and the end states should be built without them. Therefore prmtop files for non soft core simulations may have to be adapted to be used with soft core potentials and vice versa.

All bonded interactions of the unique atoms are recorded separately in the output file (see below). Any bond, angle, dihedral or 1-4 term that involves at least one appearing or disappearing atom is not scaled by λ and does not contribute to $\langle \partial V/\partial\lambda \rangle$. Therefore, output from both processes will not be identical when soft core potentials are used. Softcore transformations avoid the origin singularity effect and therefore linear mixing can (and should) always be used with them. Since the unique atoms become decoupled from their surroundings at high or low lambdas and energy exchange between them and surrounding solvent becomes inefficient, a Berendsen type thermostat should not be used for SC calculations. Unlike in previous versions, SHAKE constraints are not automatically removed from bonds between common and unique atoms. Instead, the coordinates corresponding to common atoms in $V_0$are synchronized with those of V_1. The original behavior can be restored using *tishake*. The icfe and klambda parameters should be set to 1 for a soft core run and the desired lambda value will be specified by clambda. When using softcore potentials with *sander*, λ values should be picked so that 0.01 < clambda < 0.99. The *pmemd* implementation allows lambda to be set to any value between 0.0 and 1.0, thus simulations at the endpoints are possible.

Additionally, the following parameters are available to control the TI calculation:

`ifsc` Flag for soft core potentials

= 0 SC potentials are not used (default)

= 1 SC potentials are used. Be sure to use prmtop files that are suitable for this, i.e. not-containing dummy atoms (see above)

`scalpha` The α parameter in 11.5 and 11.6, its default value is 0.5. Other values have not been extensively tested

`logdvdl` If set to .ne. 0, a summary of all $\partial V/\partial\lambda$ values calculated during every step of the run will be printed out at the end of the simulation for postprocessing.

`dvdl_norest` This option is now deprecated. Restraints involving soft core atoms are now decoupled from the rest of the system. The energy is listed separately and does not contribute to $\partial V/\partial\lambda$.

`dynlmb` If set to a value .gt. zero, clambda is increased by dynlmb every ntave steps. This can be used to perform simulations with dynamically changing lambdas.

`crgmask` Specifies a number of atoms (in ambmask format) that will have their atomic partial charges set to zero. This is mainly for convenience because it removes the need to build additional prmtop files with uncharged atoms for TI calculations involving the removal of partial charges.

11.1.4.1. Input flags specific to Sander

`scmask` Specifies the unique (soft core) atoms for this process in ambmask format. This, along with crgmask, is the only parameter that will frequently be different in the two mdin files for V_0 and V_1. It is valid to set scmask to an empty string. A summary of the atoms in scmask is printed at the end of mdout.

11.1.4.2. One step transformations using soft core electrostatics

Alternatively to the two-step process of removing charges from atoms first and then changing the vdW parameters of chargeless atoms in a second TI calculation, *sander and pmemd* also have a soft core version of the Coulomb equation implemented for single step transformations under periodic boundary conditions. This is automatically applied to all atoms in scmask and their interactions with common atoms are given by:

$$V_{V_0,disappearing} = (1-\lambda)\frac{q_i q_j}{4\pi\varepsilon_o\sqrt{\beta\lambda + r_{ij}^2}} \tag{11.7}$$

for disappearing atoms. Replace λ by $(1-\lambda)$ and vice versa for the form for appearing atoms. This introduces a new parameter β which controls the 'softness' of the potential. This is set in the input file via:

`scbeta` The parameter β in 11.7. Default value is 12A^2, other values have not been extensively tested.

With the use of soft core vdW and electrostatics interactions, arbitrary changes between systems are possible in single TI calculations. However, due to the unusual potential function forms introduced, it is not always clear that a single-step calculation will converge faster than one broken down into several steps. Ref. [215] contains detailed information on the performance of such single step TI calculations.

11.1.5. Collecting potential energy differences for FEP calculations

In addition to the Thermodynamic Integration capabilities described above, *sander* can also collect potential energy values during free energy simulation runs for postprocessing by e.g. the Bennett acceptance ratio scheme. This will make sander calculate at given points during the simulation the total potential energy of the system as it would be for different λ-values at this conformation. This functionality is controlled by:

`ifmbar` If set to 1 (Default = 0), additional output is generated for later postprocessing.

mbar_states number of lambda windows considered.

`mbar_lambda` lambda windows simulated.

`For` example, if you want to run mbar with 15 lambda windows at 0.00, 0.05, 0.10, 0.15, 0.20, 0.30, 0.40, 0.50, 0.60, 0.70, 0.80, 0.85, 0.90, 0.95, 1.00, you would use the following options:

```
ifmbar = 1,
mbar_states = 15,
mbar_lambda = 0.00, 0.05, 0.10, 0.15, 0.20, 0.30, 0.40, 0.50, 0.60, 0.70, 0.80, 0.85, 0.90, 0.95,
```

The options below have been deprecated. They are here for anyone using AMBER 16 or older, but will not work in AMBER 18.

`bar_intervall` Compute potential energies every bar_intervall steps (Default = 100)

`bar_l_min` Minimum λ-value (Default = 0.1)

`bar_l_max` Maximum λ-value (Default = 0.9)

`bar_l_incr` The increment to increase λ by between the minimum and maximum (Default = 0.1)

Such energy collection will normally be part of a regular free energy calculation (using icfe=1 and ifsc=1) involving simulations at various λ-values. Activating this functionality will not have any influence on the simulation trajectory which will evolve according to the preset *clambda* value, it is merely a bookkeeping scheme that removes the necessity of postprocessing output files later.

11.2. Linear Interaction Energies

sander contains rudimentary facilities to compute binding free energies using the linear interaction energy model.

`ilrt` if set to 1, turns on the computation of LIE contributions (default=0)

`lrt_interval` Computer LIE contributions every lrt_interval MD steps (default=50)

`lrtmask` The 'solute'. Interaction Energies between the atoms in lrtmask and the remainder of the system are computed.

The LIE facilities work by computing the system energies several times using different charge and vdW-parameter sets. This results in reduced performance if lrt_interval is set to less than approx. 10. The LIE output at the end of the *mdout* file gives the electrostatic interaction energy between the solute and rest of the system *times 0.5*, i.e. in accordance with the original formulation of LIE theory. The solute SASA and vdW-interaction energy with its surroundings is calculated unscaled.

11.3. Adaptively Biased MD, Steered MD, Umbrella Sampling with REMD and String Method

11.3.1. Overview

The following describes a suite of modules useful for the calculation of the free energy associated with a reaction coordinate $\sigma(\mathbf{r}_1,\ldots,\mathbf{r}_N)$ (which is defined as a smooth function of the atomic positions $\mathbf{r}_1,\ldots,\mathbf{r}_N$):

$$f(\xi) = -k_B T \ln \left\langle \delta \left[\xi - \sigma(\mathbf{r}_1,\ldots,\mathbf{r}_N)\right] \right\rangle,$$

(the angular brackets denote an ensemble average, k_B is the Boltzmann constant and T is the temperature) that is also frequently referred to as the *potential of mean force*.

Specifically, new frameworks are provided for equilibrium umbrella sampling and steered molecular dynamics that enhance the functionality delivered by earlier implementations (described earlier in this manual), along with a new Adaptively Biased Molecular Dynamics (ABMD) method [216] that belongs to the general category of umbrella sampling methods with a time-dependent potential. Such methods were first introduced by Huber, Torda and van

Gunsteren (the Local Elevation Method [217]) in the molecular dynamics (MD) context, and by Wang and Landau in the context of Monte Carlo simulations [218]. More recent approaches include the metadynamics method [219, 220]. All these methods estimate the free energy of a reaction coordinate from an evolving ensemble of realizations, and use that estimate to bias the system dynamics to flatten an effective free energy surface. Collectively, these methods may all be considered to be umbrella sampling methods with an evolving potential. The algorithms discussed here were developed by the group of Prof. Celeste Sagui (sagui@ncsu.edu) and Prof. Christopher Roland (cmroland@ncsu.edu); the current version was implemented by Dr. Volodymyr Babin.

The ABMD method grew out of attempts to speed up and streamline the metadynamics method for free energy calculations with a *controllable* accuracy. It is characterized by a favorable scaling in time, and only a few (two) control parameters. It is formulated in terms of the following equations:

$$m_a \frac{\mathrm{d}^2 \mathbf{r}_a}{\mathrm{d}t^2} = \mathbf{F}_a + \frac{\partial}{\partial \mathbf{r}_a} U\left[t|\sigma\left(\mathbf{r}_1,\ldots,\mathbf{r}_N\right)\right],$$

$$\frac{\partial U(t|\xi)}{\partial t} = \frac{k_B T}{\tau_F} G\left[\xi - \sigma\left(\mathbf{r}_1,\ldots,\mathbf{r}_N\right)\right],$$

where the first equation represents Newton's law that governs ordinary MD (temperature and pressure regulation terms are not shown) augmented with an additional force coming from the time dependent biasing potential $U(t|\xi)$ [$U(t=0|\xi)=0$], whose time evolution is given by the second equation. $G(\xi)$ is a positive definite and symmetric kernel, which may be thought of as a smoothed Dirac delta function. For large enough τ_F (the flooding timescale) and small kernel width, the biasing potential $U(t|\xi)$ converges towards $-f(\xi)$ as $t \to \infty$.

Our numerical implementation of the ABMD method involves the use of a bi-weight kernel along with the use of cubic B-splines (or products thereof) to discretize the biasing potential $U(t|\xi)$ w.r.t. ξ, and an Euler-like scheme for time integration. ABMD admits two important extensions, which lead to a more uniform flattening of $U(t|\xi)+f(\xi)$ due to an improved sampling of the "evolving" canonical distribution. The first extension is identical in spirit to the *multiple walkers metadynamics* [221, 222]. It amounts to carrying out several different MD simulations biased by the same $U(t|\xi)$, which evolves via:

$$\frac{\partial U(t|\xi)}{\partial t} = \frac{k_B T}{\tau_F} \sum_{\alpha} G\left[\xi - \sigma\left(\mathbf{r}_1^{\alpha},\ldots,\mathbf{r}_N^{\alpha}\right)\right],$$

where α labels different MD trajectories. A second extension is to gather several different MD trajectories, each bearing its own biasing potential and, if desired, its own distinct collective variable, into a generalized ensemble for "replica exchange" with modified "exchange" rules [223–225]. Both extensions are advantageous and lead to a more uniform flattening of $U(t|\xi)+f(\xi)$.

In order to assess and improve the accuracy of the free energies, the ABMD accumulations may need to be followed up with equilibrium umbrella sampling runs, which make use of the biasing potential $U(t|\xi)$ as is. Such a procedure is very much in the spirit of adaptive umbrella sampling. With these runs, one calculates the biased probability density:

$$p^B(\xi) = \left\langle \delta\left[\xi - \sigma(\mathbf{r}_1,\ldots,\mathbf{r}_N)\right]\right\rangle_B.$$

The idea here is that if, as a result of an ABMD run, $f(\xi)+U(t|\xi)=0$ exactly, then the biased probability density $p^B(\xi)$ would be flat (constant). In practice, this is typically not the case, but one can use $p^B(\xi)$ to "correct" the free energy via:

$$f(\xi) = -U(\xi) - k_B T \ln p^B(\xi).$$

With the ABMD procedure, one can obtain accurate free energy curves and equilibrium properties. We note that to obtain ABMD free energies requires a (minor) amount of post-processing by means of the nfe-umbrella-slice utility freely available in AmberTools as described in Subsection 11.3.7. This methodology has been applied to a variety of biomolecular systems, including small peptides [216, 226, 227], sugar puckering [228], polyproline systems [229–231], guest-host systems [232, 233], polyglutamine systems [234, 235], and DNA systems [236–240]. In addition, SMD simulations (discussed below) have been used to examine transition pathways and mechanisms, to estimate free energy differences [230, 241], and to calculate transition rates [242–244].

While the above represents the basic ABMD implementation, AMBER20 introduced three additional algorithms

– the Well-Tempered (WT) ABMD, a selection mechanism for multiple walker ABMD and Driven ABMD (D-ABMD)[245] – all of which enhance the stability and convergence of an ABMD simulation. Also implemented is the Swarms-of-Trajectories String Method (STSM) [246], which gives a way of exploring the Minimum Free Energy Path (MFEP) on free energy lansdscape. Current version of these codes were inplemented by Dr. Mahmoud Moradi (moradi@uark.edu), Dr. Feng Pan (fpan3@ncsu.edu) and Ashkan Fakharzadeh (afakhar@ncsu.edu).

The Well-Tempered ABMD: An alternative to the follow-up equilibrium simulations for increased ABMD accuracy is provided by the WT-ABMD, which is implemented in the spirit of the WT-metadynamics [247]. In the original ABMD implementation, the history dependent biasing potential is built up at a fixed rate:

$$U(\xi,t) = U^0(\xi) + \int_0^t dt'\, \omega\, G(\xi - \xi^{t'}), \tag{11.8}$$

in which $U(\xi,t)$ is the biasing potential at time t, U^0 is an arbitrary function that typically represents the initial guess for the biasing potential (in the absence of a guess, this is assumed to be flat) and $\omega = k_B T/\tau_F$ is a constant, unbiased rate. As the simulation proceeds and reaches convergence, then $\left\langle U(\xi,t\to\infty)\right\rangle_a \approx U^s(\xi)+u(t)$, in which $\left\langle . \right\rangle_a$ is the ensemble-average over the adaptive trajectories, the stationary term is $U^s \approx -F(\xi)$, and $u(t)$ is an additive time-dependent constant [247]. Unfortunately, updating the biasing potential at the same rate throughout the simulation may lead to a poorly converged result, since the biasing potential ends up fluctuating around $-F(\xi)$ with an amplitude that depends on ω.

One way to resolve this problem is to update the kernel at a non-uniform rate by means of a "well-tempered" ω:

$$U(\xi,t) = U^0(\xi) + \int_0^t dt'\, \omega(\xi^{t'},t')\, G(\xi - \xi^{t'}), \tag{11.9}$$

in which $\omega(\xi,t)$ is a time-dependent, non-uniform rate chosen to be $\omega_0 \mathrm{e}^{-\beta' U(\xi,t)}$ ($1/\beta' = k_B T'$ where T' is a pseudo-temperature) that reduces to a constant ω_0 in the $\beta' \to 0$ limit (*i.e.*, resulting in conventional ABMD). With this choice, one can show that $\left\langle U(\xi,t\to\infty)\right\rangle_a \approx U^s(\xi)+u(t)$, ($u(t)$ is an additive constant) in which $U^s(\xi)$ and $F(\xi)$ are related via $U^s(\xi) = -(1+\frac{\beta'}{\beta})^{-1}F(\xi)$ or $F(\xi) = -(1+\frac{T}{T'})U^s(\xi)$. This way of updating the biasing potential leads to a considerably smoother convergence to the desired free energy and more stable ABMD simulations.

Multiple walker selection algorithm: The ABMD multiple walker algorithm can be improved by allowing for periodic interactions between the different walkers and "resampling" on-the-fly. The rationale behind this is that not all walkers are equally effective in sampling the configuration space. A situation that is all too common is that different walkers end up being “bunched up" or clustered together in some local metastable region, because of hidden barriers that are oriented along orthogonal degrees of freedom to the reaction coordinate. To improve this situation, one would like to facilitate walkers that are sampling the undersampled regions of phase space, and force the walkers in the oversampled regions to move away and explore regions not yet covered. Such an algorithm has previously been implemented via scripts in the NAMD code for the adaptive biasing force algorithm [248].

A resampling or selection algorithms for interacting multiple walkers requires a continual monitoring of the walkers by means of a periodic evaluation of a fitness function and a resampling of the walkers according to their fitness efficiency[248]. Efficient walkers that are wandering in the undersampled regions are enhanced by being cloned, while inefficient walkers found in the oversampled regions of phase space are correspondingly killed. This procedure is then repeated periodically during the simulation, thereby accelerating convergence to a more uniform distribution of walkers and flattening of the free energy landscape.

Our specific interacting/resampling/selection multiple-walker algorithm is implemented as follows. Each walker n is assigned a weight w_n, which is evaluated at the end of each resampling period of time τ. At the i^{th} resampling period, i.e., from time $t_{(i-1)} = (i-1)\tau$ to $t_i = i\tau$, walker n moves through configuration space building up its own trajectory $(r_1^n,\ldots,r_N^n)$. The weights are then tested and updated every fixed time interval of length τ. Specifically, after the i^{th} time interval, weights are estimated by:

$$w_n = K^{-1} \exp\left(\int_{t_{i-1}}^{t_i} S(\xi_n^t) dt \right),$$

where ξ_n^t represents the collective variable evaluated at time t for trajectory n, $K = \sum_{n=1}^{N_w} w_n$ is the normalization factor, and

$$S(\xi) = C\nabla^2(\rho(\xi))/\rho(\xi),$$

with $\rho(\xi)$ representing the density of microstates in the collective variable space and C a constant. The quantity $S(\xi)$ will be positive typically if the walker is found in the undersampled regions, which have a convex density function. Similarly, a negative $S(\xi)$ value indicates that the system is in the concave region of the density function, which typically is oversampled. In the context of ABMD implementation, the biasing potential is approximately proportional to the histogram of the collective variable by construction, and represents a good estimate for ρ. The implementation is therefore straightforward; the integral above is estimated for each trajectory independently by summing over $S(\xi_n^t)$ at every step from $t = t_{i-1}$ to $t = t_i$, in which Δt is the MD *timestep*. At the end of each period the walkers send their unnormalized weight estimates to the "master processor" to normalize them. A stochastic resampling method is then used to clone/kill the replicas based on their weight factors [248]. The number of copies present in the next period for walker n is determined by the integer number:

$$\begin{aligned} W_1 &= \lfloor \eta_1 + N_w w_1 \rfloor, \\ W_n &= \lfloor \eta_n + N_w \sum_{m=1}^{n} w_m \rfloor - \lfloor \eta_n + N_w \sum_{m=1}^{n-1} w_m \rfloor, \quad \text{for } n > 1. \end{aligned}$$

in which $0 < \eta_n < 1$ is drawn from a uniform distribution (using a random number generator). The atomic coordinates and velocities of the walkers with $N_n > 0$ are "sent" to N_n walkers. The resampling algorithm above guarantees $\sum_n W_n = N_w$.

In terms of an ABMD simulation, the selection algorithm is most beneficial during the initial and middle parts of the simulation when there are large variations in the biasing potential. In the latter parts, when the effective free energy is almost flat, the distribution of walkers should be roughly uniform. In that case, the selection mechanism is unnecessary and, if one wishes to continue the simulation, it is best to proceed with the non interacting multiple walker algorithm. It has been found that a convenient stopping mechanism may be based on the entropy of the weights. Defining $H = \sum_n w_n log(w_n)$, the selection mechanism will be stopped if $E_w = H - log(1/N_w)$ goes below -$\varepsilon log(1/N_w)$. Here, $log(1/N_w)$ represents the entropy of uniform weights, and the stopping parameter ε varies between $0 \leq \varepsilon \leq 1$. When $\varepsilon = 0$, the algorithm never stops, while $\varepsilon = 1$ forces a stop irrespective of the values of the weights.

In addition to ε, there are also two other user-defined variables in the selection algorithm, including the constant C and the interval time τ. While the physical interpretation of τ is straightforward, C represents a pseudo diffusion constant. One may think of the selection algorithm as an induced diffusion in the reaction coordinate space; The larger the value of C, the faster the system will diffuse along the reaction coordinate space. Therefore C determines the strength or aggressiveness of the resampling algorithm. The most efficient value for C is dependent on the nature of the collective variable and the shape of its density ρ. Since the best choice of C for a given problem is somewhat of an art, we refer the interested reader to the ABMD tutorials on the AMBER webpage for insight into choosing this variable. Finally, we also note that the multiple walker selection mechanism can be invoked as is or in conjunction with the WT-ABMD for enhanced stability and convergence.

Driven ABMD: ABMD and SMD schemes are both powerful nonequilibrium sampling methods; however, each comes with its own practical limitations. For instance, SMD is often associated with a very slow convergence if used for free energy calculations. However it can be used to explore the transition paths, at least qualitatively; an advantage over ABMD, in which the system starting from one end of the configuration space (the reactant) may take a long time to visit the other end (the product). SMD and ABMD schemes, however can be integrated into a novel driven adaptive-bias 3 scheme, termed driven ABMD (D-ABMD) that takes advantage of both its driven and adaptive-bias components and is advantageous over both components in isolation. D-ABMD has an advantage over conventional (or well-tempered) ABMD in that it ensures the exploration of the transition pathway (from one end to the other) in the early stages of the simulation and gradually improves the estimate of the free energies almost uniformly along the reaction coordinate. D-ABMD has also an advantage over the conventional SMD in that the effective free energy surface gradually becomes smooth and flat such that the system can move along the reaction coordinate with progressively less amount of work. The D-ABMD method is similar to D-MetaD method,

```
&colvar
   cv_type = STRING
   cv_ni = N, cv_nr = M
   cv_i = i1, i2, ..., iN
   cv_r = r1, r2, ..., rM
/
```

Figure 11.1.: *Syntax of reaction coordinate definition:* `cv_type` *is a* `STRING`*,* `cv_i` *is a list of integer numbers and* `cv_r` *is a list of real numbers.*

which was recently introduced in Ref.[245] as an example of driven, adaptive-bias schemes.

In order to combine the two schemes described above, we have developed a driven adaptive-bias scheme that adds an adaptive $U_a(\xi,t)$ and a driving $U_d(\xi,t)$ potential to the Hamiltonian. We use an iterative approach in which an independent simulation is performed from time t = 0 to t = T in the n^{th} iteration (n = 1, 2, . . .), biased by the potential $U_d(\xi,t)+U_a^n(\xi,t)$ in which $U_d(\xi,t)=\frac{k}{2}(\xi-\eta(t))^2$ for all n ($\eta(t)$ is moving center of the SMD harmonic potential in the ξ space), and:

$$U_a^n(\xi,t)=U^{n-1}(\xi)+\int_0^t dt'\omega(\xi^{t'},t')K(\xi-\xi^{t'})e^{-\beta\omega^{t'}}$$

in which ω^t is either defined as the accumulated work or the transferred work. The accumulated and transferred works are defined as $\omega_{ac}^t=\int_0^t dt'\frac{\partial}{\partial t'}U_d(\xi^{t'},t')$ and $\omega_{tr}^t=\omega_{ac}^t-U_d(\xi^t,t)$. Theoretically the $e^{-\beta\omega_{tr}^{t'}}$ factor or "constant weight" is more accurate but for practical reasons the $e^{-\beta\omega_{ac}^{t'}}$ factor or "pulling wight" is preferred. Particularly, in our algorithm, the constant weight $e^{-\beta\omega_{tr}^{t'}}=e^{-\beta\omega_{ac}^{t'}}e^{\beta U_d(\xi^{t'},t')}$ may become instable for large biasing potentials. To avoid the instability in either case a cutoff for ω^t is used (i.e., the algorithm will not be applied if ω^t is smaller than the cutoff). At the moment, Driven ABMD is only applicable to one-dimensional reaction coordinate.

If any of these modules prove to be useful, please consider quoting the following papers: V. Babin, C. Roland and C. Sagui, *"Adaptively biased molecular dynamics for free energy calculations"*, J. Chem. Phys. **128**, 134101 (2008); V. Babin, V. Karpusenka, M. Moradi, C. Roland and C. Sagui, *"Adaptively biased molecular dynamics: an umbrella sampling method with a time-dependent potential"*, Int. J. Quant. Chem. **109**, 3666 (2009).

From Amber16, we implement these modules from SANDER to PMEMD and the modules are GPU compatible. To keep the consistency in format, we do a series of changes and updates to the usage of these modules. One big change is that you must set ***infe*** = 1 in &cntrl to activate these modules. Also, the input format has been changed to namelist style and reaction coordinate variables will be read from separate files. For the details, please read Subsection **??**

infe This variable controls the usage of the non-equilibrium free energy method. When *infe=0*, the ABMD and related methods are turned off; when *infe=1*, they are turned on and the blocks `&smd`, `&pmd`, `&abmd`, `&bbmd` and `&stsm` will be recognized. The default value is 0. Note that use of these algorithms may require a (minor) amount of post-processing by means of the nfe-umbrella-slice utility freely available in AmberTools described in Subsection 11.3.7.

11.3.2. Reaction Coordinates

A reaction coordinate is defined in the `colvar` namelist in a separate file. (see Fig. 11.1). This section must contain a `cv_type` keyword along with a value of type `STRING` and a list of integers `cv_i` (the number of integers is defined by `cv_ni`). For some types of reaction coordinates the `colvar` section must also contain a list of real numbers, `cv_r`, whose length is defined by `cv_nr`.

The following reaction coordinates (specfied by `cv_type`) are currently implemented:

DISTANCE: distance (in Å) between two atoms whose indexes are read from the list `cv_i`.

COM_DISTANCE: distance between the center of mass of two atom groups. The `cv_i` list is interpreted as a list of indexes of participating atoms. Zeros separate the groups, the last zero is optional. eg: `cv_i = a1,...,aN,0,b1,...,bM,0`.

DF_COM_DISTANCE: difference of distances between the center of mass of first two atom groups and second two atom groups. The `cv_i` list is interpreted as a list of indexes of participating atoms. Zeros separate the groups, the last zero is optional. eg: `cv_i = a1,...,aN,0,b1,...bM,0,c1,...cL,0,d1,...,dK,0`, `DF_COM_DISTANCE` is `COM_DISTANCE(a1,...,aN,0,b1,...,bM) - COM_DISTANCE(c1,...cL,0,d1,...,dK)`.

LCOD: linear combination of distances (in Å) between pairs of atoms listed in `cv_i` with the coefficients read from `cv_r` list. For example, `i = 1,2,3,4` and `r = 1.0,-1.0` define the difference between 1-2 and 3-4 distances, *i.e.* LCOD = r1*distance(1, 2) + r2*distance(3,4).

ANGLE: angle (in radians) between the lines joining atoms with indexes `i1` and `i2` and atoms with indexes `i2` and `i3`.

COM_ANGLE: angle (in radians) formed by the center of mass of three atom groups. The `cv_i` list is interpreted as a list of indexes of participating atoms. Zeros separate the groups, the last zero is optional. eg: `cv_i = a1,...,aN,0,b1,...bM,0,c1,...,cK,0`.

TORSION: dihedral angle (in radians) formed by atoms with indexes `i1`, `i2`, `i3` and `i4`.

COM_TORSION: dihedral angle (in radians) formed by the center of mass of four atom groups. The `cv_i` list is interpreted as a list of indexes of participating atoms. Zeros separate the groups, the last zero is optional. eg: `cv_i = a1,...,aN,0,b1,...bM,0,c1,...,CK,0,d1,...,dL,0`.

COS_OF_DIHEDRAL: sum of cosines of dihedral angles formed by atoms with indexes in the list `cv_i`. The number of atoms must be a multiple of four.

SIN_OF_DIHEDRAL: sum of sines of dihedral angles formed by atoms with indexes in the list `cv_i`. The number of atoms must be a multiple of four.

PAIR_DIHEDRAL: sum of cosines of a list of angles each formed by summing two neighboring dihedral angles from a list formed by atoms with indices `cv_i`. The number of atoms must be a multiple of four. For a list of dihedral angles such as $\{\alpha_1, \ldots, \alpha_N\}$, PAIR_DIHEDRAL is $\sum_{i=1}^{N-1} cos(\alpha_i + \alpha_{i+1})$ which ranges between $-N+1$ and $N-1$.

PATTERN_DIHEDRAL: a particular pattern-recognizing function defined on a list of dihedral angles formed by atoms with indices `cv_i`. The number of atoms must be a multiple of four. The definition is particularly relevant for the dihedral angles with a binary-like behavior of being either around 0 or 180 (e.g., ω backbone dihedral angle). For a list of dihedral angles such as $\{\alpha_1, \ldots, \alpha_N\}$, PATTERN_DIHEDRAL is $\sum_{i=1}^{N} cos^2(\alpha_i/2)2^{i-1}$ which ranges between 0 and $2^N - 1$.

R_OF_GYRATION: radius of gyration (in Å) of atoms with indexes given in the `cv_i` list (mass weighted).

MULTI_RMSD: `RMS` (in Å, mass weighted) of `RMSD`s of several groups of atoms w.r.t. reference positions provided in the `cv_r` list. The `cv_i` list is interpreted as a list of indexes of participating atoms. Zeros separate the groups. An atom may enter several groups simultaneously. The `cv_r` array is expected to contain the reference positions (without zero sentinels). The implementation uses the method (and the code) introduced in Ref. [249]. An example of variable of this type is presented in Fig. 11.2. Two groups are defined here: one comprises the atoms with indexes 1, 2, 3, 4 (line 3 in Fig. 11.2, numbers prior to the first zero) and another one of atoms with indexes 3, 4, 5. The code will first compute the (mass weighted) `RMSD` (R_1) of atoms belonging to the first group w.r.t. reference coordinates provided in the `cv_r` array (first $12 = 4 \times 3$ real numbers of it; lines 4, 5, 6, 7 in Fig. 11.2). Next, the (mass weighted) `RMSD` (R_2) of atoms of the second

```
&colvar
   cv_type = 'MULTI_RMSD'
   cv_ni = 9, cv_nr = 21,
   cv_i = 1, 2, 3, 4, 0, 3, 4, 5, 0 ! the last zero is optional
   cv_r = 1.0, 1.0, 1.0, ! group #1, atom 1
        2.0, 2.0, 2.0, ! group #1, atom 2
        3.0, 3.0, 3.0, ! group #1, atom 3
        4.0, 4.0, 4.0, ! group #1, atom 4
       23.0, 23.0, 23.0, ! group #2, atom 3
          4.0, 4.0, 4.0, ! group #2, atom 4
          5.0, 5.0, 5.0 ! group #2, atom 5
/
```

Figure 11.2.: *An example of* `MULTI_RMSD` *variable definition.*

group w.r.t. the corresponding reference coordinates (last $9 = 3 \times 3$ elements of the `cv_r` array in Fig. 11.2) will be computed. Finally, the code will compute the value of the variable as follows:

$$\text{value} = \sqrt{\frac{M_1}{M_1+M_2}R_1^2 + \frac{M_2}{M_1+M_2}R_2^2},$$

where M_1 and M_2 are the total masses of atoms in the corresponding groups.

N_OF_BONDS:

$$\text{value} = \sum_p \frac{1-\left(r_p \Big/ r_0\right)^6}{1-\left(r_p \Big/ r_0\right)^{12}},$$

where the sum runs over pairs of atoms p, r_p denotes distance between the atoms of pair p and r_0 is a parameter measured in Å. The `cv_r` array must contain exactly one element that is interpreted as r_0. The `cv_i` array is expected to contain pairs of indexes of participating atoms. For example, if 1 and 2 are the indexes of Oxygen atoms and 3, 4, 5 are the indexes of Hydrogen atoms and one intents to count all possible O-H bonds, the `cv_i` list must be (1, 3, 1, 4, 1, 5, 2, 3, 2, 4, 2, 5), that is, it must explicitly list all the pairs to be counted.

HANDEDNESS:

$$\text{value} = \sum_a \frac{\mathbf{u}_{a,3} \cdot [\mathbf{u}_{a,1} \times \mathbf{u}_{a,2}]}{|\mathbf{u}_{a,1}|\,|\mathbf{u}_{a,2}|\,|\mathbf{u}_{a,3}|},$$

where

$$\begin{aligned}
\mathbf{u}_{a,1} &= \mathbf{r}_{a+1} - \mathbf{r}_a \\
\mathbf{u}_{a,2} &= \mathbf{r}_{a+3} - \mathbf{r}_{a+2} \\
\mathbf{u}_{a,3} &= (1-w)(\mathbf{r}_{a+2} - \mathbf{r}_{a+1}) + w(\mathbf{r}_{a+3} - \mathbf{r}_a),
\end{aligned}$$

and $\mathbf{r}_a$ denote the positions of participating atoms. The `cv_i` array is supposed to contain indexes of the atoms and the `cv_r` array may provide the value of w ($0 \le w \le 1$, the default is zero).

N_OF_STRUCTURES:

$$\text{value} = \sum_g \frac{1-\left(R_g \Big/ R_{0,g}\right)^6}{1-\left(R_g \Big/ R_{0,g}\right)^{12}},$$

where the sum runs over groups of atoms, R_g denotes the `RMSD` of the group g w.r.t. some reference coordinates and $R_{0,g}$ are positive parameters measured in Å. The `cv_i` array is expected to contain indexes of

```
&colvar
   cv_type = 'N_OF_STRUCTURES'
   cv_ni = 9, cv_nr = 23,
   cv_i = 1, 2, 3, 4, 0, 3, 4, 5, 0 ! the last zero is optional
   cv_r = 1.0, 1.0, 1.0, ! group #1, atom 1
        2.0, 2.0, 2.0, ! group #1, atom 2
        3.0, 3.0, 3.0, ! group #1, atom 3
        4.0, 4.0, 4.0, ! group #1, atom 4
        1.0,           ! R0 for group #1
     23.0, 23.0, 23.0, ! group #2, atom 3
        4.0, 4.0, 4.0, ! group #2, atom 4
        5.0, 5.0, 5.0, ! group #2, atom 5
        2.0           ! R0 for group #2
/
```

Figure 11.3.: *An example of* `N_OF_STRUCTURES` *variable.*

participating atoms with zeros separating different groups. The elements of the `cv_r` array are interpreted as the reference coordinates of the first group followed by their corresponding R_0; then followed by the reference coordinates of the atoms of the second group, followed by the second R_0, and so forth. To make the presentation clearer, let us consider the example presented in Fig. 11.3. The atomic groups and reference coordinates are the same as the ones shown in Fig. 11.2. Lines 7 and 11 in Fig. 11.3 contain additional entries that set the values of the threshold distances $R_{0,1}$ and $R_{0,2}$. To compute the variable, the code first computes the mass weighted `RMSD` values R_1 and R_2 for both groups –much like in the `MULTI_RMSD` case– and then combines those in a manner similar to that used in the `N_OF_BONDS` variable.

$$\text{value} = \frac{1-\left(R_1\Big/R_{0,1}\right)^6}{1-\left(R_1\Big/R_{0,1}\right)^{12}} + \frac{1-\left(R_2\Big/R_{0,2}\right)^6}{1-\left(R_2\Big/R_{0,2}\right)^{12}}\,.$$

In other words, the variable "counts" the number of structures that match (stay close in `RMSD` sense) with the reference structures.

QUATERNIONS: Describing large-scale atomistic conformational changes in biomolecular systems requires one to deal with orientational changes of atomistic domains with large numbers of atoms. While there are several ways of defining a collective variable that quantifies an orientation based conformational change, the orientation quaternion technique[250–253] has proven successful as a well-behaved, flexible method for defining system-specific CVs, specifically aimed at inducing interdomain orientational changes or restraining the orientation of certain domains. The CVs in the orientation quaternion class, are all derived from an 'optimal rotation' between a set of reference coordinates $\mathbf{X}_k$ ($1 \leq k \leq N$; where N is the number of atoms involved) and the set of target coordinates $\mathbf{Y}_k$. A 'quaternion' is introduced as a four-component vector that can be expressed as $q_0 + q_1\hat{i} + q_2\hat{j} + q_3\hat{k}$ where q_0 and $q_1\hat{i} + q_2\hat{j} + q_3\hat{k}$ are called scalar and vector parts respectively. The optimal rotation can be parametrized[1] by a unit quaternion, $\hat{q} = (q_0, q_1, q_2, q_3)$, that minimize $< ||\hat{q}\mathbf{X}_k\hat{q}^* - \mathbf{Y}_k||^2 >$ in which $< . >$ denotes an average over k,q^* is the conjugate of q and $||q||^2 = q^*q$ (see Ref.[250] for more details). The optimal rotation unit quaternion (or orientation quaternion)$\hat{q}$ can be written as $\big(cos(\theta/2), sin(\theta/2)\hat{u}\big)$, where θ is the optimal rotation angle and $\hat{u}$ is a unit vector associated with the optimal axis of rotation. To deal with large atomistic conformational changes, a set of quaternion-based CVs has implemented in AMBER20. For the details of usage, send emails to Ashkan Fakharzadeh (afakhar@ncsu.edu), Dr. Feng Pan (fpan3@ncsu.edu), and Prof. Mahmoud Moradi (moradi@uark.edu). The specific quaternion-based CVs implemented are: ORIENTATION_ANGLE, ORIENTATION_PROJ, TILT, SPINANGLE, QUATERNION0, QUATERNION1, QUATERNION2, and QUATERNION3.

[1] Assuming both sets have been already shifted to bring their barycenters to the origin (optimum translation).

```
&colvar
   cv_type = 'QUATERNION0'

   ! number of participating atoms
   cv_ni = ni

   ! index of participating atoms
   cv_i = a1, a2,..., aN

   ! AMBER coordinate/restart file to read reference coordinates
   refcrd_file = 'refcrd_file'

   ! number of references which must be 3*ni; Should not be set if
   ! refcrd_file is being used
   cv_nr = nr

   ! reference coordinates of participating atoms; Should not be set if
   ! refcrd_file is being used
   cv_r = a1x, a1y, a1z, a2x, a2y, a2z, a3x, a3y, a3z, ...

   ! an arbitrary integer between 1 to 100
   q_index = n
/
```

Figure 11.4.: *Syntax of Quaternion reaction coordinates.*

Orientation (QUATERNION0,...,QUATERNION3): These define the orientation of several atoms with respect to a set of reference coordinates in terms of a unit quaternion vector $\hat{q} = (q_0, q_1, q_2, q_3)$ according to the method introduced in Ref.[250, 251]. These variables return the best-fit rotation, also used in best-fit RMSD calculation procedures, to superimpose the coordinates $\mathbf{X}$ onto a set of reference coordinates $\mathbf{X}_0$. The unit quaternion $\hat{q} = (q_0, q_1, q_2, q_3)$ can be written as $\big(cos(\theta/2), sin(\theta/2)\hat{u}\big)$, where θ is the rotation angle and $\hat{u}$ is a unit vector associated with the axis of rotation; for example, a rotation of 90° around the z axis $(0,0,1)$ is expressed as$(cos(90^\circ/2), 0.0, 0.0, sin(90^\circ/2)) = (\sqrt{2}/2, 0, 0, \sqrt{2}/2)$. The components of the unit quaternion (q_0, q_1, q_2, q_3) were implemented separately as QUATERNION0, QUATERNION1, QUATERNION2, and QUATERNION3 CVs. To find the orientation, all four CVs QUATERNION0,...,QUATERNION3 are being used. To calculate the quaternion CVs one needs to specify a list of participating atoms and also their reference coordinates. The reference coordinates may be passed to AMBER either via direct specification inside the CV call, or by passing the name of a reference coordinates file. It is recommended that if the set of participating atoms is small (say no larger than 15), then these are specified directly inside the CV call. Otherwise, the passing of information via filename is recommended since these lists may contain hundreds if not thousands of atoms. Relevant parameters pertaining to the input of this information are: *cv_ni* represents the number of participating atoms; *cv_i* represents the list of the indices of all participating atoms; *cv_r* represents the reference coordinates (when passed directly) and $refcrd\_file$ is the filename for the reference coordinates when they are to be read from file. The file $refcrd\_file$ should be an AMBER coordinates/restart file containing coordinates, velocities, etc. of all atoms. The list participating atoms, *cv_i*, and their reference coordinates (*cv_r* and or $refcrd\_file$) must be the same for all QUATERNION0,...,QUATERNION3. The CVs are linked together using an attribute*'q_index'*. The*'q_index'* accepts an integer between $1, ..., 100$, where its default value is one. The Fig. 11.4 is an example of Quaternion CVs syntax. An example this type of CVs is presented in the Fig. 11.5. Two set of orientations are defined here: each set consists of QUATERNION0,.., QUATERNION3. The first set comprises 18 atoms with indexes $11, 41, 48, 74, 104, \ldots$ and another one of 24 atoms with indexes $12, 16, 46, 55, 75, \ldots$. A file, 'inpcrd' is used as an AMBER coordinate/restart file to read reference coordinates. There is no need to set *'q_index'* for the first four quaternions since the default value is one, but it is set to be 2 for all quaternion CVs in the second set

to link and normalize them. The returned value of each QUATERNION0,...,QUATERNION3 CVs is the corresponding component of the unit orientation vector $\hat{q} = (q_0, q_1, q_2, q_3)$.

ORIENTATION_ANGLE: The angle of rotation $\theta = 2cos^{-1}(q_0)$ between the current and the reference positions. This angle is between 0° to 180°. The *cv_i* list is interpreted as a list of indexes of participating atoms.

$$\textit{orientation angle: } \theta = 2cos^{-1}(q_0)$$

ORIENTATION_PROJ: The cosine of the angle of rotation θ between the current and the reference positions. While ORIENTATION_ANGLE diverges near $\theta = 0$, because of $\nabla_X \theta$, ORIENTATION_PROJ might be used instead to apply forces. The range of ORIENTATION_PROJ is $[-1, 1]$. The *cv_i* array is supposed to contain indexes of the atoms.

$$\textit{orientation proj: } 2q_0^2 - 1$$

SPINANGLE: Angle of rotation ϕ around a given unit axis $\hat{\mathbf{e}}$. The axis $\hat{\mathbf{e}}$ is being used to decompose a complete orientation rotation in two sub-rotations, spin ϕ and tilt ω. An advantage of this decomposition is ϕ and ω have the same values, regardless of which one is applied first (in comparison to Euler angles methods). The participating atoms with indexes are given in the *cv_i*. The *'axis'* must provide three components of the axis[2] $\hat{\mathbf{e}}$ in A°. The default axis of rotation is $(0.0, 0.0, 1.0)$. The range of SPINANGLE is between $[-180 : 180]$ degrees. The reference coordinates are specified either via *cv_r* or *refcrd_file*.

$$\textit{spin angle: } \phi = 2tan^{-1}(\mathbf{q}.\mathbf{e}/q_0)$$

where $\mathbf{q}$ is the vector part of quaternion, namely (q_1, q_2, q_3). An example of SPINANGLE cv is presented in Fig. 11.6. The same atoms as example one are used, but the axis of rotation is set to be 'x-axis'. The reference coordinates are given by *cv_nr*, *cv_r* options.

TILT: Cosine of the rotation orthogonal to an unit given axis. The tilt angle ω, shows a rotation away from the direction $\hat{\mathbf{e}}$. The tilt combined with the 'spin' sub-rotation provides the complete orientation rotation of a group of atoms. Similar to ORIENTATION_PROJ, to avoid the discontinuity around 0° and 180°, the cosine of the tilt is implemented instead of the tilt angle itself, so that derivatives are continuous almost everywhere. The *cv_i* and *'axis'* are the participating atoms with indexes and the given axis, respectively. The reference coordinates are specified either via *cv_r* or *refcrd_file*. The value of TILT is between -1 to 1, where the value 1 represents an orientation fully parallel to $\hat{\mathbf{e}}$ ($\omega = 0^\circ$), and the value -1 represents an anti-parallel orientation.

$$\textit{tilt: } t = cos(\omega) = 2\Big(\frac{q_0}{cos\big(\frac{tan^{-1}\mathbf{q}.\mathbf{e}}{q_0}\big)}\Big)^2 - 1$$

11.3.3. Steered Molecular Dynamics

The `&smd` namelist, if present in the `MDIN` file, activates the steered `MD` code (the method itself is extensively described in the literature: see for example Ref. [254] and references therein). The prefix NFE appears in several switches to do with steered MD: this stands for "Non-equilibrium Free Energy".

The following is recognized within the `&smd` namelist:

`output_file` sets the output file name. Default is '`nfe-smd.txt`'.

`output_freq` sets the output frequency (in `MD` steps). Default is 50.

`cv_file` sets the collective variable file name. Default is '`nfe-smd-cv`'.

There must be at least one reaction coordinate defined (that is, there must be at least one `&colvar` namelist in the `cv_file`). The steered `MD` code requires that additional entries be present in the `&colvar` namelist:

[2] The axis is from the origin$(0.0, 0.0, 0.0)$ to that point.

path the steering path whose elements must be real numbers. The path must include at least two elements. The upper limit on the number of entries is 20000. The elements define Catmull-Rom spline used for steering.

npath sets the number of elements in path. Default is 0.

path_mode The way steering paths are constructed. There are two modes available. In SPLINE mode (default) the path is approximated by a spline that passes through the given points; in LINES mode the path is represented by the line segments joining the control points.

harm specifies the harmonic constant. If a single number is provided, e.g., harm = 10.0, then it is constant throughout the run. If two or more numbers are provided, e.g., harm = 10.0, 20.0, then the harmonic constant follows a Catmull-Rom spline built upon the provided values.

nharm sets the number of elements in harm. Default is 0.

harm_mode The way harmonical paths are constructed, similar with path_mode.

An example of MDIN file and CV.IN file for steered MD is shown in Fig. 11.7. The reaction coordinate is defined in cv.in. The spring constant is set constant throughout the run and the steering path is configured from 5.0 to 3.0. The values of the reaction coordinate, harmonic constant and the work performed on the system are requested to be dumped to the smd.txt file every 50 MD steps.

11.3.4. Umbrella sampling

To activate the umbrella sampling code, the &pmd namlist must be present in the MDIN file.&pmd is currently available to both SANDER and PMEMD, and also can be fully applied in GPU accelerated PMEMD. The output_file, output_freq and cv_file entries are recognized just as in the steered MD case presented earlier. The cv_file must contain at least one &colvar namelist section. For umbrella sampling, the &colvar section(s) must contain two additional entries:

anchor_position: this consists of four real numbers $(r1, r2, r3, r4)$ that determine the rectangle of the umbrella (harmonic) potential. The default value is that all of the $r's$ is set to zero.

anchor_strength: two non-negative real numbers $(k1, k2)$ that set the harmonic constant for the umbrella (harmonic) potential. The default value is zero.

The umbrella (harmonic) potential U is determined by (supposing R is the value of reaction coordinate)

- $U = k1 * (r1 - r2) * R \quad (R \leqslant r1)$
- $U = 0.5 * k1 * (R - r2)^2 \quad (r1 < R \leqslant r2)$
- $U = 0 \quad (r2 < R \leqslant r3)$
- $U = 0.5 * k2 * (R - r3)^2 \quad (r3 < R \leqslant r4)$
- $U = k2 * (r4 - r3) * R \quad (R > r4)$

A plot of the umbrella potential is shown below

eg1: if $r2 = r3$, $r1 \ll r2$ and $r4 \gg r3$, then the generated U is simply the traditional harmonic potential.

eg2: if $r1$ is slightly less than $r2$ and $r4$ is slightly larger than $r3$, also with very large $k1$, $k2$, the reaction coordinate is restrained in the range ($r2$, $r3$) with no potential added.

An example of an `MDIN` file and `CV.IN` file for an umbrella sampling simulation is shown in Fig. 11.8. The first reaction coordinate here is the angle formed by the lines joining the 5th with 9th and 9th with 15th atoms. It is to be harmonically restrained near 1.0 *rad* (`anchor_position` entry) using the spring of strength 10.0 $kcal/mol/rad^2$ (`anchor_strength` entry). The second reaction coordinate requested in Fig. 11.8 is a dihedral angle (`type = 'TORSION'`) formed by the 1st, 2nd, 3rd and 4th atoms (the `cv_i` array). It is to be restrained near zero with strength 23.8 $kcal/mol/rad^2$. The values of the reaction coordinate(s) are to be dumped every 50 `MD` steps to the `pmd.txt` file. Another example of restraining reaction coordinate in a specific range is shown in Fig. 11.9. The reaction coordinates here are φ and ψ angles of dialanine. φ is restrained between -2.0 *rad* and 2.0 *rad*, ψ is restrained between -1.8 *rad* and 1.8 *rad*.

The `NFE` implementation of umbrella sampling works correctly with the Amber standard replica-exchange MD described earlier in this manual (compatible with different types of REMD for different values of `-rem` flag in both SANDER and PMEMD). For example, the typical umbrella sampling with Hamiltonian Replica Exchange can be performed by setting `-rem` to 3. In this case, both `anchor_position` and `anchor_strength` may be different for different temperatures. Even the number and type of reaction coordinate(s) could vary for different replicas. The output files (set by the `output_file` keyword on a per-replica basis) are `MDIN`-bound, consistent with `-rem`.

11.3.5. Adaptively Biased Molecular Dynamics

The implementation has a very simple and intuitive interface: the code is activated if either an `&abmd` (both SANDER and PMEMD) or an `&bbmd` (both SANDER and PMEMD) namelist is present in the `MDIN` file (the difference between those "flavors" is purely technical and will become clear later). Unlike in the `&smd` and `&pmd` cases, the dimensionality of a reaction coordinate (the number of `&colvar` namelists in the `cv_file`) cannot exceed five (though three is already hardly useful due to statistical reasons).

As previously noted, in order to activate the ABMD and related algorithm, the variable ***infe*** in &cntrl must be set to unity (*i.e.* ***infe*** = 1; default value ***infe*** = 0).

In addition to the `cv_file` entry, the following entries are recognized within the `&abmd` (or `&bbmd`) namelist:

mode sets the execution mode. There are three modes available: `'ANALYSIS' | 'UMBRELLA' | 'FLOODING'`. In `ANALYSIS` mode the dynamics is not altered. The only effect of this mode is that the value(s) of the reaction coordinate(s) is(are) dumped every `monitor_freq` to `monitor_file`. In `UMBRELLA` mode, biasing potential from the `umbrella_file` is used to bias the simulation ($\tau_F = \infty$, biasing potential does not change). In `FLOODING` mode the adaptive biasing is enabled.

monitor_file sets the name of the file to which value(s) of reaction coordinate(s) (along with the magnitude of biasing potential in `FLOODING` mode) are dumped.

monitor_freq the frequency of the output to the `monitor_file`.

timescale τ_F, the flooding timescale in picoseconds (only required in `FLOODING` mode).

umbrella_file biasing potential file name (the file must exist for the `UMBRELLA` mode).

In `FLOODING` mode, the following two entries are optional:

snapshots_basename sets the name of the file to which the biasing potential is dumped during the simulation for snapshot.

snapshots_freq the frequency of dumping snapshot biasing potential (in MD steps). If `snapshots_freq` is not specified, the snapshot biasing potential will not be dumped.

and the &colvar namelist for &abmd method must also contain the following entries:

cv_min smallest desired value of the reaction coordinate (required, unless the reaction coordinate is limited from below).

cv_max largest desired value of the reaction coordinate (required, unless the reaction coordinate is limited from above).

resolution the "spatial" resolution for the reaction coordinate.

To access the biasing potential files created in the course of FLOODING simulations, the nfe-umbrella-slice utility is provided (it prints a short description of itself if invoked with --help option).

The multiple-walker selection algorithm can improve the simulation by resampling between different walkers. The well-tempered ABMD can lead to a smoother convergence to the desired free energy. These two algorithm are implemented to SANDER and PMEMD from Amber16 onwards.

The multiple-walker selection algorithm currently works with &abmd only. The algorithm should be used only within the multiple-walker scheme (*i.e.*, when command-line **-rem** flag is set to zero). The following entries are recognized regarding with the selection algorithm (selection algorithm can work with FLOODING and UMBRELLA mode):

selection_freq positive integer number that sets the frequency of the resampling algorithm (in MD steps). If selection_freq is not specified, the selection algorithm will not be used.

selection_constant positive real number that sets the parameter *C*. if selection_freq is specified, specifying selection_constant is required (no default value). Parameter *C* is to determine how strong the selection mechanism is. If *C* is too large, all the walkers will be replaced with the most dominant one. If *C* is too small, there will be no killing/duplicating of walkers.

selection_epsilon positive real number (typically less than unity) that sets the stopping criterion parameter ε . Parameter ε determines the threshold for stopping the selection algorithm. If selection_epsilon is not specified, there will be no stop to the algorithm. If selection_epsilon is equal or larger than one, the algorithm will be stopped after the first attempt.

The well-tempered flavor can be used within either &abmd or &bbmd namelist. There are two entries relevant to the well-tempered feature:

wt_temperature positive real number that sets the pseudo-temperature T'. If this flag is not specified, conventional ABMD will be used (*i.e.*, T' $\rightarrow \infty$ or β' $\rightarrow 0$). The smaller the T'; the smoother/slower the convergence.

wt_umbrella_file the file name of true biasing potential after modification by 1 + (T /T') in which T is the reference temperature of the system (temp0).

An example MDIN file and CV.IN file for the &abmd flavor of ABMD is shown in the Fig. 11.10.

In this example, the reaction coordinate is defined as the distance between the 5th and 9th atoms (more than one reaction coordinates might be requested by mere inclusion of additional &colvar subsections). The mode is set to FLOODING thus enabling the adaptive biasing with flooding timescale $\tau_F = 100ps$. The region of interest of the reaction coordinate is specified to be between -1 Å and 10 Å and the resolution is set to 0.5 Å. The lower bound (-1 Å) could have been omitted for DISTANCE variable: the default value of zero would be used in such case. The code will try to load the biasing potential from the umbrella.nc file and use it as the value of $U(t|\xi)$ at the beginning of the run. The biasing potential built in the course of simulation will be saved to the same file (umbrella.nc) every time the RESTRT file is written. The selection algorithm is used with the frequency of selection defined as 10000 MD steps and selection constant defined as 0.001. The well-tempered algorithm is also used, with the pseudo-temperature defined as 10000 K in and the true biasing potential will be dumped as wt_umbrella.nc file. The nfe-umbrella-slice utility can then be used to access its content. An MDIN

file for the follow up biased run at equilibrium would look much like the one shown in the Fig. 11.10, but with mode changed from FLOODING to UMBRELLA.

Driven ABMD can be performed using &smd block (for the SMD part of the algorithm) along with &abmd block (for the ABMD part of the algorithm). There is no additional flag for the &smd block relevant to the algorithm; however, there are two additional flags to ABMD relevant to the "driven" feature.

driven_weight string that sets the weighting scheme. The default option (i.e., not using the flag) is NONE which indicates no reweighting is used (NOT RECOMMENDED if SMD is performed along ABMD). Other options include CONSTANT and PULLING for constant and pulling reweighting protocols.

driven_cutoff positive real number that sets a cutoff for work for applying the reweighting algorithm (default: 0.0). If the work (accumulated or transferred depending on the scheme) at any given time is lower than the cutoff, no reweighting is done at that particular time. If the cutoff is too small, it may result in instability of the algorithm.

For both SANDER and PMEMD since Amber18, the &abmd code works correctly with Amber replica-exchange similar with &pmd (that is, for -rem flag set to different values). If-rem is set to 3, ABMD with replica-exchange is carried out. In such case different replicas can have different temperatures, collective variables and even different mode.The monitor and umbrella files are MDIN-bound. If number of sander groups exceeds one (the flag -ng is greater than one) and -rem flag is set to zero, the code runs *multiple walkers* ABMD. In both cases the number and type(s) of variable(s) must be the same across all replicas.

Finally, the &bbmd flavor allows one to run replica-exchange (AB)MD with different reaction coordinates and different modes (ANALYSIS, UMBRELLA or FLOODING) in different replicas (along with different temperatures, if desired). This module is outdated since &abmd has been compatible with -rem equals 3. The only advantage of &bbmd is that the number of replicas can be odd numbers if desired by runs, while this cannot be achieved in any -rem types. To applying &bbmd module, the -rem flag must be set to zero and the &bbmd sections must be present in all MDIN files. The MDIN file for the replica of rank zero (first line in the group file) is expected to contain additional information as compared to &abmd case (an example of such MDIN file for replica zero is shown in Fig. 11.11). The MDIN files for all other replicas except zero do not need any additional information, and therefore take the same form as in the &abmd flavor (except that the namelist is changed from &abmd to &bbmd, thus activating a slightly different code path). Each MDIN file may define its own reaction coordinates, have different mode and temperature if desired.

Within the first replica &bbmd namelist the following additional entries are recognized:

exchange_freq number of MD steps between the exchange attempts.

exchange_log_file the name of the file to which exchange statistics is to be reported.

exchange_log_freq frequency of exchange_log_file updates.

mt19937_seed seed for the random generator (Mersenne twister [255]).

mt19937_file the name of the file to which the state of the Mersenne twister is dumped periodically (for restarts).

The MDOUT, MDCRD, RESTRT, umbrella_file and monitor_file files are MDIN-bound in course of the bbmd-enabled run. An example that uses this kind of replica exchange is presented in Ref.228.

11.3.6. Swarms-of-Trajectories String Method

ABMD is a robust method for calculating free energy landscapes as a function of a small number of collective variables. Since the required computer time grows enormously with the number of collective variables, ABMD is best for exploring one- or two-dimensional phase spaces. However, rather than calculating full n-dimensional free energy maps, it is often fruitful to focus on the so-called Minimum Free Energy Path (MFEP) which the system is likely to take when transitioning between two minima. Calculating a MFEP in a complicated phase space is often

difficult, and so-called "string methods"[256][246] represent one of the best approaches for finding the MFEP. Since sampling in string methods is essentially limited to regions around the MFEP, the cost of the method scales linearly with the length of the string or path, but only weakly on the number of collective variables. This results in considerable computational savings since the full free energy landscape is not calculated.

The swarms-of-trajectories string method (STSM)[246] is one of the most popular versions of the string method and has been implemented here by Dr. Moradi (moradi@uark.edu). The module is available in both SANDER and PMEMD from Amber18 onwards. It is a path-finding algorithm that refines a putative transition pathway iteratively until the path is deemed to have been converged. The string is defined by a number of nodes or images parameterized in a high-dimensional space of collective variables, whose position is updated iteratively. The center of each image is first used as a restraining center to generate representative conformations at the current center before allowing of a small change in this center for the next iteration. The change in the center of each image is estimated by averaging over the drifts of a swarm of short unbiased trajectories all starting at the current image position (generated using the constrained simulations. Thus, each iteration consists of a series of restrained and free simulations. In the current serial version of the code, these simulations are performed independently. In parallel versions -- which are more efficient -- a very large number of replicas is required which are run in parallel; this method is particularly efficient on large supercomputers.

To invoke the swarms-of-trajectories string method, the `&stsm` must be invoked in the `MDIN` file. For a string consisting of N_s nodes each requiring M copies $N_s \times M$ replicas will be required. The parallel implemention of the STSM method is based on iterative restrained and free MD simulations followed by a reparametrization of the image centers defined in a multidimensional collective variable space ξ. For the i^{th} iteration, first M copies of the n^{th} image are generated around the old center ξ_n^{i-1} by MD equilibration lasting τ_E timesteps. The generated M copies of the n^{th} image are expected to be close to ξ_n^{i-1} for time τ_E, assuming that the invoked harmonic constant k for the restraining potential is large enough. The parameters τ_E and k thus need to be appropriately chosen in order to ensure that all copies of each image will be close to the image center. The restraint is then released, and each copy (swarm) is allowed to drift for τ_R timesteps. The newly shifted center ξ_n^i for the n^{th} image is then determined by averaging over all drifted copies $\xi_{n,m}^t$ at time $t = \tau_E + \tau_R$. The resulting string of images is then smoothed using a linear interpolation protocol. A smoothing parameter ε with $0 \leq \varepsilon \leq 1$ determines the smoothness of the curve; it is recommended that ε be of the order of $1/(N_s - 1)$. The last setep is a reparameterization, which gain follows a linear interpolation protocol in order to generate N_s equidistant centers along the string. The two key parameters of the method are M and τ_R. Generally, the large the M and the shorter τ_R, the smoother (but slower) the evolution of the MFEP will be. These variables must be optimized empirically, but typically 10 - 30 copies and 5 - 20 ps are reasonable values. It is often advantageous to set $\tau_E = \tau_R$.

An improved sequential repeat version of the algorithm has also been implemented, which avoids the large number of copies and does not require a large number of processors to run. Here a new variable N_R is introduced, as the number of repeat runs for each replica. Now for each copy, it will run around the old center ξ_n^{i-1} for N_R times sequentially. And each repeat run can be equally considered as a parallel run of a new copy around the old center. Namely, the new shifted center will be determined by averaging on $N_R \times M$ copies. So the number of processors needed will be reduced to $1/N_R$, while the running time will be multipled by N_R.

The following is recognized within the &stsm namelist:

`image` positive integer number that sets the image id (between 1 and N). Default is 0.

`repeats` positive integer number that sets the number of repeat runs, should be the same for each image and each copy. Equal to parallel implementation when not set. Default is 1.

`equilibration` non-negative integer number that sets the number of MD steps specified for biased equlibration (restraining) at each iteration. Default is 0.

`release` Number of MD steps specified for the release (drift) at each iteration. Note: the total number of iterations is determined by the total simulation time (`nstlim` flag in `mdin` file) divided by total time for each iteration given by `equilibration+release`.

`smoothing` positive number that sets the smoothing parameter for reparametrization (between 0 and 1). Smoothing parameter should be, preferably, on the order of $1/(N_s - 1)$. If this flag is not used, no smoothing will be performed.

report_centers a string that determines if drifted and/or smoothed and/or reparametrized centers will be reported. The default value is NONE and other available options include ALL,DRIFT,SMOOTHED, REPARAMETRIZED, NO_DRIFT, NO_SMOOTHED, NO_REPARAMETRIZED.

The output_file, output_freq and cv_file entries are recognized just as &smd and &pmd, the information of reaction coordinates will be read from cv_file. The number of collective variables can not exceed five. (here be attention that the anchor_postion and anchor_strength will be defined using the traditional harmonical potential, different with &pmd!). An example of MDIN file and CV.IN file for STSM in parallel case is shown in Fig. 11.12. Here we run 8 images along the path, with I defining the image ID. We run 980 MD steps for equlibration and 20 MD steps release at each iteration, so there are totally 1000 MD steps for each iteration. With nstlim set to 10000, 10 iterations will be carried out. The smoothing parameter is set to 0.1 and all the centers will be reported. For each image, 16 copies will be run in parallel, with J defining the copy ID. The evolution of reaction coordinate will recorded in the file stsm.00I.J.txt. For this run, at least $128(8 \times 16)$ processors are needed. Another example of MDIN file of equivalent sampling level in sequential case is shown in Fig. 11.13. Here we still have 8 images to run. We set the number of repeats to be 16, namely 16 repeat runs for each image to get the new drifted center. Therefore, 16000 MD steps are needed for one iteration, and so we set nstlim to 160000 to complete 10 iterations. For this run, 8 processors are needed at least.

Part of sample MDOUT file is shown in Fig. 11.14. The restoring restraint part will be only in sequential run, since the restraint needs to be restored after each repeat. The values of reaction coordinates before reporting centers are the averaged value over repeats for this copy and the instantaneous value. All the centers will be reported only in the MDOUT file of first copy of first image. The drifted centers are the averaged value over copies, and also the smoothed ans reparametrized centers can be reported. Always the reparametrized centers will be extracted to draw the MFEP in the phase space.

11.3.7. Post-processing of biasing potential

When you get the biasing potential (*.nc file), you can always use the nfe-umbrella-slice utility to access its content and get a friendly-written ASCII file from which one can obtain the free energy map. The output is the free energy value, which is the opposite of the biasing potential ($f = -U$ (*units* $kcal/mol$)). The nfe-umbrella-slice utility has been included in AmberTools.

Usage: nfe-umbrella-slice [options] bias_potential.nc

Options:

-h,--help Print out a usage summary

-p,--pretend Only print out the basic properties of source without biasing potential data (off by default)

-g,--gradient Print out the gradients (off by default)

-r,--reset Set the value of minimum to zero (off by default)

-t,--translate Translate the numerical value of biasing potential by a real number (0 by default)

-d,--dimensions Set the way of slice in different dimensions. The format is "D1:D2:...:Dn", where n is the number of dimensions. Each D can only be set with one number or three numbers separated by commas. If only one number is set, the variable will be fixed at that value. If three numbers are set, the first two define the boundary of the slice and the last one defines the number of points.

Example:

- nfe-umbrella-slice -r -d "-5.0,5.0,50" 1d-bias.nc > FE.dat

This processes the 1-dimensional biasing potential file 1d-bias.nc and prints out the results to FE.dat. The minimum of free energy will be set to zero. The variable will be taken from -5.0 to 5.0 using 50 points.

- `nfe-umbrella-slice -g -t 50.0 -d "1.0:-2.0,2.0,20" 2d-bias.nc > FE.dat`

This processes the 2-dimensional biasing potential file `2d-bias.nc` and prints out the results to `FE.dat`. All the free energy will be incremented by a constant 50.0. The gradients in both dimensions will be printed out. For the first dimension, the variable will be fixed at 1.0; for the second dimension, the variable will be taken from -2.0 to 2.0 using 20 points.

- `nfe-umbrella-slice wt_umbrella.nc > wt_FE.dat`

This processes the biasing potential after WT-ABMD and prints out the results to `wt_FE.dat`. The default dimensional information is obtained and used by the program from the biasing potential file.

```
&colvar
   cv_type = 'QUATERNION0'
   cv_ni = 18,
   cv_i = 11 , 41 , 48 , 74 , 104 , 111 , 137 , 167 , 174 , 199 , 229 , 236, 262 ,
          292 , 299 , 325 , 355 , 362 ,
   refcrd_file = 'inpcrd'
/
&colvar
   cv_type = 'QUATERNION1'
   cv_ni = 18,
   cv_i = 11 , 41 , 48 , 74 , 104 , 111 , 137 , 167 , 174 , 199 , 229 , 236, 262 ,
          292 , 299 , 325 , 355 , 362 ,
   refcrd_file = 'inpcrd'
/
&colvar
   cv_type = 'QUATERNION2'
   cv_ni = 18,
   cv_i = 11 , 41 , 48 , 74 , 104 , 111 , 137 , 167 , 174 , 199 , 229 , 236, 262 ,
          292 , 299 , 325 , 355 , 362 ,
   refcrd_file = 'inpcrd'
/
&colvar
   cv_type = 'QUATERNION3'
   cv_ni = 18,
   cv_i = 11 , 41 , 48 , 74 , 104 , 111 , 137 , 167 , 174 , 199 , 229 , 236, 262 ,
          292 , 299 , 325 , 355 , 362 ,
   refcrd_file = 'inpcrd'
/
&colvar
   cv_type = 'QUATERNION0'
   cv_ni = 24,
   cv_i = 12 , 16 , 46 , 55 , 75 , 79 , 109 , 118 , 138 , 142 , 172 , 181 , 200 ,
          204 , 234 , 243 , 263 , 267 , 297 , 306 , 326 , 330 , 360 , 369 ,
   refcrd_file = 'inpcrd',
   q_index = 2
/
&colvar
   cv_type = 'QUATERNION1'
   cv_ni = 24,
   cv_i = 12 , 16 , 46 , 55 , 75 , 79 , 109 , 118 , 138 , 142 , 172 , 181 , 200 ,
          204 , 234 , 243 , 263 , 267 , 297 , 306 , 326 , 330 , 360 , 369 ,
   refcrd_file = 'inpcrd',
   q_index = 2
/
&colvar
   cv_type = 'QUATERNION2'
   cv_ni = 24,
   cv_i = 12 , 16 , 46 , 55 , 75 , 79 , 109 , 118 , 138 , 142 , 172 , 181 , 200 ,
          204 , 234 , 243 , 263 , 267 , 297 , 306 , 326 , 330 , 360 , 369 ,
   refcrd_file = 'inpcrd',
   q_index = 2
/
&colvar
   cv_type = 'QUATERNION3'
   cv_ni = 24,
   cv_i = 12 , 16 , 46 , 55 , 75 , 79 , 109 , 118 , 138 , 142 , 172 , 181 , 200 ,
          204 , 234 , 243 , 263 , 267 , 297 , 306 , 326 , 330 , 360 , 369 ,
   refcrd_file = 'inpcrd',
   q_index = 2
/
```


Figure 11.5.: *An example of* `Orientation` *variable.*

```
&colvar
   cv_type = 'SPINANGLE'
   cv_ni = 18, cv_nr = 54,
   cv_i = 11 , 41 , 48 , 74 , 104 , 111 , 137 , 167 , 174 , 199 , 229 ,
          236, 262 , 292 , 299 , 325 , 355 , 362 ,
   cv_r = 0.96 , -4.47 , -0.31 , 3.48 , -3.00 , 3.06 , 0.88 , 0.01 ,
          3.36 ,4.55 , -0.51 , 6.46 , 3.93 , 2.38 , 9.81 , 0.26 ,
          0.84 , 10.12 , 1.90 , 4.16 , 13.21 , -1.06 , 4.47 , 16.58 ,
         -0.71 , 0.52 , 16.88 , -0.96 , -4.47 , 17.21 , -3.48 ,
         -3.00 , 13.84 , -0.88 , 0.01 , 13.54 , -4.55 , -0.51 , 10.44 ,
         -3.93 , 2.38 , 7.09 , -0.26 , 0.84, 6.78 , -1.90 , 4.16 ,
          3.69 , 1.06 , 4.47 , 0.32 , 0.71 , 0.52 , 0.02 ,
   axis = 1.0, 0.0, 0.0
/
```

Figure 11.6.: *An example of* `SPINANGLE` *variable.*

```
title line
&cntrl
..., infe = 1
/

&smd
   output_file = 'smd.txt'
   output_freq = 50
   cv_file = 'cv.in'
/
```

```
cv_file
&colvar
  cv_type = 'DISTANCE'
  cv_ni = 2
  cv_i = 5, 9
  npath = 2, path = 5.0, 3.0, path_mode = 'LINES',
  nharm = 1, harm = 10.0
/
```

Figure 11.7.: *An example* `MDIN` *file and* `CV.IN` *file for steered* `MD`*. Only the relevant part is shown.*

```
title line
&cntrl
..., infe = 1
/

&pmd
   output_file = 'pmd.txt'
   output_freq = 50
   cv_file = 'cv.in'
/
```

```
cv_file
&colvar ! first
   cv_type = 'ANGLE'
   cv_ni = 3, cv_i = 5, 9, 15
   anchor_position = -10.0,1.0,1.0,10.0
   anchor_strength = 10.0,10.0
/
&colvar ! second
   cv_type = 'TORSION'
   cv_ni = 4, cv_i = 1, 2, 3, 4
   anchor_position = -10.0,0.0,0.0,10.0
   anchor_strength = 23.8,23.8
/
```

Figure 11.8.: *An example* `MDIN` *file and* `CV.IN` *file for umbrella sampling (only relevant part is presented in full).*

```
cv_file
&colvar ! phi
   cv_type = 'TORSION'
   cv_ni = 4, cv_i = 5, 7, 9, 15
   anchor_position = -2.05,-2.0,2.0,2.05
   anchor_strength = 500.0,500.0
/
&colvar ! psi
   cv_type = 'TORSION'
   cv_ni = 4, cv_i = 7, 9, 15, 17
   anchor_position = -1.85,-1.8,1.8,1.85
   anchor_strength = 500.0,500.0
/
```

Figure 11.9.: *An example* `CV.IN` *file to restrain the* φ *and* ψ *of dialanine.*

```
title line
&cntrl
..., infe = 1
/

&abmd
   mode = 'FLOODING'

   monitor_file = 'abmd.txt'
   monitor_freq = 33
   cv_file = 'cv.in'

   umbrella_file = 'umbrella.nc'

   timescale = 100.0 ! in ps

   selection_freq = 10000
   selection_constant = 0.001

   wt_temperature = 10000.0
   wt_umbrella_file = 'wt_umbrella.nc'
/
```

```
cv_file
&colvar
  cv_type = 'DISTANCE'
  cv_ni = 2, cv_i = 5, 9
  cv_min = -1.0, cv_max = 10.0 ! min is not needed for DISTANCE
  resolution = 0.5 ! required for mode = FLOODING
/
```

Figure 11.10.: *An example* `MDIN` *file and* `CV.IN` *file for* `ABMD` *(only the relevant part is presented in full).*

```
title line
&cntrl
..., infe = 1
/

&bbmd

   ! 0th replica only

   exchange_freq = 100 ! try for exchange every 100 steps

   exchange_log_file = 'bbmd.log'
   exchange_log_freq = 25

   mt19937_seed = 123455 ! random generator seed
   mt19937_file = 'mt19937.nc' ! file to store/load the PRG

   ! not specific for 0th replica

   mode = 'ANALYSIS'

   monitorfile = 'bbmd.01.txt' ! it is wise to have different
                               ! names in different replicas
   monitor_freq = 123
   cv_file = 'cv.in'
/
```

```
cv_file
&colvar
  cv_type = 'DISTANCE'
  cv_ni = 2, cv_i = 5, 9
/
```

Figure 11.11.: *An example* `MDIN` *file and* `CV.IN` *file for* `&bbmd` *flavor of* `ABMD` *(only the relevant part is presented in full).*

```
title line
&cntrl
..., nstlim = 10000
..., infe = 1
/

&stsm      ! parallel case, I from 1 to 8, J from 1 to 16
  image = I
  equilibration = 980
  release = 20
  smoothing = 0.1
  report_centers = 'ALL'

  output_file = 'stsm.00I.J.txt'
  output_freq = 10
  cv_file = 'cv.I'
/
```

```
cv_file
&colvar ! phi
   cv_type = 'TORSION'
   cv_ni = 4, cv_i = 5, 7, 9, 15
   anchor_position = -3.00
   anchor_strength = 20.0
/
&colvar ! psi
   cv_type = 'TORSION'
   cv_ni = 4, cv_i = 7, 9, 15, 17
   anchor_position = 3.00
   anchor_strength = 20.0
/
```

Figure 11.12.: *An example* `MDIN` *file and* `CV.IN` *file for* `&stsm` *in parallel case (only the relevant part is presented in full)*

```
title line
&cntrl
..., nstlim = 160000
..., infe = 1
/

&stsm      ! sequential case, I from 1 to 8
  image = I
  repeats = 16
  equilibration = 980
  release = 20
  smoothing = 0.1
  report_centers = 'ALL'

  output_file = 'stsm.00I.txt'
  output_freq = 10
  cv_file = 'cv.I'
/
```

Figure 11.13.: *An example* `MDIN` *file for* `&stsm` *in sequential case (only the relevant part is presented in full)*

```
NFE : #   restoring restraint:
NFE : #   << colvar(1) = -3.000000 >>
NFE : #   << colvar(2) = 3.000000 >>
NFE : #   equilibration begins...
......
NFE : #   << colvar(1) = -2.500688 -2.586429 >>
NFE : #   << colvar(2) = 2.782725 3.082205 >>
NFE : #  drifted center of image 1 :            8      -2.54041796       2.70644813
NFE : #  drifted center of image 2 :            8      -2.54963153       2.71715138
......
NFE : #  drifted center of image 8 :            8       1.02191205       0.16837852
NFE : #  smoothed center of image 1 :            8      -2.54041796       2.70644813
NFE : #  smoothed center of image 2 :            8      -2.60416697       2.75924174
......
NFE : #  smoothed center of image 8 :            8       1.02191205       0.16837852
NFE : #  reparametrized center of image 1 :            8      -2.54041796
2.70644813
NFE : #  reparametrized center of image 2 :            8      -2.06027108
2.47738701
......
NFE : #  reparametrized center of image 8 :            8       1.02191205
0.16837852
```

Figure 11.14.: *An example of* `MDOUT` *file for STSM run (only part is presented, and some centers are also omitted)*

12. NMR refinement

We find the *sander* module to be a flexible way of incorporating a variety of restraints into a optimization procedure that includes energy minimization and dynamical simulated annealing. The "standard" sorts of NMR restraints, derived from NOE and J-coupling data, can be entered in a way very similar to that of programs like DISGEO, DIANA or X-PLOR; an aliasing syntax allows for definitions of pseudo-atoms, connections with peak numbers in spectra, and the use of "ambiguous" constraints from incompletely-assigned spectra. More "advanced" features include the use of time-averaged constraints, use of multiple copies (LES) in conjunction with NMR refinement, and direct refinement against NOESY intensities, paramagnetic and diamagnetic chemical shifts, or residual dipolar couplings. In addition, a key strength of the program is its ability to carry out the refinements (usually near the final stages) using an explicit-solvent representation that incorporates force fields and simulation protocols that are known to give pretty accurate results in many cases for unconstrained simulations; this ability should improve predictions in regions of low constraint density and should help reduce the number of places where the force field and the NMR constraints are in "competition" with one another.

Since there is no generally-accepted "recipe" for obtaining solution structures from NMR data, the comments below are intended to provide a guide to some commonly-used procedures. Generally speaking, the programs that need to be run to obtain NMR structures can be divided into three parts:

1. *front-end* modules, which interact with NMR databases that provide information about assignments, chemical shifts, coupling constants, NOESY intensities, and so on. We have tried to make the general format of the input straightforward enough so that it could be interfaced to a variety of programs. We generally use the FELIX and NMRView codes, but the principles should be similar for other ways of keeping track of a database of NMR spectral information. As the flow-chart in Section 12.7 indicates, there are only a few files that need to be created for NMR restraints; these are indicated by the solid rectangles. The primary distance and torsion angle files have a fairly simple format that is largely compatible with the DIANA programs; if one wishes to use information from ambiguous or overlapped peaks, there is an additional "MAP" file that makes a translation from peak identifiers to ambiguous (or partial) assignments. Finally, there are some specialized (but still pretty straightforward) file formats for chemical shift or residual dipolar coupling restraints.

   There are a variety of tools, besides the ones described below, that can assist in preparing input for structure refinement in Amber.

   - The SANE (Structure Assisted NOE Evaluation) package, *https://ambermd.org/sane.zip*, is widely used at The Scripps Research Institute.[257]
   - If you use Bruce Johnson's NmrView package, you might also want to look at the additions to that: *http://garbanzo.scripps.edu/nmrgrp/wisdom/pipe/tips_scripts.html*. In particular, the *xpkTOupl* and *starTOupl* scripts there convert NmrView peak lists into the "7-column" needed for input to makeDIST_RST.
   - Users of the MARDIGRAS programs from UCSF can use the *mardi2amber* program to do conversion to Amber format: *http://picasso.ucsf.edu/mardihome.html*

2. *restrained molecular dynamics,* which is at the heart of the conformational searching procedures. This is the part that *sander* itself handles.

3. *back-end* routines that do things like compare families of structures, generate statistics, simulate spectra, and the like. For many purposes, such as visualization, or the running of procheck-NMR, the "interface" to such programs is just the set of PDB files that contain the family of structures to be analyzed. These general-purpose structure analysis programs are available in many locations and are not discussed here. The

principal *sander*-specific tool is *sviol*, which prepares tables and statistics of energies, restraint violations, and the like.

12.1. Distance, angle and torsional restraints

Distance, angle, and other restraints are read from the DISANG file if *nmropt* > 0. Namelist rst ("&rst") contains the following variables; it is read repeatedly until a namelist &rst statement is found with IAT(1)=0, or until reaching the end of the DISANG file.

[In many cases, the user will not prepare this section of the input by hand, but will use the auxiliary programs *makeDIST_RST, makeANG_RST* and *makeCHIR_RST* to prepare input from simpler files. See also the programs *cyanarest_to_amberRST* and *nef_to_RST* if you have restraints in Cyana or NEF (NMR Exchange Format) formats.]

12.1.1. Variables in the &rst namelist:

`iat(1)`→`iat(8)`

- *If IRESID = 0 (normal operation):* The atoms defining the restraint. Type of restraint is determined (in order) by:
  1. If IAT(3) = 0, this is a distance restraint.
  2. If IAT(4) = 0, this is an angle restraint.
  3. If IAT(5) = 0, this is a torsional (or J-coupling, if desired) restraint or a genereralized distance restraint of 4 atoms, a type of restraint new as of Amber 10 (*sander* only, see below).
  4. If IAT(6) = 0, this is a plane-point angle restraint, a second restraint new as of Amber 10 (*sander* only). The angle is measured between the normal of a plane defined by IAT(1)..IAT(4) and the vector from the center of mass of atoms IAT(1)..IAT(4) to the position of IAT(5). The normal is defined by (r1 - r2) $\times$ (r3 - r4), where rn is the position of IAT(n).
  5. If IAT(7) = 0, this is a generalized distance restraint of 6 atoms (see below).
  6. Otherwise, if IAT(1)..IAT(8) are all nonzero, this is a plane- plane angle restraint, a third new restraint type as of Amber 10 (*sander* only, or a generalized distance restraint of 8 atoms (see below). For the plane-plane restraint, the angle is measured between the two normals of the two planes, which are defined by (r1 - r2) $\times$ (r3 - r4) and (r5 - r6) $\times$ (r7 - r8). In the case of either planar restraint, the plane may be defined using three atoms instead of four simply by using one atom twice.

  If any of IAT(n) are < 0, then a corresponding group of atoms is defined below, and the coordinate- averaged position of this group will be used in place of atom IAT(n). A new feature as of Amber 10, atom groups may be used not only in distance restraints, but also in angle, torsion, the new plane restraints, or the new generalized restraints. If this is a distance restraint, and IAT1 <0, then a group of atoms is defined below, and the coordinate-averaged position of this group will be used in place of the coordinates of atom 1 [IAT(1)]. Similarly, if IAT(2) < 0, a group of atoms will be defined below whose coordinate-averaged position will be used in place of the coordinates for atom 2 [IAT(2)].

- *If IRESID=1:* IAT(1)..IAT(8) point to the *residues* containing the atoms comprising the internal. Residue numbers are the absolute in the entire system. In this case, the variables ATNAM(1)..ATNAM(8) must be specified and give the character names of the atoms within the respective residues. If any of IAT(n) are less than zero, then group input will still be read in place of the corresponding atom, as described below.

- *Defaults for IAT(1)→IAT(8) are 0.*

`rstwt(1)`→`rstwt(4)` New as of Amber 10 (*sander* only), users may now define a single restraint that is a function of multiple distance restraints, called a "generalized distance coordinate" restraint. The energy of such a restraint has the following form:

$$U = k(w_1|\mathbf{r}_1 - \mathbf{r}_2| + w_2|\mathbf{r}_3 - \mathbf{r}_4| + w_3|\mathbf{r}_5 - \mathbf{r}_6| + w_4|\mathbf{r}_7 - \mathbf{r}_8| - r_0)^2$$

where the weights $w_n$are given in rstwt(1)..rstwt(4) and the positions $\mathbf{r}_n$are the positions of the atoms in iat(1)..iat(8).

Generalized distance coordinate restraints must be defined with either 4, 6, or 8 atoms and 2, 3, or 4 corresponding nonzero weights in rstwt(1)..rstwt(4). Weights may be any positive or negative real number.

If all the weights in rstwt(1)..rstwt(4) are zero and four atoms are given in iat(1)..iat(4) for the restraint, the restraint is a torsional or J-coupling restraint. If eight atoms are given in iat(1)..iat(8) and all weights are zero, the restraint is a plane-plane angle restraint. However, if the weights are nonzero, the restraint will be a generalized distance coordinate restraint.

Default for rstwt(1)..rstwt(4) is 0.0

`restraint` New as of Amber 10 (*sander* only), users may now use a "natural language" system to define restraints by using the RESTRAINT character variable. Valid restraints defined in this manner will begin with a "distance( )" "angle( )" "torsion( )" or "coordinate( )" keyword. Within the parentheses, the atoms that make up the restraint are specified. Atoms may be defined either with an explicit atom number or by using ambmask format, namely :(residue#)@(atom name). Atoms may be separated by commas, spaces, or parentheses. Additionally, negative integers may be used if atom groupings are defined in other variables in the namelist as described below. In addition to the principle distance, angle, torsion, and coordinate keywords, Some keywords may be used within the principle keywords to define more complicated restraints. The keyword "plane( )" may be used once or twice within the parentheses of the "angle( ) keyword to define a planar restraint. Defining one plane grouping plus one other atom in this manner will create a plane-point angle restraint as described above. Defining two plane groupings will create a plane-plane angle restraint. The keyword "plane( )" may only be used inside of "angle( )," and is necessary to define either a plane-point or plane-plane restraint.
Within the "coordinate( )" keyword, the user must use 2 to 4 "distance( )" keywords to define a generalized distance coordinate restraint. The "distance( )" keyword functions just like it does when used to define a traditional distance restraint. The user may specify any two atom numbers, masks, or negative numbers corresponding to atom groups defined outside of RESTRAINT. Additionally, following each "distance( )" keyword inside "coordinate( )" the user must specify a real-number weight to be applied to each distance making up the generalized coordinate.
The "com( )" keyword may be used within any other keyword to define a center of mass grouping of atoms. Within the parenthesis, the user will enter a list of atom numbers or masks. Negative numbers, which correspond to externally-defined groups, may not be used.
Any type of parenthetical character, i.e., ( ), [ ], or { }, may be used wherever parentheses have been used above.

The following are all examples of valid restraint definitions:

```
restraint = "distance( (45) (49) )"
= "angle (:21@C5' :21@C4' 108)"
= "torsion[-1,-1,-1, com(67, 68, 69)]"
= "angle( -1, plane(81, 85, 87, 90) )"
= "angle(plane(com(9,10),:5@CA,31,32),plane(14,15,15,16))"
= "coordinate(distance(:5@C3',:6@O5'),-1.0,distance(134,-1),1.0)"
```

There is a 256 character limit on RESTRAINT, so if a particularly large atom grouping is desired, it is necessary to specify a negative number instead of "com( )" and define the group as described below. RESTRAINT will only be parsed if IAT(1) = 0, otherwise the information in IAT(1) .. IAT(8) will define the restraint. *Default for restraint is ' '.*

atnam If IRESID = 1, then the character names of the atoms defining the internal are contained in ATNAM(1)→ATNAM(8). Residue IAT(1) is searched for atom name ATNAM(1); residue IAT(2) is searched for atom name ATNAM(2); etc. *Defaults for ATNAM(1)→ATNAM(8) are ' '.*

iresid Indicates whether IAT(I) points to an atom # or a residue #. See descriptions of IAT() and ATNAM() above. If RESTRAINT is used to define the internal instead of IAT(), IRESID has no effect on how RESTRAINT is parsed. However, it will affect the behavior of atom group definitions as described below if negative numbers are specified within RESTRAINT. *Default = 0.*

nstep1, nstep2 This restraint is applied for steps/iterations NSTEP1 through NSTEP2. If NSTEP2 = 0, the restraint will be applied from NSTEP1 through the end of the run. Note that the first step/iteration is considered step zero (0). *Defaults for NSTEP1, NSTEP2 are both 0.*

irstyp Normally, the restraint target values defined below (R1→R4) are used directly. If IRSTYP = 1, the values given for R1→R4 define relative displacements from the current value (value determined from the starting coordinates) of the restrained internal. For example, if IRSTYP=1, the current value of a restrained distance is 1.25, and R1 (below) is -0.20, then a value of R1=1.05 will be used. *Default is IRSTYP=0.*

ialtd Determines what happens when a distance restraint gets very large. If IALTD=1, then the potential "flattens out", and there is no force for large violations; this allows for errors in constraint lists, but might tend to ignore constraints that *should* be included to pull a bad initial structure towards a more correct one. When IALTD=0 the penalty energy continues to rise for large violations. See below for the detailed functional forms that are used for distance restraints. Set IALTD=0 to recover the behavior of earlier versions of *sander*. Default value is 0, or the last value that was explicitly set in a previous restraint. This value is set to 1 if *makeDIST_RST* is called with the *-altdis* flag.

ifvari If IFVARI $>$ 0, then the force constants/positions of the restraint will vary with step number. Otherwise, they are constant throughout the run. If IFVARI $>$0, then the values R1A→R4A, RK2A, and RK3A must be specified (see below). *Default is IFVARI=0.*

ninc If IFVARI $>$ and NINC $>$ 0, then the change in the target values of of R1→R4 and K2,K3 is applied as a step function, with NINC steps/ iterations between each change in the target values. If NINC = 0, the change is effected continuously (at every step). *Default for NINC is the value assigned to NINC in the most recent namelist where NINC was specified. If NINC has not been specified in any namelist, it defaults to 0.*

imult If IMULT=0, and the values of force constants RK2 and RK3 are changing with step number, then the changes in the force constants will be linearly interpolated from rk2→rk2a and rk3→rk3a as the step number changes. If IMULT=1 and the force constants are changing with step number, then the changes in the force constants will be effected by a series of multiplicative scalings, using a single factor, R, for all scalings. *i.e.*

```
rk2a = R**INCREMENTS * rk2
rk3a = R**INCREMENTS * rk3.
```

INCREMENTS is the number of times the target value changes, which is determined by NSTEP1, NSTEP2, and NINC. *Default for IMULT is the value assigned to IMULT in the most recent namelist where IMULT was specified. If IMULT has not been specified in any namelist, it defaults to 0.*

r1→r4, rk2, rk3, r1a→r4a, rk2a, rk3a If IALTD=0, the restraint is a well with a square bottom with parabolic sides out to a defined distance, and then linear sides beyond that. If R is the value of the restraint in question:

- R $<$ r1 Linear, with the slope of the "left-hand" parabola at the point R=r1.

- r1 <= R < r2 Parabolic, with restraint energy $k_2(R-r_2)^2$.
- r2 <= R < r3 E = 0.
- r3 <= R < r4 Parabolic, with restraint energy $k_3(R-r_3)^2$.
- r4 <= R Linear, with the slope of the "right-hand" parabola at the point R=r4.

For torsional restraints, the value of the torsion is translated by +-n*360, if necessary, so that it falls closest to the mean of r2 and r3. Specified distances are in Angstroms. Specified angles are in degrees. Force constants for distances are in kcal/mol-Å2 Force constants for angles are in kcal/mol-rad 2 . (Note that angle positions are specified in degrees, but force constants are in radians, consistent with typical reporting procedures in the literature).

If IALTD=1, distance restraints are interpreted in a slightly different fashion. Again, If R is the value of the restraint in question:

- R < r2 Parabolic, with restraint energy $k_2(R-r_2)^2$.
- r2 <= R < r3 E = 0.
- r3 <= R < r4 Parabolic, with restraint energy $k_3(R-r_3)^2$.
- r4 <= R Hyperbolic, with energy $k_3[b/(R-r_3)+a]$, where $a = 3(r_4-r_3)^2$ and $b = -2(r_4-r_3)^3$. This function matches smoothly to the parabola at $R = r_4$, and tends to an asymptote of ak_3at large R. The functional form is adapted from that suggested by Michael Nilges, *Prot. Eng.* **2,** 27-38 (1988). Note that if *ialtd=1*, the value of r1 is ignored.

`ifvari` **= 0** The values of r1→r4, rk2, and rk3 will remain constant throughout the run.

> 0 The values r1a, r2a, r3a, r4a, r2ka and r3ka are also used. These variables are defined as for r1→r4 and rk2, rk3, but correspond to the values appropriate for NSTEP = NSTEP2: e.g., if IVARI >0, then the value of r1 will vary between NSTEP1 and NSTEP2, so that, e.g. r1(NSTEP1) = r1 and r1(NSTEP2) = r1a. Note that you *must* specify an explicit value for *nstep1* and *nstep2* if you use this option. *Defaults for r1→r4,rk2,rk3,r1a→r4a,rk2a and rk3a are the values assigned to them in the most recent namelist where they were specified. They should always be specified in the first* &rst *namelist.*

`r0, k0, r0a, k0a` New as of Amber 10 (*sander* only), the user may more easily specify a large parabolic well if desired by using R0 and K0, and then R0A and K0A if IFVARI > 0. The parabolic well will have its zero at R = R0 and a force constant of K0. These variables simply map the disired parabolic well into r1→r4, rk2, rk3, r1a→r4a, rk2a, and rk3a in the following manner:

- R1 = 0 for distance, angle, and planar restraints, R1 = R0 - 180 for torsion restraints
- R1A = 0 for distance, angle, and planar restraints, R1A = R0A - 180 for torsion restraints
- R2 = R0; R3 = R0
- R2A = R0A; R3A = R0A
- R4 = R0 + 500 for distance restraints, R4 = 180 for angle and planar restraints, R4 = R0 + 180 for torsion restraints
- RK2 = K0; RK3 = K0
- RK2A = K0A; RK3A = K0A

`rjcoef(1)`→`rjcoef(3)` By default, 4-atom sequences specify torsional restraints. It is also possible to impose restraints on the vicinal 3 J-coupling value related to the underlying torsion. J is related to the torsion τ by the approximate Karplus relationship: $J = A\cos^2(\tau)+B\cos(\tau)+C$. If you specify a nonzero value for either RJCOEF(1) or RJCOEF(2), then a J-coupling restraint, rather than a torsional restraint, will be imposed. At every MD step, J will be calculated from the Karplus relationship with A = RJCOEF(1), B = RJCOEF(2) and C = RJCOEF(3). In this case, the target values (R1->R4, R1A->R4A) and force constants (RK2, RK3, RK2A, RK3A) refer to J-values

for this restraint. RJCOEF(1)->RJCOEF(3) must be set individually for each torsion for which you wish to apply a J-coupling restraint, and RJCOEF(1)->RJCOEF(3) may be different for each J-coupling restraint. With respect to other options and reporting, J-coupling restraints are treated identically to torsional restraints. This means that if time-averaging is requested for torsional restraints, it will apply to J-coupling restraints as well. The J-coupling restraint contribution to the energy is included in the "torsional" total. And changes in the relative weights of the torsional force constants also change the relative weights of the J-coupling restraint terms. Setting RJCOEF has no effect for distance and angle restraints. *Defaults for RJCOEF(1)->RJCOEF(3) are 0.0.*

igr1(i),i=1→200, igr2(i),i=1→200, ... igr8(i),i..1=1→200 If IAT(n) < 0, then IGRn() gives the atoms defining the group whose coordinate averaged position is used to define "atom n" in a restraint. Alternatively, if RESTRAINT is used to define the internal, then if the nth atom specified is a number less than zero, IGRn() gives the atoms defining the group whose coordinate averaged position is used to define "atom n" in a restraint. If IRESID = 0, absolute atom numbers are specified by the elements of IGRn(). If IRESID = 1, then IGRn(I) specifies the number of the residue containing atom I, and the name of atom I must be specified using GRNAMn(I). A maximum of 200 atoms (N # of atoms if using pmemd) are allowed in any group. Only specify those atoms that are needed. Default value for any unspecified element of IGRn(i) is 0.

`fxyz` If iat(3)=0 and igr1 and/or igr2 is defined then it is possible to weight the x, y, z components of the force in the restraint to 0 (no force) or 1 (full restraint force). Ex: fxyz=0, 0, 1. This sets no additional restraint force on the x component or y-component of the restraint force, and full z-component restraint force. Default fxyz=1,1,1. Note: When setting fxyz, the r1, r2, r3, r4 values should be set relative to a weighted distance $\sqrt{(w_x * d_x)^2 + (w_y * d_y)^2 + (w_z * d_z)^2}$, so if fxyz=0,0,1 then the only distance taken into account when comparing to r1,r2,r3,r4 is the z distance between the molecule and the center of mass. Note that the DUMPAVE value when outxyz=0 is also just the weighted distance.

`outxyz` If iat(3)=0 and igr1 and/or igr2 is defined then it is possible to output the x, y, z components of the force in the restraint if outxyz is set to 1. Default outyz=0. When outxyz is set to 1, the components of the distance and total distance are outputed in DUMPAVE in the order of the x-component, y-component, z-component, total distance.

`grnam1(i),i=1→200,` grnam2(i),i=1→200, ... grnam8(i),i=1→200 If group input is being specified (IGRn(1) > 0), *and* IRESID = 1, then the character names of the atoms defining the group are contained in GRNAMn(i), as described above. In the case IAT(1) < 0, each residue IGR1(i) is searched for an atom name GRNAM1(i) and added to the first group list. In the case IAT(2) < 0, each residue IGR2(i) is searched for an atom name GRNAM2(i) and added to the second group list. *Defaults for GRNAMn(i) are ' '.*

`ir6` If a group coordinate-averaged position is being used (see IGR1 and IGR2 above), the average position can be calculated in either of two manners: If IR6 = 0, center-of-mass averaging will be used. If IR6=1, the $\left\langle r^{-6} \right\rangle^{-1/6}$ average of all interaction distances to atoms of the group will be used. *Default for IR6 is the value assigned to IR6 in the most recent namelist where IR6 was specified. If IR6 has not been specified in any namelist, it defaults to 0.*

`ifntyp` If time-averaged restraints have been requested (see DISAVE/ANGAVE/TORAVE above), they are, by default, applied to all restraints of the class specified. Time-averaging can be overridden for specific internals of that class by setting IFNTYP for that internal to 1. IFNTYP has no effect if time-averaged restraint are not being used. *Default value is IFNTYP=0.*

`ixpk, nxpk` These are user-defined integers than can be set for each constraint. They are typically the "peak number" and "spectrum number" associated with the cross-peak that led to this particular distance restraint. Nothing is ever done with them except to print them out in the "violation summaries", so that NMR people can more easily go from a constraint violation to the corresponding peak in their spectral database. Default values are zero.

`iconstr` If *iconstr* $>$ 0, (default is 0) a Lagrangian multiplier is also applied to the two-center internal coordinate defined by IAT(1) and IAT(2). The effect of this Lagrangian multiplier is to maintain the initial orientation of the internal coordinate. The rotation of the vector IAT(1)->IAT(2) is prohibited, though translation is allowed. For each defined two-center internal coordinate, a separate Lagrangian multiplier is used. Therefore, although one can use as many multipliers as needed, defining centers should NOT appear in more than one multiplier. This option is compatible with mass centers (i.e., negative IAT(1) or IAT(2)). ICONSTR can be used together with harmonic restraints. RK2 and RK3 should be set to 0.0 if the two-center internal coordinate is a simple Lagrangian multiplier. An example has been included in $AMBERHOME/example/lagmul.

Namelist &rst is read for each restraint. Restraint input ends when a namelist statement with iat(1) = 0 (or iat(1) not specified) is found. Note that comments can precede or follow any namelist statement, allowing comments and restraint definitions to be freely mixed.

12.2. NOESY volume restraints

After the previous section, NOESY volume restraints may be read. This data described in this section is only read if NMROPT = 2. The molecule may be broken in overlapping submolecules, in order to reduce time and space requirements. Input *for each submolecule* consists of namelist "&noeexp", followed *immediately* by standard Amber "group" cards defining the atoms in the submolecule. In addition to the submolecule input ("&noeexp"), you may also need to specify some additional variables in the cntrl namelist; see the "NMR variables" description in that section.

In many cases, the user will not prepare this section of the input by hand, but will use the auxiliary program *makeDIST_RST* to prepare input from simpler files.

Variables in the &noeexp namelist:

For each submolecule, the namelist "&noeexp" is read (either from *stdin* or from the NOESY redirection file) which contains the following variables. There are no effective defaults for *npeak, emix, ihp, jhp,* and *aexp*: you must specify these.

`npeak(imix)` Number of peaks for each of the "imix" mixing times; if the last mixing time is *mxmix*, set NPEAK(*mxmix+1*) = -1. End the input when NPEAK(1) $<$ 0.

`emix(imix)` Mixing times (in seconds) for each mixing time.

`ihp(imix,ipeak), jhp(imix,ipeak)` Atom numbers for the atoms involved in cross-peak "ipeak" at mixing time "imix"

`aexp(imix,ipeak)` Experimental target integrated intensity for this cross peak. If AEXP is negative, this cross peak is part of a set of overlapped peaks. The computed intensity is added to the peak that follows; the next time a peak with AEXP $>$ 0 is encountered, the running sum for the calculated peaks will be compared to the value of AEXP for that last peak in the list. In other words, a set of overlapped peaks is represented by one or more peaks with AEXP $<$ 0 followed by a peak with AEXP $>$ 0. The computed total intensity for these peaks will be compared to the value of AEXP for the final peak.

`arange(imix,ipeak)` "Uncertainty" range for this peak: if the calculated value is within $\pm$ARANGE of AEXP, then no penalty will be assessed. Default uncertainties are all zero.

`awt(imix,ipeak)` Relative weight for this cross peak. Note that this will be multiplied by the overall weight given by the NOESY weight change cards in the weight changes section (Section 1). Default values are 1.0, unless INVWT1,INVWT2 are set (see below), in which case the input values of AWT are ignored.

`invwt1,invwt2` Lower and upper bounds on the weights for the peaks respectively, such that the relative weight for each peak is 1/intensity if 1/intensity lies between the lower and upper bounds. This is the intensity after being scaled by *oscale*. The inverse weighing scheme adopted by this option prevents placing too much influence on the strong peaks at the expense of weaker peaks and was previously invoked using the compilation flag "INVWGT". Default values are INVWT1=INVWT2=1.0, placing equal weights on all peaks.

`omega` Spectrometer frequency, in Mhz. Default is 500. It is possible for different sub-molecules to have different frequencies, but omega will only change when it is explicitly re-set. Hence, if all of your data is at 600 Mhz, you need only set *omega* to 600. in the first submolecule.

`taurot` Rotational tumbling time of the molecule, in nsec. Default is 1.0 nsec. Like *omega*, this value is "sticky", so that a value set in one submolecule will remain until it is explicitly reset.

`taumet` Correlation time for methyl jump motion, in ns. This is only used in computing the intra-methyl contribution to the rate matrix. The ideas of Woessner are used, specifically as recommended by Kalk & Berendsen.[258] Default is 0.0001 ns, which is effectively the fast motion limit. The default is consistent with the way the rest of the rate matrix elements are determined (also in the fast motion limit,) but probably is not the best value to use, since methyl groups appear to have T1 values that are systematically shorter than other protons, and this is likely to arise from the fact that the methyl correlation time can be near to the inverse of the spectrometer frequency. A value of 0.02 - 0.05 ns is probably better than 0.0001, but this is still an active research area, and you are on your own here, and should consult the literature for further discussion.[259] As with *omega*, *taumet* can be different for different sub-molecules, but will only change when it is explicitly re-set.

`id2o` Flag for determining if exchangeable protons are to be included in the spin-diffusion calculation. If ID2O=0 (default) then all protons are included. If ID2O=1, then all protons bonded to nitrogen or oxygen are assumed to not be present for the purposes of computing the relaxation matrix. No other options exist at present, but they could easily be added to the subroutine *indexn*. Alternatively, you can manually rename hydrogens in the *prmtop* file so that they do not begin with "H": such protons will not be included in the relaxation matrix. (*Note:* for technical reasons, the HOH proton of tyrosine must always be present, so setting ID2O=1 will not remove it; we hope that this limitation will be of minor importance to most users.) The *id2o* variable retains its value across namelist reads, *i.e.* its value will only change if it is explicitly reset.

`oscale` overall scaling factor between experimental and computed volume units. The experimental intensities are multiplied by *oscale* before being compared to calculated intensities. This means that the weights WNOESY and AWT always refer to "theoretical" intensity scales rather than to the (arbitrary) experimental units. The *oscale* variable retains its value across namelist reads, *i.e.* its value will only change if it is explicitly reset. The initial (default) value is 1.0.

The atom numbers *ihp* and *jhp* are the absolute atom numbers. For methyl groups, use the number of the last proton of the group; for the delta and epsilon protons of aromatic rings, use the delta-2 or epsilon-2 atom numbers. Since this input requires you to know the absolute atom numbers assigned by Amber to each of the protons, you may wish to use the separate *makeDIST_RST* program which provides a facility for more turning human-readable input into the required file for *sander*.

Following the &noeexp namelist, give the Amber "group" cards that identify this submolecule. This combination of "&noeexp" and "group" cards can be repeated as often as needed for many submolecules, subject to the limits described in the *nmr.h* file. As mentioned above, this input section ends when NPEAK(1) $<$ 0, or when and end-of-file is reached.

12.3. Chemical shift restraints

After reading NOESY restraints above (if any), read the chemical shift restraints in namelist *&shf*, or the pseudocontact restraints in namelist *&pcshift*. Reading this input is triggered by the presence of a SHIFTS line in the I/O redirection section. In many cases, the user will not prepare this section of the input by hand, but will use the auxiliary programs *shifts* or *fantasian* to prepare input from simpler files.

Variables in the &shf namelist.

(Defaults are only available for *shrang, wt*, *nter*, and *shcut*; you must specify the rest.)

`nring` Number of rings in the system.

`natr`(i) Number of atoms in the *i-th* ring.

`iatr`(j,i) Absolute atom number for the *j-th* atom of the *i-th* ring.

`namr`(i) Eight-character string that labels the *i-th* ring. The first three characters give the residue name (in caps); the next three characters contain the residue number (right justified); column 7 is blank; column 8 may optionally contain an extra letter to distinguish the two rings of trp, or the 5 or 8 rings of the heme group.

`str`(i) Ring current intensity factor for the *i-th* ring. Older values are summarized by Cross and Wright;[260] more recent empirical parametrizations seem to give improved results.[261, 262]

`nprot` Number of protons for which penalty functions are to be set up.

`iprot`(i) Absolute atom number of the *i-th* proton whose shifts are to be evaluated. For equivalent protons, such as methyl groups or rapidly flipping phenylalanine rings, enter all two or three atom numbers in sequence; averaging will be controlled by the *wt* parameter, described below.

`obs`(i) Observed secondary shift for the *i-th* proton. This is typically calculated as the observed value minus a random coil reference value.

`shrang`(i) "Uncertainty" range for the observed shift: if the calculated shift is within $\pm$SHRANG of the observed shift, then no penalty will be imposed. The default value is zero for all shifts.

`wt`(i) Weight to be assigned to this penalty function. Note that this value will be multiplied by the overall weight (if any) given by the SHIFTS command in the assignment of weights (above). Default values are 1.0. For sets of equivalent protons, give a negative weight for all but the last proton in the group; the last proton gets a normal, positive value. The average computed shift of the group will be compared to *obs* entered for the last proton.

`shcut` Values of calculated shifts will be printed only if the absolute error between calculated and observed shifts is greater than this value. *Default = 0.3 ppm.*

`nter` Residue number of the N-terminus, for protein shift calculations; *default = 1.*

`cter` Residue number of the C-terminus, for protein shift calculations. Believe it or not, the current code cannot figure this out for itself.

In typical usage, the *shifts* program (`http://casegroup.rutgers.edu/shifts.html`) would be used to create this file, with a typical command line:

```
shifts -readobs -sander '::H*' gcg10
```

Sample input and output files are in `$AMBERHOME/test/rdc`.

12.4. Pseudocontact shift restraints

The PCSHIFT module allows the inclusion of pseudocontact shifts as constraints in energy minimization and molecular dynamics calculations on paramagnetic molecules. The pseudocontact shift depends on the magnetic susceptibility anisotropy of the metal ion and on the location of the resonating nucleus with respect to the axes of the magnetic susceptibility tensor. For the nucleus i, it is given by:

$$\delta^i_{pc} = \sum_j \frac{1}{12\pi r^3_{ij}} \left[\Delta\chi^j_{ax}(3n^2_{ij} - 1) + (3/2)\Delta\chi^j_{rh}(l^2_{ij} - m^2_{ij}) \right]$$

where l_{ij}, m_{ij}, and n_{ij} are the direction cosines of the position vector of atom i with respect to the j-th magnetic susceptibility tensor coordinate system, r_{ij} is the distance between the j-th paramagnetic center and the proton i, $\Delta\chi_{ax}$ and $\Delta\chi_{rh}$ are the axial and the equatorial (rhombic) anisotropies of the magnetic susceptibility tensor of the j-th paramagnetic center. For a discussion, see Ref. [263].

The PCSHIFT module to be used needs a namelist file which includes information on the magnetic susceptibility tensor and on the paramagnetic center, and a line of information for each nucleus. This module allows to include more than one paramagnetic center in the calculations. To include pseudocontact shifts as constraints in energy minimization and molecular dynamics calculations the NMROPT flag should be set to 2, and a *PCSHIFT=filename* statement entered in the I/O redirection section.

To perform molecular dynamics calculations it is necessary to eliminate the rotational and translational degree of freedom about the center of mass (this because during molecular dynamics calculations the relative orientation between the external reference coordinate system and the magnetic anisotropy tensor coordinate system has to be fixed).This option can be obtained with the NSCM flag of *sander*.

Variables in the pcshift namelist

`nprot` number of pseudocontact shift constraints.

`nme` number of paramagnetic centers.

`nmpmc` name of the paramagnetic atom

`optphi(n), opttet(n), optomg(n), opta1(n), opta2(n)` the five parameters of the magnetic anisotropy tensor for each paramagnetic center.

`optkon` force constant for the pseudocontact shift constraints

Following this, there is a line for each nucleus for which the pseudocontact shift information is given has to be added. Each line contains :

`iprot(i)` atom number of the i-th proton whose shift is to be used as constraint.

`obs(i)` observed pseudocontact shift value, in ppm

`wt(i)` relative weight

`tolpro(i)` relative tolerance ix mltpro

`mltpro(i)` multiplicity of the NMR signal (for example the protons of a methyl group have mltprot(i)=3)

Example

Here is a &pcshf namelist example: a molecule with three paramagnetic centers and 205 pseudocontact shift constraints.

```
&pcshf
nprot=205,
nme=3,
nmpmc='FE ',
optphi(1)=-0.315416,
opttet(1)=0.407499,
optomg(1)=0.0251676,
opta1(1)=-71.233,
opta2(1)=1214.511,
optphi(2)=0.567127,
opttet(2)=-0.750526,
optomg(2)=0.355576,
opta1(2)=-60.390,
opta2(2)=377.459,
optphi(3)=0.451203,
opttet(3)=-0.0113097,
optomg(3)=0.334824,
opta1(3)=-8.657,
opta2(3)=704.786,
optkon=30,
iprot(1)=26, obs(1)=1.140, wt(1)=1.000, tolpro(1)=1.00, mltpro(1)=1,
iprot(2)=28, obs(2)=2.740, wt(2)=1.000, tolpro(2)=.500, mltpro(2)=1,
iprot(3)=30, obs(3)=1.170, wt(3)=1.000, tolpro(3)=.500, mltpro(3)=1,
iprot(4)=32, obs(4)=1.060, wt(4)=1.000, tolpro(4)=.500, mltpro(4)=3,
iprot(5)=33, obs(5)=1.060, wt(5)=1.000, tolpro(5)=.500, mltpro(5)=3,
iprot(6)=34, obs(6)=1.060, wt(6)=1.000, tolpro(6)=.500, mltpro(6)=3,
...
...
iprot(205)=1215, obs(205)=.730, wt(205)=1.000, tolpro(205)=.500,
mltpro(205)=1,
/
```

An *mdin* file that might go along with this, to perform a maximum of 5000 minimization cycles, starting with 500 cycles of steepest descent. PCSHIFT=./pcs.in redirects the input from the namelist "pcs.in" which contains the pseudocontact shift information.

```
Example of minimization including pseudocontact shift constraints
&cntrl
ibelly=0,imin=1,ntpr=100,
ntr=0,maxcyc=500,
ncyc=50,ntmin=1,dx0=0.0001,
drms=.1,cut=10.,
nmropt=2,pencut=0.1, ipnlty=2,
/
&wt type='REST', istep1=0,istep2=1,value1=0.,
value2=1.0, /
&wt type='END' /
DISANG=./noe.in
PCSHIFT=./pcs.in
LISTOUT=POUT
```

12.5. Direct dipolar coupling restraints

Energy restraints based on direct dipolar coupling constants are entered in this section. All variables are in the namelist &align; reading of this section is triggered by the presence of a DIPOLE line in the I/O redirection

section.

When dipolar coupling restraints are turned on, the five unique elements of the alignment tensor are treated as additional variables, and are optimized along with the structural parameters. Their effective masses are determined by the *scalm* parameter entered in the &cntrl namelist. Unlike some other programs, the variables used are the Cartesian components of the alignment tensor in the axis system defined by the molecule itself: *e.g.* $S_{mn} \equiv \langle (3\cos\theta_m \cos\theta_n - \delta_{mn})/2 \rangle$, where $m,n = x,y,z$, and θ_x is the angle between the x axis and the spectrometer field.[264] The factor of 10^5 is just to make the values commensurate with atomic coordinates, since both the coordinates and the alignment tensor values will be updated during the refinement. The calculated dipolar splitting is then

$$D_{calc} = -\left(\frac{10^{-5}\gamma_i\gamma_j h}{2\pi^2 r_{ij}^3}\right) \sum_{m,n=xyz} \cos\phi_m \cdot S_{mn} \cdot \cos\phi_n$$

where ϕ_x is the angle between the internuclear vector and the x axis. Geometrically, the splitting is proportional to the transformation of the alignment tensor onto the internuclear axis. This is just Eqs. (5) and (13) of the above reference, with any internal motion corrections (which might be a part of S_{system}) set to unity. If there is an internal motion correction which is the same for all observations, this can be assimilated into the alignment tensor. The current code does not allow for variable corrections for internal motion. See Ref. [265] for a fuller discussion of these issues.

At the end of the calculation, the alignment tensor is diagonalized to obtain information about its principal components. This allows the alignment tensor to be written in terms of the "axial" and "rhombic" components that are often used to describe alignment.

Variables in the &align namelist.

`ndip` Number of observed dipolar couplings to be used as restraints.

`id, jd` Atom numbers of the two atoms involved in the dipolar coupling.

`dobsl, dobsu` Limiting values for the observed dipolar splitting, in Hz. If the calculated coupling is less than *dobsl*, the energy penalty is proportional to $(D_{calc} - D_{obs,l})^2$; if it is larger than *dobsu*, the penalty is proportional to $(D_{calc} - D_{obs,lu})^2$. Calculated values between *dobsl* and *dobsu* are not penalized. Note that *dobsl* must be less than *dobsu*; for example, if the observed coupling is -6 Hz, and a 1 Hz "buffer" is desired, you could set *dobsl* to -7 and *dobsu* to -5.

`dwt` The relative weight of each observed value. Default is 1.0. The penalty function is thus:

$$E^i_{align} = D^i_{wt}(D^i_{calc} - D^i_{obs(u,l)})^2$$

where D_{wt} may vary from one observed value to the next. Note that the default value is arbitrary, and a smaller value may be required to avoid overfitting the dipolar coupling data.[265]

`dataset` Each dipolar peak can be associated with a "dataset", and a separate alignment tensor will be computed for each dataset. This is generally used if there are several sets of experiments, each with a different sample or temperature, etc., that would imply a different value for the alignment tensor. By default, there is one dataset to which each observed value is assigned.

`num_datasets` The number of datasets in the constraint list. Default is 1.

`s11,s12,s13,s22,s23` Initial values for the Cartesian components of the alignment tensor. The tensor is traceless, so S33 is calculated as -(S11+S22). In order to have the order of magnitude of the S values be roughly commensurate with coordinates in Angstroms, the alignment tensor values must be multiplied by 10^5.

`gigj` Product of the nuclear "g" factors for this dipolar coupling restraint. These are related to the nuclear gyromagnetic rations by $\gamma_N = g_N\beta_N/$. Common values are ^{1}H = 5.5856, ^{13}C = 1.4048, ^{15}N = -0.5663, ^{31}P = 2.2632.

`dij` The internuclear distance for observed dipolar coupling. If a nonzero value is given, the distance is considered to be fixed at the given value. If a *dij* value is zero, its value is computed from the structure, and it is assumed to be a variable distance. For one-bond couplings, it is usually best to treat the bond distance as "fixed" to an effective zero-point vibration value.[266]

`dcut` Controls printing of calculated and observed dipolar couplings. Only values where abs(dobs(u,l) - dcalc) is greater than *dcut* will be printed. Default is 0.1 Hz. Set to a negative value to print all dipolar restraint information.

`freezemol` If this is set to *.true.*, the molecular coordinates are not allowed to vary during dynamics or minimization: only the elements of the alignment tensor will change. This is useful to fit just an alignment tensor to a given structure. Default is *.false.*.

12.6. Residual CSA or pseudo-CSA restraints

Resonance positions in partially aligned media will be shifted from their positions in isotropic media, and this can provide information that is very similar to residual dipolar coupling constriants. This section shows how to input these sorts of restraints. The entry of the alignment tensor is done as in Section 12.5, so you must have a DIPOLE file (with an &align namelist) even if you don't have any RDC restraints. Then, if there is a CSA line in I/O redirection section, that file will be read with the following inputs:

Variables in the &csa namelist.

`ncsa` Number of observed residual CSA peaks to be used as restraints.

`icsa,jcsa,kcsa` Atom numbers for the csa of interest: *jcsa* is the atom whose $\Delta\sigma$ value has been measured; *icsa* and *kcsa* are two atoms bonded to it, used to define the local axis frame for the CSA tensor. See *amber12/test/pcsa/RST.csa* for examples of how to set these.

`cobsl, cobsu` Limiting values for the observed residual CSA, in Hz (not ppm or ppb!). If the calculated value of $\Delta\sigma$ is less than *cobsl*, the energy penalty is proportional to $(\Delta\sigma_{calc} - \Delta\sigma_{obs,l})^2$; if it is larger than *cobsu*, the penalty is proportional to $(\Delta\sigma_{calc} - \Delta\sigma_{obs,u})^2$. Calculated values between *cobsl* and *cobsu* are not penalized. Note that *cobsl* must be less than *cobsu*.

`cwt` The relative weight of each observed value. Default is 1.0. The penalty function is thus:

$$E^i_{csa} = C^i_{wt}(\Delta\sigma^i_{calc} - \Delta\sigma^i_{obs(u,l)})^2$$

where C_{wt} may vary from one observed value to the next. Note that the default value is arbitrary, and a smaller value may be required to avoid overfitting the data.

`datasetc` Each residual CSA can be associated with a "dataset", and a separate alignment tensor will be computed for each dataset. This is generally used if there are several sets of experiments, each with a different sample or temperature, etc., that would imply a different value for the alignment tensor. By default, there is one dataset to which each observed value is assigned. The tensors themselves are entered for each dataset in the DIPOLE file.

`field` Magnetic field (in MHz) for the residual CSA being considered here. This is indexed from 1 to *ncsa,* and is nucleus dependent. For example, if the proton frequency is 600 MHz, then *field* for ^{13}C would be 150, and that for ^{15}N would be 60.

`sigma11, sigma22, sigma12, sigma13, sigma23` Values of the CSA tensor (in ppm) for atom *icsa*, in the local coordinate frame defined by atoms *icsa*, *jcsa* and *kcsa*. See *$AMBERHOME/test/pcsa/RST.csa* for examples of how to set these.

`ccut` Controls printing of calculated and observed residual CSAs. Only values where abs(cobs(u,l) - ccalc) is greater than *ccut* will be printed. Default is 0.1 Hz. Set to a negative value to print all information.

The residual CSA facility is new as of Amber 10, and has not been used as much as other parts of the NMR refinement package. You should study the example files listed above to see how things work. The residual CSA values should closely match those found by the RAMAH package (*http://www-personal.umich.edu/~hashimi/Software.html*), and testing this should be a first step in making sure you have entered the data correctly.

12.7. Preparing restraint files for Sander

Fig. 12.1 shows the general information flow for auxiliary programs that help prepare the restraint files. Once the restraint files are made, Fig. 12.2 shows a flow-chart of the general way in which *sander* refinements are carried out.

The basic ideas of this scheme owe a lot to the general experience of the NMR community over the past decade. Several papers outline procedures in the Scripps group, from which a lot of the NMR parts of *sander* are derived.[257, 267–271] They are by no means the only way to proceed. We hope that the flexibility incorporated into *sander* will encourage folks to experiment with refinement protocols.

12.7.1. Preparing distance restraints: makeDIST_RST

The *makeDIST_RST* program converts a simplified description of distance bounds into a detailed input for *sander*. A variety of input and output filenames may be specified on the command line:

input:

```
-upb <filename> 7-col file of upper distance bounds, OR
-ual <filename> 8-col file of upper and lower bounds, OR
-vol <filename> 7-col file of NOESY volumes
-pdb <filename> Brookhaven format file
-map <filename> MAP file (default:map.DG-AMBER)
-les <filename> LES atom mappings, made by addles
```

output:

```
-dgm <filename> DGEOM95 restraint format
-rst <filename> SANDER restraint format
-svf <filename> Sander Volume Format, for NOESY refinement
```

other options:

```
-help (gives you this explanation, overrides other parameters)
-report (gives you short runtime diagnostic output)
-nocorr (do not correct upper bound for r**-6 averaging)
-altdis (use alternative form for the distance restraints)
```

The *7/8 column distance bound* file is essentially that used by the DIANA or DISGEO programs. It consists of one-line per restraint, which would typically look like the following:

```
23 ALA HA 52 VAL H 3.8 # comments go here
```

The first three columns identify the first proton, the next three the second proton, and the seventh column gives the upper bound. Only the first three letters of the residue name are used, so that DIANA files that contain residues like "ASP-" will be correctly interpreted. An alternate, 8-column, format has both upper and lower bounds as the seventh and eighth columns, respectively. A typical line might in an "8-col" file might look like this:

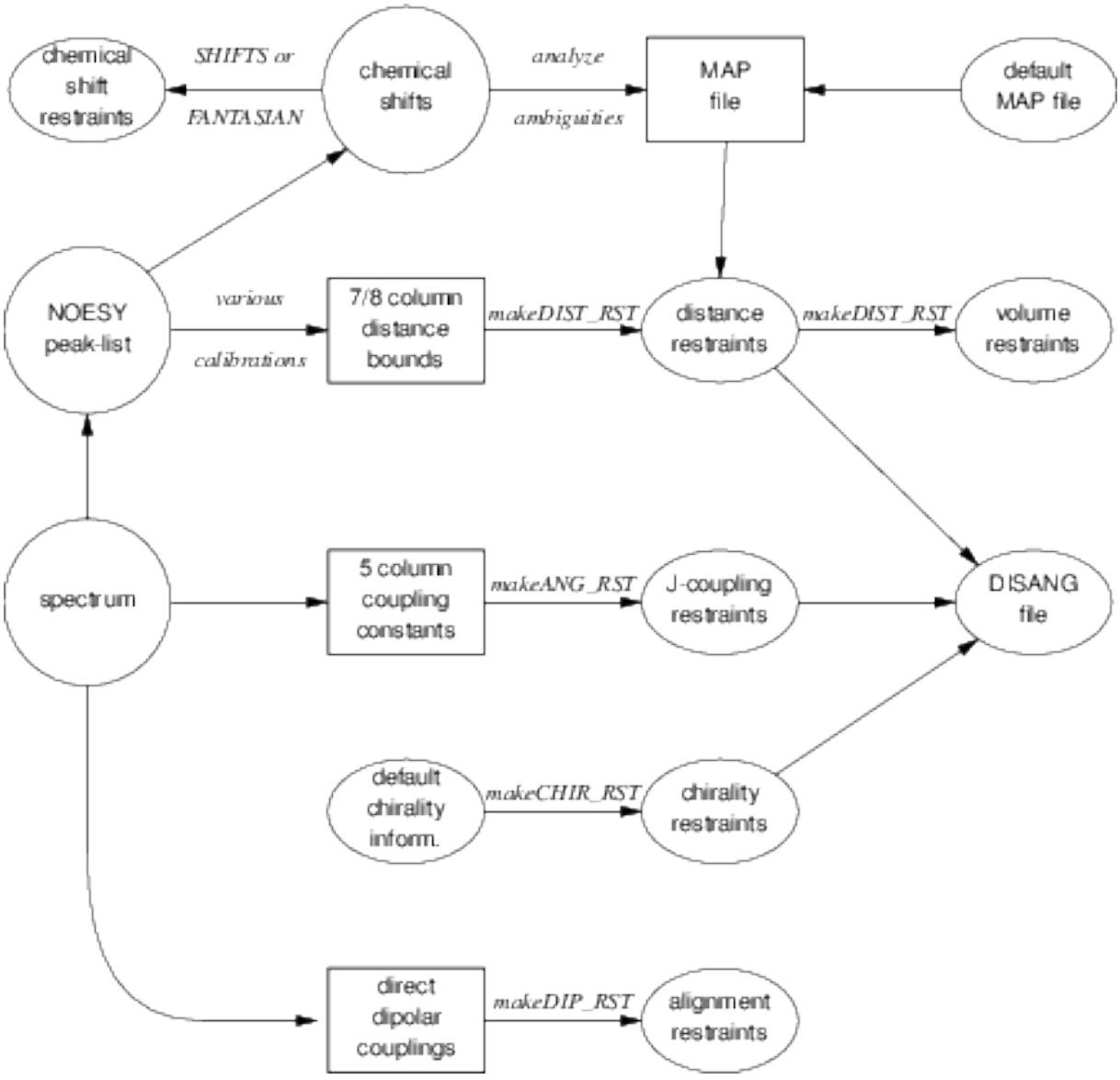

Figure 12.1.: *Notation:* circles *represent logical information, whose format might differ from one project to the next;* solid rectangles *are in a specific format (largely compatible with DIANA and other programs), and are intended to be read and edited by the user;* ellipses *are specific to* sander*, and are generally not intended to be read or edited manually. The conversion of NOESY volumes to distance bounds can be carried out by a variety of programs such as* mardigras *or* xpk2bound *that are not included with Amber. Similarly, the analysis and partial assignment of ambiguous or overlapped peaks is a separate task; at TSRI, these are typically carried out using the programs* xpkasgn *and* filter.pl

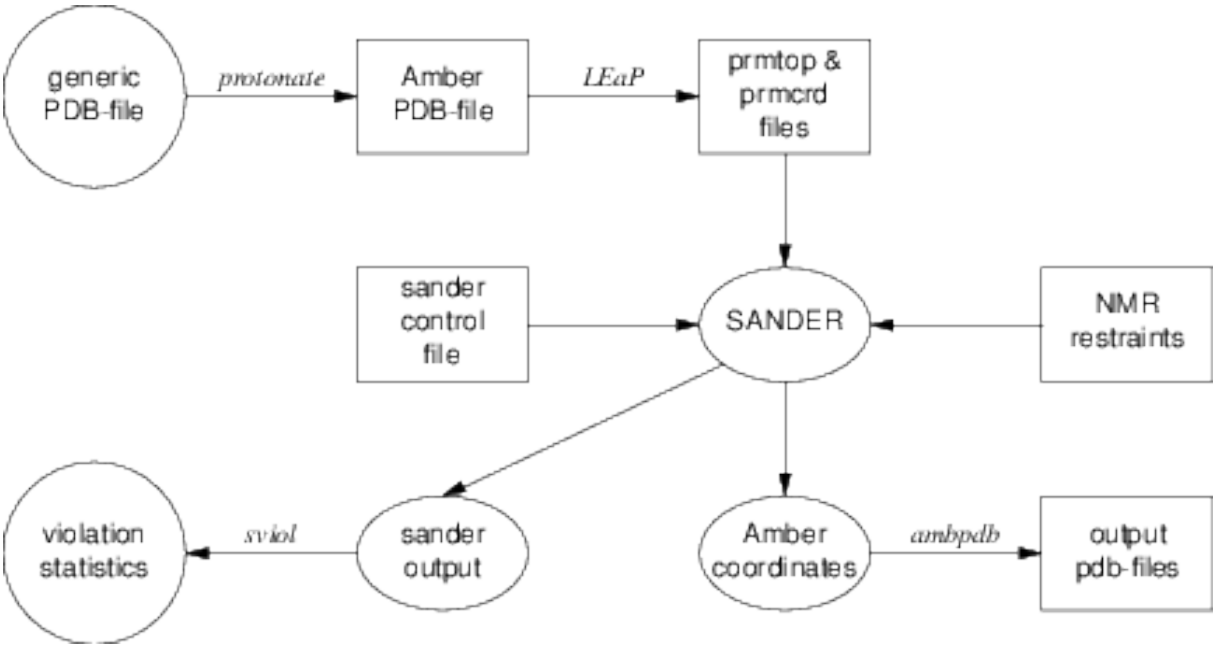

Figure 12.2.: *General organization of NMR refinement calculations.*

```
23 ALA HA 52 VAL H 3.2 3.8 # comments go here
```

Here the lower bound is 3.2 Åand the upper bound is 3.8 Å. Comments typically identify the spectrum and peak-number or other identification that allow cross-referencing back to the appropriate spectrum. If the comment contains the pattern "<integer>:<integer>", then the first integer is treated as a peak-identifier, and the second as a spectrum-identifier. These identifiers go into the *ixpk* and *nxpk* variables, and will later be printed out in *sander*, to facilitate going back to the original spectra to track down violations, etc.

The format for the *-vol* option is the same as for the *-upb* option except that the seventh column holds a peak intensity (volume) value, rather than a distance upper bound.

The input PDB file must exactly match the Amber *prmtop* file that will be used; use the ambpdb -aatm command to create this.

If all peaks involved just single protons, and were fully assigned, this is all that one would need. In general, though, some peaks (especially methyl groups or fast-rotating aromatic rings) represent contributions from more than one proton, and many other peaks may not be fully assigned. *Sander* handles both of these situations in the same way, through the notion of an "ambiguous" peak, that may correspond to several assignments. These peaks are given two types of special names in the 7/8-column format file:

1. Commonly-occurring ambiguities, like the lack of stereospecific assignments to two methylene protons, are given names defined in the default MAP file. These names, also more-or-less consistent with DIANA, are like the names of "pseudo-atoms" that have long been used to identify such partially assigned peaks, e.g. "QB" refers to the (HB2,HB3) combination in most residues, and "MG1" in valine refers collectively to the three methyl protons at position CG1, etc.

2. There are generally also molecule-specific ambiguities, arising from potential overlap in a NOESY spectrum. Here, the user assigns a unique name to each such ambiguity or overlap, and prepares a list of the potential assignments. The names are arbitrary, but might be constructed, for example, from the chemical shifts that identify the peak, e.g. "p_2.52" might identify the set of protons that could contribute to a peak at 2.52 ppm. The chemical shift list can be used to prepare a list of potential assignments, and these lists can often be pruned by comparison to approximate or initial structures.

The default and molecule-specific MAP files are combined into a single file, which is used, along with the 7-column restraint file, the the program *makeDIST_RST* to construct the actual *sander* input files. You should

consult the help file for makeDIST_RST for more information. For example, here are some lines added to the MAP file for a recent TSRI refinement:

```
AMBIG n2:68 = HE 86 HZ 86
AMBIG n2:72 = HE 24 HD 24 HZ 24
AMBIG n2:73 = HN 81 HZ 13 HE 13 HD 13 HZ 24
AMBIG n2:78 = HN 76 HZ 13 HE 13 HZ 24
AMBIG n2:83 = HN 96 HN 97 HD 97 HD 91
AMBIG n2:86 = HD1 66 HZ2 66
AMBIG n2:87 = HN 71 HH2 66 HZ3 66 HD1 66
```

Here the spectrum name and peak number were used to construct a label for each ambiguous peak. Then, an entry in the restraint file might look like this:

```
123 GLY HN 0 AMB n2:68 5.5
```

indicating a 5.5 Åupper bound between the amide proton of Gly 123 and a second proton, which might be either the HE or HZ protons of residue 86. (The "zero" residue number just serves as a placeholder, so that there will be the same number of columns as for non-ambiguous restraints.) If it is possible that the ambiguous list might not be exhaustive (e.g. if some protons have not been assigned), it is safest to set *ialtd*=1, which will allow "mistakes" to be present in the constraint list. On the other hand, if you want to be sure that every violation is "active", set *ialtd*=0.

If the *-les* flag is set, the program will prepare distance restraints for multiple copies (LES) simulations. In this case, the input PDB file is one *without* LES copies, i.e. with just a single copy of the molecule. The "lesfile" specified by this flag is created by the *addles* program, and contains a mapping from original atom numbers into the copy numbers used in the multiple-copies simulation.

The *-rst* and *-svf* flags specify outputs for *sander*, for distance restraints and NOESY restraints, respectively. In each case, you may need to hand-edit the outputs to add additional parameters. You should make it a habit to compare the outputs with the descriptions given earlier in this chapter to make sure that the restraints are what you want them to be.

It is common to run makeDIST_RST several times, with different inputs that correspond to different spectra, different mixing times, etc. It is then expected that you will manually edit the various output files to combine them into the single file required by *sander*.

12.7.2. Preparing torsion angle restraints: makeANG_RST

There are fewer "standards" for representing coupling constant information. We have followed the DIANA convention in the program *makeANG_RST*. This program takes as input a five-column torsion angle constraint file along with an Amber PDB file of the molecule. It creates as output (to standard out) a list of constraints in RST format that is readable by Amber.

```
Usage: makeANG_RST -help
makeANG_RST -pdb ambpdb_file [-con constraint] [-lib libfile]
[-les lesfile ]
```

The input torsion angle constraint file can be read from standard in or from a file specified by the -con option on the command line. The input constraint file should look something like this:

```
1 GUA PPA 111.5 144.0
2 CYT EPSILN 20.9 100.0
2 CYT PPA 115.9 134.2
3 THY ALPHA 20.4 35.6
4 ADE GAMMA 54.7 78.8
5 GLY PHI 30.5 60.3
6 ALA CHI 20.0 50.0
....
```

Lines beginning with "#" are ignored. The first column is the residue number; the second is the residue name (three letter code, or as defined in your personal torsion library file). Only the first three letters of the residue name are used, so that DIANA files that contain residues like "ASP-" will be correctly interpreted. Third is the angle name (taken from the torsion library described below).The fourth column contains the lower bound, and the fifth column specifies the upper bound. Additional material on the line is (presently) ignored.

Note: It is assumed that the lower bound and the upper bound define a region of allowed conformation on the unit circle that is swept out in a clockwise direction from $lb \rightarrow ub$. If the number in the *lb* column is greater than the the number in the *ub* column, 360°will successively be subtracted from the *lb* until $lb < ub$. This preserves the clockwise definition of the allowed conformation space, while also making the number that specifies the lower bound less than the number that specifies the upper bound, as is required by Amber. If this occurs, a warning message will be printed to *stderr* to notify the user that the data has been modified.

The angles that one can constrain in this manner are defined in the library file that can be optionally specified on the command line with the -lib flag, or the default library "tordef.lib" (written by Garry P. Gippert) will be used. If you wish to specify your own nomenclature, or add angles that are not already defined in the default file, you should make a copy of this file and modify it to suit your needs. The general format for an entry in the library is:

```
LEU PSI N CA C N+
```

where the first column is the residue name, the second column is the angle name that will appear in the input file when specifying this angle, and the last four columns are the atom names that define the torsion angle. When a torsion angle contains atom(s) from a preceding or succeeding residue in the structure, a "-" or "+" is appended to those atom names in the library, thereby specifying that this is the case. In the example above, the atoms that define PSI for LEU residues are the N, CA, and C atoms of that same LEU and the N atom of the residue after that LEU in the primary structure. Note that the order of atoms in the definition is important and should reflect that the torsion angle rotates about the two central atoms as well as the fact that the four atoms are bonded in the order that is specified in the definition.

If the first letter of the second field is "J", this torsion is assumed to be a J-coupling constraint. In that case, three additional floats are read at the end of the line, giving the A,B and C coefficients for the Karplus relation for this torsion. For example:

```
ALA JHNA H N CA HA 9.5 -1.4 0.3
```

will set up a J-coupling restraint for the HN-HA 3-bond coupling, assuming a Karplus relation with A,B, C as 9.5, -1.4 and 0.3. (These particular values are from Brüschweiler and Case, JACS 116: 11199 (1994).)

This program also supports pseudorotation phase angle constraints for prolines and nucleic acid sugars; each of these will generate restraints for the 5 component angles which correspond to the *lb* and *ub* values of the input pseudorotation constraint. In the torsion library, a pseudorotation definition looks like:

```
PSEUDO CYT PPA NU0 NU1 NU2 NU3 NU4
CYT NU0 C4' O4' C1' C2'
CYT NU1 O4' C1' C2' C3'
CYT NU2 C1' C2' C3' C4'
CYT NU3 C2' C3' C4' O4'
CYT NU4 C3' C4' O4' C1'
```

The first line describes that a PSEUDOrotation angle is to be defined for CYT that is called PPA and is made up of the five angles NU0-NU4. Then the definition for NU0-NU4 should also appear in the file in the same format as the example given above for LEU PSI.

PPA stands for Pseudorotation Phase Angle and is the angle that should appear in the input constraint file when using pseudorotation constraints. The program then uses the definition of that PPA angle in the library file to look for the 5 other angles (NU0-NU4 in this case) which it then generates restraints for. PPA for proline residues is included in the standard library as well as for the DNA nucleotides.

If the *-les* flag is set, the program will prepare torsion angle restraints for multiple copies (LES) simulations. In this case, the input PDB file is one *without* LES copies, i.e. with just a single copy of the molecule. The "lesfile" specified by this flag is created by the *addles* program, and contains a mapping from original atom numbers into the copy numbers used in the multiple-copies simulation.

Torsion angle constraints defined here cannot span two different copy sets, i.e., there cannot be some atoms of a particular torsion that are in one multiple copy set, and other atoms from the same torsion that are in other copy sets. It *is* OK to have some atoms with single copies, and others with multiple copies in the same torsion. The program will create as many duplicate torsions as there are copies.

A good alternative to interpreting J-coupling constants in terms of torsion angle restraints is to refine directly against the coupling constants themselves, using an appropriate Karplus relation. See the discussion of the variable RJCOEF, above.

12.7.3. Chirality restraints: makeCHIR_RST

Usage: makeCHIR_RST <pdb-file> <output-constraint-file>

We also find it useful to add chirality constraints and *trans*-peptide ω constraints (where appropriate) to prevent chirality inversions or peptide bond flips during the high-temperature portions of simulated annealing runs. The program *makeCHIR_RST* will create these constraints. Note that you may have to edit the output of this program to change *trans* peptide constraints to *cis*, as appropriate.

12.7.4. Direct dipolar coupling restraints: makeDIP_RST

For simulations with residual dipolar coupling restraints, the *makeDIP_RST.protein*, *makeDIP_RST.dna* and *makeDIP_RST.diana* are simple codes to prepare the input file. Use *-help* to obtain a more detailed description of the usage. For now, this code only handles backbone NH and CαH data. The header specifying values for various parameters needs to be manually added to the output of *makeDIP_RST*.

Use of residual dipolar coupling restraints is new both for Amber and for the general NMR community. Refinement against these data should be carried out with care, and the optimal values for the force constant, penalty function, and initial guesses for the alignment tensor components are still under investigation. Here are some suggestions from the experiences so far:

1. Beware of overfitting the dipolar coupling data in the expense of Amber force field energy. These dipolar coupling data are very sensitive to tiny changes in the structure. It is often possible to drastically improve the fitting by making small distortions in the backbone angles. We recommend inclusion of explicit angle restraints to enforce ideal backbone geometry, especially for those residues that have corresponding residual dipolar coupling data.

2. The initial values for the Cartesian components of the alignment tensor can influence the final structure and alignment if the structure is not fixed (ibelly = 0). For a fixed structure (ibelly = 1), these values do not matter. Therefore, the current "best" strategy is to fit the experimental data to the fixed starting structure, and use the alignment tensor[s] obtained from this fitting as the initial guesses for further refinement.

3. Amber is capable of simultaneously fitting more than one set of alignment data. This allows the use of individually obtained datasets with different alignment tensors. However, if the different sets of data have equal directions of alignment but different magnitudes, using an overall scaling factor for these data with a single alignment tensor could greatly reduce the number of fitting parameters.

4. Because the dipolar coupling splittings depend on the square root of the order parameters ($0 \leq S2 \leq 1$), these order parameters describing internal motion of individual residues are often neglected (N. Tjandra and A. Bax, *Science* **278**, 1111-1113, 1997). However, the square root of a small number can still be noticeably smaller than 1, so this may introduce undesirable errors in the calculations.

12.7.5. Using NMR exchange format (NEF) files

The NMR community, in collaboration with the worldwide PDB, is developing a common format for encoding of NMR restraints, including all of the kinds discussed above. This format is not yet finalized, but we are including here a conversion script, *nef_to_RST*, that would convert these files to *sander* format. Because this format is so new, and is still subject to revisions, care should be taken in using this script: make sure that the

output files do what they should be doing. Here are the usage instructions (which you can also get by typing "nef_to_RST -help" at the command line:

```
# nef_to_RST
convert NEF restraints to Amber format
input:
   -nef <filename>: NEF file
   -pdb <filename>: PDBFILE using AMBER nomenclature and numbering
   -map <filename>: MAP file  (default:map.NEF-AMBER)
output:
   -rst <filename>:  SANDER restraint format
   -rdc <filename>:  SANDER DIP format

other options:
   -nocorr (do not correct upper bound for r**-6 averaging)
   -altdis (use alternative form for the distance restraints)
   -help   (gives you this explanation, overrides other parameters)
   -report (gives you short runtime diagnostic output)
errors come to stderr.
```

12.8. Getting summaries of NMR violations

If you specify LISTOUT=POUT when running *sander*, the output file will contain a lot of detailed information about the remaining restraint violations at the end of the run. When running a family of structures, it can be useful to process these output files with *sviol*, which takes a list of *sander* output files on the command line, and sends a summary of energies and violations to STDOUT. If you have more than 20 or so structures to analyze, the output from *sviol* becomes unwieldy. In this case you may also wish to use *sviol2*, which prints out somewhat less detailed information, but which can be used on larger families of structures. The *senergy* script gives a more detailed view of force-field energies from a series of structures. (We thank the TSRI NMR community for helping to put these scripts together, and for providing many useful suggestions.)

12.9. Time-averaged restraints

The model of the previous sections involves the "single-average-structure" idea, and tries to fit all constraints to a single model, with minimal deviations. A generalization of this model treats distance constraints arising from from NOE crosspeaks (for example) as being the average distance determined from a trajectory, rather than as the single distance derived from an average structure.

Time-averaged bonds and angles are calculated as

$$\bar{r} = (1/C)\left\{\int_0^t e^{(t'-t)/\tau} r(t')^{-i} dt'\right\}^{-1/i} \tag{12.1}$$

where

$\bar{r}$ = time-averaged value of the internal coordinate (distance or angle)

t = the current time

τ = the exponential decay constant

$r(t')$ = the value of the internal coordinate at time t'

i = average is over internals to the inverse of i. Usually i = 3 or 6 for NOE distances, and -1 (linear averaging) for angles and torsions.

C = a normalization integral.

Time-averaged torsions are calculated as

$$< \varphi >= \tan^{-1}(< \sin(\phi) > / < \cos(\phi) >)$$

where ϕ is the torsion, and $< \sin(\phi) >$ and $< \cos(\phi) >$ are calculated using the equation above with $\sin(\phi(t'))$ or $\cos(\phi(t'))$ substituted for $r(t')$.

Forces for time-averaged restraints can be calculated either of two ways. This option is chosen with the DISAVI / ANGAVI / TORAVI commands. In the first (the default),

$$\partial E/\partial x = (\partial E/\partial \bar{r})(\partial \bar{r}/\partial r(t))(\partial r(t)/\partial x) \tag{12.2}$$

(and analogously for y and z). The forces then correspond to the standard flat-bottomed well functional form, with the instantaneous value of the internal replaced by the time-averaged value. For example, when $r_3 < \bar{r} < r_4$,

$$E = k_3(\bar{r} - r_3)^2$$

and similarly for other ranges of $\bar{r}$.

When the second option for calculating forces is chosen (IINC = 1 on a DISAVI, ANGAVI or TORAVI card), forces are calculated as

$$\partial E/\partial x = (\partial E/\partial \bar{r})(\partial r(t)/\partial x) \tag{12.3}$$

For example, when $r_3 < \bar{r} < r_4$,

$$\partial E/\partial x = 2k_3(\bar{r} - r_3)(\partial r(t)/\partial x)$$

Integration of this equation does not give Eq. 12.2, but rather a non-intuitive expression for the energy (although one that still forces the bond to the target range). The reason that it may sometimes be preferable to use this second option is that the term $\partial \bar{r}/\partial r(t)$, which occurs in the exact expression [Eq. 12.2], varies as $(\bar{r}/r(t))^{1+i}$. When i=3, this means the forces can be varying with the fourth power the distance, which can possibly lead to very large transient forces and instabilities in the molecular dynamics trajectory. [Note that this will not be the case when linear scaling is performed, i.e. when $i = -1$, as is generally the case for valence and torsion angles. Thus, for linear scaling, the default (exact) force calculation should be used].

It should be noted that forces calculated using Eq. 12.3 are not conservative forces, and would cause the system to gradually heat up, if no velocity rescaling were performed. The temperature coupling algorithm should act to maintain the average temperature near the target value. At any rate, this heating tendency should not be a problem in simulations, such as fitting NMR data, where MD is being used to sample conformational space rather than to extract thermodynamic data.

This section has described the methods of time-averaged restraints. For more discussion, the interested user is urged to consult studies where this method has been used.[272–276]

12.10. Multiple copies refinement using LES

NMR restraints can be made compatible with the multiple copies (LES) facility; see the following chapter for more information about LES. To use NMR constraints with LES, you need to do two things:

(1) Add a line like "file wnmr name=(lesnmr) wovr" to your input to *addles*. The filename (lesnmr in this example) may be whatever you wish. This will cause *addles* to output an additional file that is needed at the next step.

(2) Add "-les lesnmr" to the command line arguments to *makeDIST_RST*. This will read in the file created by *addles* containing information about the copies. All NMR restraints will then be interpreted as "ambiguous" restraints, so that if any of the copies satisfies the restraint, the penalty goes to zero.

Note that although this scheme has worked well on small peptide test cases, we have yet not used it extensively for larger problems. This should be treated as an experimental option, and users should use caution in applying or

interpreting the results.

12.11. Some sample input files

The next few pages contain excerpts from some sample NMR refinement files used at TSRI. The first example just sets up a simple (but often effective) simulated annealing run. You may have to adjust the length, temperature maximum, etc. somewhat to fit your problem, but these values work well for many "ordinary" NMR problems.

12.11.1. 1. Simulated annealing NMR refinement

```
15ps simulated annealing protocol
&cntrl
  nstlim=15000, ntt=1, !(time limit, temp. control)
  ntpr=500, pencut=0.1, !(control of printout)
  ipnlty=1, nmropt=1, !(NMR penalty function options)
  vlimit=10, !(prevent bad temp. jumps)
  ntb=0, !(non-periodic simulation)
  igb=8, !(generalize Born solvent model)
/
#
# Simple simulated annealing algorithm:
#
# from steps 0 to 1000: raise target temperature 10-1200K
# from steps 1000 to 3000: leave at 1200K
# from steps 3000 to 15000: re-cool to low temperatures
#
&wt type='TEMP0', istep1=0,istep2=1000,value1=10.,
  value2=1200., /
&wt type='TEMP0', istep1=1001, istep2=3000, value1=1200.,
  value2=1200.0, /
&wt type='TEMP0', istep1=3001, istep2=15000, value1=0.,
  value2=0.0, /
#
# Strength of temperature coupling:
# steps 0 to 3000: tight coupling for heating and equilibration
# steps 3000 to 11000: slow cooling phase
# steps 11000 to 13000: somewhat faster cooling
# steps 13000 to 15000: fast cooling, like a minimization
#
&wt type='TAUTP', istep1=0,istep2=3000,value1=0.2,value2=0.2, /
&wt type='TAUTP', istep1=3001,istep2=11000,value1=4.0,value2=2.0, /
&wt type='TAUTP', istep1=11001,istep2=13000,value1=1.0,value2=1.0, /
&wt type='TAUTP', istep1=13001,istep2=14000,value1=0.5,value2=0.5, /
&wt type='TAUTP', istep1=14001,istep2=15000,value1=0.05,value2=0.05, /
#
# "Ramp up" the restraints over the first 3000 steps:
#
&wt type='REST', istep1=0,istep2=3000,value1=0.1,value2=1.0, /
&wt type='REST', istep1=3001,istep2=15000,value1=1.0,value2=1.0, /
&wt type='END' /
LISTOUT=POUT (get restraint violation list)
DISANG=RST.f (file containing NMR restraints)
```

The next example just shows some parts of the actual RST file that *sander* would read. This file would ordinarily *not* be made or edited by hand; rather, run the programs *makeDIST_RST*, *makeANG_RST* and *makeCHIR_RST*, combining the three outputs together to construct the RST file.

12.11.2. Part of the RST.f file referred to above

```
# first, some distance constraints prepared by makeDIST_RST:
 # (comment line is input to makeRST, &rst namelist is output)
 #
 #( proton 1 proton 2 upper bound)
 #-----------------------------------------
 #
 # 2 ILE HA 3 ALA HN 4.00
 #
 &rst iat= 23, 40, r3= 4.00, r4= 4.50,
 r1 = 1.3, r2 = 1.8, rk2=0.0, rk3=32.0, ir6=1, /
 #
 # 3 ALA HA 4 GLU HN 4.00
 #
 &rst iat= 42, 50, r3= 4.00, r4= 4.50, /
 #
 # 3 ALA HN 3 ALA MB 5.50
 #
 &rst iat= 40, -1, r3= 6.22, r4= 6.72,
 igr1= 0, 0, 0, 0, igr2= 44, 45, 46, 0, /
 #
 # .......etc......
 #
 # next, some dihedral angle constraints, from makeANG_RST:
 #
 &rst iat= 213, 215, 217, 233, r1=-190.0,
 r2=-160.0, r3= -80.0, r4= -50.0, /
 &rst iat= 233, 235, 237, 249, r1=-190.0,
 r2=-160.0, r3= -80.0, r4= -50.0, /
 # .......etc.......
 #
 # next, chirality and omega constraints prepared by makeCHIR_RST:
 #
 #
 # chirality for residue 1 atoms: CA CG HB2 HB3
 &rst iat= 3 , 8 , 6 , 7 ,
 r1=10., r2=60., r3=80., r4=130., rk2 = 10., rk3=10., /
 #
 # chirality for residue 1 atoms: CB SD HG2 HG3
 &rst iat= 5 , 11 , 9 , 10 , /
 #
 # chirality for residue 1 atoms: N C HA CB
 &rst iat= 1 , 18 , 4 , 5 , /
 #
 # chirality for residue 2 atoms: CA CG2 CG1 HB
 &rst iat= 22 , 26 , 30 , 25 , /
 #
 ......etc........
 # trans-omega constraint for residue 2
 &rst iat= 22 , 20 , 18 , 3 ,
 r1=155., r2=175., r3=185., r4=205., rk2 = 80., rk3=80., /
 #
```

```
 # trans-omega constraint for residue 3
 &rst iat= 41 , 39 , 37 , 22 , /
 #
 # trans-omega constraint for residue 4
 &rst iat= 51 , 49 , 47 , 41 , /
 #
 # ......etc........
 #
 The next example is an input file for volume-based NOE refinement. As with the distan
```

12.11.3. 3. Sample NOESY intensity input file

```
# A part of a NOESY intensity file:
 &noeexp
 id2o=1, (exchangeable protons removed)
 oscale=6.21e-4, (scale between exp. and calc. intensity units)
 taumet=0.04, (correlation time for methyl rotation, in ns.)
 taurot=4.2, (protein tumbling time, in ns.)
 NPEAK = 13*3, (three peaks, each with 13 mixing times)
 EMIX = 2.0E-02, 3.0E-02, 4.0E-02, 5.0E-02, 6.0E-02,
 8.0E-02, 0.1, 0.126, 0.175, 0.2, 0.25, 0.3, 0.35,
 (mixing times, in sec.)
 IHP(1,1) = 13*423, IHP(1,2) = 13*1029, IHP(1,3) = 13*421,
 (number of the first proton)
 JHP(1,1) = 78*568, JHP(1,2) = 65*1057, JHP(1,3) = 13*421,
 (number of the second proton)
 AEXP(1,1) = 5.7244, 7.6276, 7.7677, 9.3519,
 10.733, 15.348, 18.601,
 21.314, 26.999, 30.579,
 33.57, 37.23, 40.011,
 (intensities for the first cross-peak)
 AEXP(1,2) = 8.067, 11.095, 13.127, 18.316,
 22.19, 26.514, 30.748,
 39.438, 44.065, 47.336,
 54.467, 56.06, 60.113,
 AEXP(1,3) = 7.708, 13.019, 15.943, 19.374,
 25.322, 28.118, 35.118,
 40.581, 49.054, 53.083,
 56.297, 59.326, 62.174,
 /
 SUBMOL1
 RES 27 27 29 29 39 41 57 57 70 70 72 72 82 82 (residues in this submol)
END END
```

Next, we illustrate the form of the file that holds residual dipolar coupling restraints. Again, this would generally be created from a human-readable input using the program *makeDIP_RST*.

12.11.4. Residual dipolar restraints, prepared by makeDIP_RST:

```
&align
ndip=91, dcut=-1.0, gigj = 37*-3.1631, 54*7.8467,
s11=3.883, s22=53.922, s12=33.855, s13=-4.508, s23=-0.559,
id(1)=188, jd(1)=189, dobsu(1)= 6.24, dobsl(1)= 6.24,
```

```
    id(2)=208, jd(2)=209, dobsu(2)= -10.39, dobsl(1)= -10.39,
    id(3)=243, jd(3)=244, dobsu(3)= -8.12, dobsl(1)= -8.12,
    ....
    id(91)=1393, jd(91)=1394, dobsu(91)= -19.64, dobsl(91) = -19.64,
    /
```

Finally, we show how the detailed input to *sander* could be used to generate a more complicated restraint. Here is where the user would have to understand the details of the RST file, since there are no "canned" programs to create this sort of restraint. This illustrates, though, the potential power of the program.

12.11.5. A more complicated constraint

```
# 1) Define two centers of mass. COM1 is defined by
# {C1 in residue 2; C1 in residue 3; N2 in residue 4; C1 in residue 5}.
# COM2 is defined by {C4 in residue 1; O4 in residue 1; N* in residue 1}.
# (These definitions are effected by the igr1/igr2 and grnam1/grnam2
# variables; You can use up to 200 atoms to define a center-of-mass
# group)
#
# 2) Set up a distance restraint between COM1 and COM2 which goes from a
# target value of 5.0A to 2.5A, with a force constant of 1.0, over steps 1-5000.
#
# 3) Set up a distance restraint between COM1 and COM2 which remains fixed
# at the value of 2.5A as the force slowly constant decreases from
# 1.0 to 0.01 over steps 5001-10000.
#
# 4) Sets up no distance restraint past step 10000, so that free (unrestrained)
# dynamics takes place past this step.
#
&rst iat=-1,-1, nstep1=1,nstep2=5000,
  iresid=1,irstyp=0,ifvari=1,ninc=0,imult=0,ir6=0,ifntyp=0,
  r1=0.00000E+00,r2=5.0000,r3=5.0000, r4=99.000,rk2=1.0000,rk3=1.0000,
  r1a=0.00000E+00,r2a=2.5000,r3a=2.5000, r4a=99.000,rk2a=1.0000,rk3a=1.0000,
  igr1 = 2,3,4,5,0, grnam1(1)='C1',grnam1(2)='C1',grnam1(3)='N2',
  grnam1(4)='C1', igr2 = 1,1,1,0, grnam2(1)='C4',grnam2(2)='O4',grnam2(3)='N*',
/
&rst iat=-1,-1, nstep1=5001,nstep2=10000,
  iresid=1,irstyp=0,ifvari=1,ninc=0,imult=0,ir6=0,ifntyp=0,
  r1=0.00000E+00,r2=2.5000,r3=2.5000, r4=99.000,rk2=1.0000,rk3=1.0000,
  r1a=0.00000E+00,r2a=2.5000,r3a=2.5000, r4a=99.000,rk2a=1.0000,rk3a=0.0100,
  igr1 = 2,3,4,5,0, grnam1(1)='C1',grnam1(2)='C1',grnam1(3)='N2',
  grnam1(4)='C1', igr2 = 1,1,1,0, grnam2(1)='C4',grnam2(2)='O4',grnam2(3)='N*',
/
```

13. Xray and cryoEM refinement

13.1. EMAP restraints for rigid and flexible fitting into EM maps

EMAP restrained simulation[167, 277] was developed to incorporate electron microscopy (EM) image information into macromolecular structure determination. Different from NMR and X-ray data, EM images have low resolutions (5~50Å). However, EM images of large molecular assemblies up to millions of atoms and in various biologically relevant environments are available. These low resolution images provide precious structural information that can help to determine structures of many molecular assemblies and machineries[277–287].

With EMAP restraints, Sander and PMEMD can be used to perform both rigid[277] and flexible[167] fitting of molecules into experimental maps of complexes to obtain both complex structures and conformations agreeing with experimental maps. In addition to experimental map information, homologous structural information can be used by EMAP to perform targeted conformational search (TCS) to induce simulation systems to form structures of interest.

If the restraint map or structure is very different from the starting conformation, SGLD is recommended to induce large conformational change by setting *isgld*=1. This is often used to simulate conformational transition between different states. See the Sampling and free energy search section 10.1 for details on running SGLD.

If domain motion is desired while domain structures need to be maintained, one can use an EMAP restraint generated from the initial coordinates for each domain and set *move*=1 to allow the restraint map to move with the domain, so that domains can search the conformational space without unfolding or changing shape.

Each EMAP restraint is defined by a map file and a selection of atoms, as well as related parameters. Multiple EMAP restraints can be defined. The map can be either input from an image file, or generated from a pdb structure or derived from the starting coordinates. The definition of EMAP restraints are read in from the input file as "&emap" namelists. The following are variables in each &emap namelist.

`mapfile` The filename of a restraint map or structure. The restraint maps must be in "map", "ccp4", or "mrc" format. The structure must be in pdb format. The structure need not be the same as the simulation system. A resolution can be specified for the conversion to a density map. When a blank filename is specified, *mapfile*=", the input coordinates of the masked atoms will be used to generate a restraint map (default=").

`atmask` The atom mask for selecting atoms to be restrained (default=':*').

`fcons` The restraining constant (default=0.05 kcal/g).

`move` Allow the restraint map to move when *move*>0 (default=0).

`resolution` The resolution used to convert an atomic structure to a map (default=2 Å).

`ifit` Perform rigid fitting before simulation when *ifit*>0. One would do this when the initial coordinates don't match those of the map (default=0). When ifit=1, the map is transformed (by translation and rotation) to match the coordinates; the coordinates are not altered. EMAP allows output of the re-oriented map (*mapfit*=...) that matches the (final) simulation coordinates, *and/or* output of the coordinates (*molfit*=...) that would match the orientation of the original map. When ifit=2, the masked atoms will be transformed to fit the map and the transformed coordinates will be used for the following simulation. For periodic systems, ifit=2 may cause atoms to clash with periodic image atoms.

`grids` Grid numbers in x,y,z,phi,psi,theta dimensions for grid-threading rigid fitting[277]. For example, grids=2,2,2,3,3,3 defines 2 grid points in each of x,y,z directions between the minimum and maximum coordinates, and 3 grid points in each of phi (0-360), psi(0-360), theta(0-180) angles. A

search for local minimums starts from every grid point and the global minimum is identified from all the local minimums (default=1,1,1,1,1,1).

`mapfit` The filename for the final constraint map after rigid fitting and/or moving. The filename must has an extension of .map, .ccp4, or .mrc (default=″, for no map output).

`molfit` The filename for the final restrained atom coordinates after rigid fitting and/or simulation. The filename must have an extension of .pdb (default=″, for no structure output).

Here is an example input file for an EMAP constrained SGLD simulation:

```
Map Constraint Self-Guided Langevin dynamics
&cntrl ntx=1, ntb=0,nstlim=100000, imin=0, maxcyc=1, ntc=2, ntf=2, cut=9.0,
ntpr=1000, ntwr=100000,ntwx=10000, ntt=3, gamma_ln=10.0,nscm=100, dt=0.001,
ntb=0,igb=0,ips=1, isgld=1, tsgavg=1.0, sgft=0.5,tempsg=0,          (SGLD)
iemap=1,                    (turn on EMAP )
/
&emap           (EMAP restraint 1 )
mapfile='data/1gb1.ccp4',   (map is input from a map file)
atmask=':1-20',             (residues 1-20 are restrained)
fcons=0.1,
move=1,                     (restraint map can move)
ifit=1,                     (perform rigid fitting first)
mapfit='scratch/gb1n_1.ccp4',  (final map)
molfit='scratch/gb1n_1.pdb', / (final restrained atoms related to initial map)
&emap           (EMAP restraint 2)
mapfile='data/1gb1.pdb',   (map is generated from a pdb file)
atmask=':22-37',           (residues 22-37 are restrained)
fcons=0.1,move=0,          (restraint map is fixed)
ifit=1,                    (perform rigid fitting first)
mapfit='scratch/gb1h_1.ccp4',   (final map, same as initial)
molfit='scratch/gb1h_1.pdb', / (final restrained atoms related to initial map)
&emap           (EMAP restraint 3)
mapfile='',     (map is generated from initial coordinates)
atmask=':41-56',  (residues 41-56 are restrained)
fcons=0.1,move=1, (restraint map can move)
ifit=1,        (perform rigid fitting first)
mapfit='scratch/gb1c_1.ccp4',  (final map)
molfit='scratch/gb1c_1.pdb', / (final restrained atoms related to initial map)
```

13.2. X-ray functionality and diffraction-based restraints

New to Amber 20 and updated in Amber 22, the *pmemd* and *pmemd.cuda* programs include an experimental module dedicated to biomolecular crystallography. It is envisioned that in future Amber can be used as a platform to address various crystallography-related problems, e.g. to refine crystallographic structures of proteins and nucleic acids (similar to the existing capability in the area of biomolecular NMR). This module is intended for use with an MD simulation of the crystal unit cell or a "supercell"[288]. For information on how to set up a crystal simulation, including periodic boundary conditions to emulate crystalline lattice, see Chapter 7. It is expected that the crystal is solvated using an explicit (or implicit) solvent. While a number of crystallographic concepts are implemented in the new Amber module, many others have not yet been implemented. For example, an MD model of a crystal unit cell can naturally accommodate different side-chain conformations; however, the concept of alternate side-chain conformations, as employed in protein crystallography, is currently unavailable in Amber.

Although it is not a part of Amber, it is worth noting that the *phenix* crystallographic package now allows for X-ray refinement using Amber (or other) force fields[289]. This was accomplished by using the python API to *sander*, discussed in Section 8.13, and uses locally-enhanced sampling (see Chap. 14) to handle alternate conformations.

It supports all of the X-ray related options in *phenix.refine*, but has limited options for molecular dynamics, and no GPU acceleration.

13.2.1. Structure factor calculations

For the crystal simulation, the program can calculate crystallographic structure factors (SFs) for individual MD frames. The calculations are conducted using direct summation formula[290]; a mask is available to define the subset of atoms included in these calculations (*atom_selection_mask*). For example, this mask could select the macromolecules, but not the solvent or the neutralizing ions present in the simulation. The B-factors used in the direct summation formula are supplied through a designated PDB file. The set of Miller indices for SF calculations is supplied as a part of the *reflection_infile*.

In principle, explicit solvent and ions can also be accounted for via the direct summation formula. However, any single individual frame does not offer an adequate statistical sampling with regard to the positioning of water molecules (if desired, such statistical sampling can be obtained by modeling of a very large supercell or otherwise by means of time averaging). As a commonly accepted alternative, Amber 22 offers two mask-based models of bulk solvent. The first one (*bulk_solvent_model* = 'simple') is a simple variant of flat mask bulk solvent model[291], which calculates the contribution from interstitial solvent into SFs using two generic parameters (*k_sol* and *b_sol*), as reported by Fokine and Urzhumtsev[292]. The implementation, including the scheme to build solvent mask, is analogous to the one in cctbx library[293]. The more advanced version (*bulk_solvent_model* = 'afonine-2013') employs the varaible k_{mask}, which replaces *k_sol* and *b_sol*; this variable is automatically optimized over the individual resolution bins[294]. The iterative optimization procedure to determinate k_{mask} also adjusts the overall scaling coefficient that is applied to calculated SFs. The implementation follows the one in cctbx library with several minor modifications.

It is worth noting that crystal MD simulations in Amber (as described in Chap. 7) do not maintain a perfect space group symmetry. Therefore, strictly speaking, the calculated SFs correspond to P(1) space group with the unit cell that is identical to the simulation box. During the course of the simulation, the calculated SFs can be collected frame-by-frame at a specified interval (*ntwsf*) and stored in a form of special trajectory file (*sf_outfile*).

13.2.2. Structure-factor-based potential

The X-ray energy term E_{xray} is added to the total potential energy with the user-specified weight w_{xray} (controlled by *xray_weight_initial* / *xray_weight_final* variables):

$$E_{total} = E_{force-field} + w_{xray} E_{xray} \tag{13.1}$$

E_{xray} uses the set of target (i.e. experimentally observed) SFs, which are supplied via the input file *reflection_infile*. This file must also contain flags to divide all reflections into a "working" set and a "free" (test) set. Currently *pmemd* offers two variants of the E_{xray} term. The first one is a very simple least squares objective function (*target* = 'ls') involving the aplitudes and of calculated and observed structure factors:

$$E_{xray} = \frac{\sum_{h,k,l} \left(F_{calc}(h,k,l) - F_{obs}(h,k,l)\right)^2}{\sum_{h,k,l} F_{obs}^2(h,k,l)} \tag{13.2}$$

The sum in this expression is over the working set of reflections.

The second option for E_{xray} is the Maximum Likelihood target function (*target* = 'ml')[295, 296]:

$$E_{xray} = \sum_{h,k,l} \left(-\ln\left(\frac{2F_{obs}(h,k,l)}{\varepsilon\beta}\right) + \frac{F_{obs}^2(h,k,l)}{\varepsilon\beta} + \frac{\alpha^2 F_{calc}^2(h,k,l)}{\varepsilon\beta} - \ln I_0\left(\frac{2\alpha F_{obs}(h,k,l) F_{calc}(h,k,l)}{\varepsilon\beta}\right)\right) \tag{13.3}$$

where α and β are (resolution-shell-dependent) ML likelihood distribution parameters, ε is the symmetry coefficient ($\varepsilon = 1$ for the space group P(1) at hand), $I_0(x)$ is the zeroth-order modified Bessel function of the first kind, and the summation is over the working set of reflections. For practical applications, we recommend using *target* = 'ml' along with the more advanced version of solvent, *bulk_solvent_model* = 'afonine-2013'.

The expression for E_{xray}, along with the direct-summation formula for F_{calc}, provides a basis to evaluate forces. These "restraint" forces act like those used in NMR refinement, discussed in Chap.12, and are generally used to drive minimization or MD simulations that minimize E_{total}. The value of E_{xray} is reported in the mdout file, together with R_{work} and R_{free}.

We envisage that SF-based restraints can be used for a number of purposes. For example, they can be viewed as an empirical addition to the force fields, which can potentially remedy certain existing biases[297, 298]. Another promising application is refinement of crystallographic structures. Such an Amber-based protocol has been reported by Mikhailovskii et al.[299] (the web interface is available at https://arx.bio-nmr.spbu.ru). Ultimately, the entire process of crystallographic structure determination can be incorporated into Amber. This approach may be particularly valuable for lower-quality diffraction data sets and incomplete structural models (e.g. in the case of weak or missing electron density for mobile side chains, loops or terminal regions in protein molecules). In this situation, the state-of-the-art force field provides a natural solution to model the poorly resolved or unresolved elements of the structure. This is accomplished in a highly realistic manner, by using the explicit representation of the crystal unit cell (supercell), taking into consideration the effect of solvent, crystal contacts, etc.

13.2.3. Inputs and file formats

System setup follows the general procedures outlined in Chap. 7. For users with access to the *phenix* package of crystallographic analysis tools, the XrayPrep tool can prepare the system: inputs are simply a PDB file (*xxxx.pdb,* where xxxx is a PDB id) and the corresponding structure factor file (*xxxx-sf.cif*).

For those who will prepare their own inputs, one needs a PDB file, expanded to the unit cell (see Chap. 7) that contains the B-factors. The structure factors have to be listed in the r*eflection_infile*, which is a human-readable ascii file containing the same information that can be normally found in .mtz files. The first line contains a total number of reflections followed by a zero. Subsequent lines list Miller indices *h, k* and *l*, followed by the respective SF values and their standard deviations, followed by an R-free flag (we adopt the convention that "1" indicates a member of the working set, and "0" a member of the test set). An example file is given below. Note that column spacing or number formatting is not critical, but each entry should be separated by at least one space.

```
41243 0
-19        -6        1        13.86329        9.685285        0
-19        -6        2        46.38137        3.528763        1
-19        -5        1        9.675193        21.28529        1
...
 19         6        1        13.86329        9.685285        0
 19         6        2        46.38137        3.528763        1
```

Input variables in the &xray namelist The X-ray functionalities are activated by adding the *&xray* namelist to the mdin file. User-assigned parameters that are not used by the algorithm are silently ignored(e.g.\ *k_sol* and *b_sol* in 'afonine-2013' bulk solvent model). The keywords in *&xray* namelist include the following:

File handling:

`pdb_infile`	name of the PDB input file containing B-factors
`pdb_read_coordinates`	if true, use coordinates from the PDB file, not inpcrd, as starting coordinates
`pdb_outfile`	name of PDB file to write the final atomic coordinates from the simulation. Currently writes back the input B-factors and occupancies as read from *pdb_infile*
`reflection_infile`	name of the input file containing experimental SFs and R-flags
`sf_outfile`	name of the trajectory file with calculated SFs
`ntwsf`	time interval to write calculated SFs to *sf_outfile*

Bulk solvent parameters:

`bulk_solvent_model`	the type of bulk solvent to use. Possible values: 'none' for disabled bulk solvent contribution, 'simple' or 'afonine-2013' (default)
`k_sol`	solvent electron density (default 0.35 e $Å^{-3}$)
`b_sol`	determines the blurring of the boundary between the solvent region and the macromolecule (default 46 $Å^2$)
`solvent_mask_adjustment`	increment to be added to atomic radii of the atoms selected by *atom_selection_mask* as a part of the algorithm to build bulk solvent mask (default 1.11 Å)
`solvent_mask_probe_radius`	the radius of solvent probe to apply as a part of the algorithm to build bulk solvent mask (default 0.9 Å)
`mask_update_period`	time interval for bulk solvent grid update (expressed as a multiple of integration step, default 100 steps)

Structure-factor-based potential:

`atom_selection_mask`	ambmask-format mask to specify the atoms that contribute to F_{calc} via direct summation formula (default '!@H=')
`scale_update_period`	time interval to re-scale F_{calc} to F_{obs} (expressed as a multiple of integration step, default 100 steps)
`target`	the type of crystallographic target function. Possible values: 'ls' for Least Squares, 'ml' for Maximum Likelihood (default)
`ml_update_period`	time interval to update ML parameters α and β (expressed as a multiple of integration step, default 100 steps)
`xray_weight_initial`	initial value that defines linear scaling of w_{xray}weight along the trajectory (default 1.0)
`xray_weight_final`	final value that defines linear scaling of w_{xray} weight along the trajectory (if unassigned, assumed to be equal to *xray_weight_initial*)

14. Locally-enchanced sampling

Locally-enhanced sampling (LES) is a method to allow for multiple local copies of regions within a larger biomolecule. An example would be to allow sidechains in a protein to be "disordered" (that is, to be described as a superposition of several configurations), while the backbone is represented as a single configuration. This is similar to the "alternate conformer" model often used by crystallographers to describe local disorder in proteins. As the method name implies, this method can achieve enhanced sampling compared to conventional MD. Explanations of the approach, along with key examples, can be found in early, seminal papers.[300–303]

The LES functionality for *sander* was written by Carlos Simmerling. It basically functions by modifying the *prmtop* file using the program addles. The modified *prmtop* file is then used with a slightly modified version of *sander* called *sander.LES*.

14.1. Preparing to use LES with Amber

The first decision that must be made is whether LES is an appropriate technique for the system that you are studying. For further guidance, you may wish to consult published articles to see where LES has proven useful in the past. Several examples will also be given at the end of this section in order to provide models that you may wish to follow.

There are three main issues to consider before running the ADDLES module of Amber.

1. What should be copied?
2. How many copies should be used?
3. How many regions should be defined?

A brief summary of my experience with LES follows.

1. You should make copies of flexible regions of interest. This sounds obvious, and in some cases it is. If you are interested in determining the conformation of a protein loop, copy the loop region. If you need to determine the position of a side chain in a protein after a single point mutation, copy that side chain. If the entire biomolecule needs refinement, then copy the entire molecule. Some other cases may not be obvious- you may need to decide how far away from a particular site structural changes may propagate, and how far to extend the LES region.

2. You should use as few copies as are necessary. While this doesn't sound useful, it illustrates the general point–too few copies and you won't get the full advantages of LES, and too many will not only increase your system size unnecessarily but will also flatten the energy surface to the point where minima are no longer well defined and a wide variety of structures become populated. In addition, remember that LES is an approximation, and more copies make it more approximate. Luckily, published articles that explore the sensitivity of the results to the number of copies show that 3-10 copies are usually reasonable and provide similar results, with 5 copies being a good place to start.

3. Placing the divisions between regions can be the most difficult choice when using LES. This is essentially a compromise between surface smoothing and copy independence. The most effective surface-smoothing in LES takes places between LES regions. This is because Na copies in region A interact with all Nb copies in region B, resulting in Na*Nb interactions, with each scaled by 1/(Na*Nb) compared to the original interaction. This is better both from the statistics of how many different versions of this interaction contribute to the LES average, and how much the barriers are reduced. Remember that since the copies of a given region do not interact with different copies of that same region, interactions inside a region are only scaled by 1/N.

The other thing to consider is whether these enhanced statistics are actually helpful. For example, if the copies cannot move apart, you will obtain many copies of the same conformation–obviously not very helpful. This will also result in less effective reduction in barriers, since the average energy barriers will be very similar to the non-average barrier. The independence of the copies is also related to how the copies are attached. For example, different copies of an amino acid side chain are free to rotate independently (at least within restrictions imposed by the surroundings and intrinsic potential) and therefore each side chain in the sequence could be placed into a separate LES region. If you are interested in backbone motion, however, placing each amino acid into a separate region is not the best choice. Each copy of a given amino acid will be bonded to the neighbor residues on each side. This restriction means that the copies are not very independent, since the endpoints for each copy need to be in nearly the same places. A better choice is to use regions of 2-4 amino acids. As the regions get larger, each copy can start to have more variety in conformation- for example, one segment may have some copies in a helical conformation while others are more strand-like or turn-like. The general rule is that larger regions are more independent, though you need to consider what types of motions you expect to see.

The best way to approach the division of the atoms that you wish to copy into regions is to make sure that you have several LES regions (unless you are copying a very small region such as a short loop or a small ligand). This will ensure plenty of inter-copy averaging. Larger regions permit wider variations in structure, but result in less surface smoothing. A subtle point should be addressed here- the statistical improvement available with LES is not a benefit in all cases and care must be taken in the choice of regions. For example, consider a ligand exiting a protein cavity in which a side chain acts as a *gate* and needs to move before the ligand can escape. If we make multiple copies of the gate, and do not copy the ligand, the ligand will interact in an average way with the *gates*. If the gate was so large that even the softer copies can block the exit, then the ligand would have to wait until ALL of the gate copies opened in order to exit. This may be more statistically difficult than waiting for the original, single gate to open despite the reduced barriers. Another way to envision this is to consider the ligand trying to escape against a true probability distribution of the gate- if it was open 50% of the time and closed 50%, then the exit may still be completely blocked. Continuum representations are therefore not always the best choice.

Specific examples will be given later to illustrate how these decisions can be made for a particular system.

14.2. Using the ADDLES program

The ADDLES module of Amber is used to prepare input for simulations using LES. A non-LES prmtop and prmcrd file are generated using a program such as LEaP. This prmtop file is then given to ADDLES and replaced by a new prmtop file corresponding to the LES system. All residues are left intact- copies of atoms are placed in the same residue as the original atom, so that analysis based on sequence is preserved. Atom numbering is changed, but atom names are unchanged, meaning that a given residue may have several atoms with the same name. A different program is available for taking this new topology file and splitting the copies apart into separate residues, if desired. All copies are given the same coordinates as in the input coordinate file for the non-LES system.

Using addles:

```
addles < inputfile > outputfile
```

SAMPLE INPUT FILE:

```
~ a line beginning with ~ is a comment line.
~ all commands are 4 letters.
~ the maximum line length is 80 characters;
~ a trailing hyphen, "-", is the line continuation token.
~ use 'file' to specify an input/output file, then the type of file
 'rprm' means this is the file to read the prmtop
~ the 'read' means it is an input file
~
file rprm name=(solv200.topo) read
~
~ 'rcrd' reads the original coordinates- optional, only if you want
```

```
~ a set of coords for the new topology
~ you can also use 'rcvd' for coords+velocities, 'rcvb' for coords,
~ velos and box dimensions, 'rcbd' for coords and box dimensions.
~ use "pack=n" option to read in multiple sets of coordinates and
~ assign different coordinates to different copies.
file rcrd name=(501v200.coords) read
~ 'wprm' is the new topology file to be written. the 'wovr' means to
~ write over the file if it exists, 'writ' means don't write over.
file wprm name=(lesparm) wovr
~ 'wcrd is for writing coords, it will automatically write velo and box
~ if they were read in by 'rcvd' or 'rcvb'
file wcrd name=(lescrd) wovr
~ now put 'action' before creating the subspaces
action
~ the default behavior is to scale masses by 1/N.
~ omas leaves all masses at the original values
omas
~ now we specify LES subspaces using the 'spac' keyword, followed
~ by the number of copies to make and then a pick command to tell which
~ atom to copy for this subspace
~ 3 copies of the fragment consisting of monomers (=residues) 1 and 2
spac numc=3 pick #mon 1 2 done
~ 3 copies of the fragment consisting of monomers 3 and 4
spac numc=3 pick #mon 3 4 done
~ 3 copies of the fragment consisting of residues 5 and 6
spac numc=3 pick #mon 5 6 done
~ 2 copies of the side chain on residue 1
~ note that this replaces each of the side chains ON EACH OF THE 3
~ COPIES MADE ABOVE with 2 copies - net 6 copies
~ each of the 3 copies of residue 1-2 has 2 side chain copies.
~ the '#sid' command picks all atoms in the residue except
~ C,O,CA,HA,N,H and HN.
spac numc=2 pick #sid 1 1 done
spac numc=2 pick #sid 2 2 done
spac numc=2 pick #sid 3 3 done
spac numc=2 pick #sid 4 4 done
spac numc=2 pick #sid 5 5 done
~ use the *EOD to end the input
*EOD
```

What this does: all of the force constants are scaled in the new prmtop file by 1/N for N copies, so that this scaling does not need to be done for each pair during the nonbond calculation. Charges and VDW epsilon values are also scaled. New bond, angle, torsion and atom types are created. Any of the original types that were not used are discarded. Since each LES copy should not interact with other copies of the SAME subspace, the other copies are placed in the exclusion list. If you define very large LES regions, the exclusion list will get large and you may have trouble with the fixed length for this entry in the prmtop file- currently 8 digits.

The coordinates are simply copied - that means that all of the LES copies initially occupy the same positions in space. In this setup, the potential energy should be identical to the original system- this is a good test to make sure everything is functioning properly. Do a single energy evaluation of the LES system and the original system, using the copied coordinate file. All terms should be nearly identical (to within machine precision and roundoff). With PME on non- neutral systems, all charges are slightly modified to neutralize the system. For LES, there are a different number of atoms than in the original system, and therefore this charge modification to each atom will differ from the non-LES system and electrostatic energies will not match perfectly.

IMPORTANT: After creating the LES system, the copies will all feel the same forces, and since the coordinates are identical, they will move together unless the initial velocities are different. If you are initializing velocities using INIT=3 and TEMPI>0, this is not a problem. In order to circumvent this problem, addles slightly (and randomly) modifies the copy velocities if they were read from the coordinate input file. If the keyword "nomodv" is specified, the program will leave all of the velocities in the same values as the original file. If you do not read velocities, make sure to assign an initial nonzero temperature to the system. You should think about this and change the behavior to suit your needs. In addition, the program scales the velocities by sqrt(N) for N copies to maintain the correct thermal energy (mv2), but only when the masses are scaled (not using omas option). Again, this requires some thought and you may want different behavior. Regardless of what options are used for the velocities, further equilibration should be carried out. These options are simple attempts to keep the system close to the original state.[304]

Sometimes it is critical that different copies can have different initial coordinates (NEB for example), this is why the option "pack" is added to command rcrd(rcvd,rcvb,rcbd). To use this option, user need first concatenate different coordinates into a single file, and use "pack=n" to indicate how many sets of coordinates there are in the file, like the following example:

```
file rcrd name=(input.inpcrd) pack=4 read
```

Then addles will assign coordinates averagely. For example, if 4 sets coordinates exists in input file, and 20 copies are generated, then copy 1-5 will have coordinate set 1, copy 6-10 will have coordinates set 2, and so on. Note this option can't work with multiple copy regions now.

It is important to understand that each subsequent pick command acts on the ORIGINAL particle numbers. Making a copy of a given atom number also makes copies of all copies of that atom that were already created. This was the simplest way to be able to have a hierarchical LES setup, but you can't make extra copies of part of one of the copies already made. I'm not sure why you would want to, or if it is even correct to do so, but you should be warned. Copies can be anything -spanning residues, copies of fragments already copied, non-contiguous fragments, etc. Pay attention to the order in which you make the copies, and look carefully at the output to make sure you get what you had in mind. Addles will provide a list at the end of all atoms, the original parent atom, and how many copies were made.

There are array size limits in the file SIZE.h, I apologize in advance for the poor documentation on these. Mail *carlos.simmerling@stonybrook.edu* if you have any questions or problems.

14.3. More information on the ADDLES commands and options

`file:` open a file, also use one of

`rcrd:` read coords from this file

`rcvd:` read coords + velo from file

`rcvb:` read coords, velo and box from file

`wcrd:` write coords (and more if rcvd, rcvb) to file

`wprm:` write new topology file

`action:` start run, all of the following options must come AFTER action

`nomodv:` do NOT slightly randomize the velocities of the copies

`spac:` add a new subspace definition, using a pick command (see below); follow with "*numc=# pickcmd*", where # is the number of copies to make and *pickcmd* is a pick command that selects the group of atoms to copy.

`omas:` leave all masses at original values (otherwise scale 1/N)

`pimd:` write an prmtop file for PIMD simulation, which contains a much smaller non-bond exclusion list, atoms from other copy will not be included in this non-bond exclusion list.

Syntax for 'pick' commands

Currently, the syntax for picking atoms is somewhat limited. Simple Boolean logic is followed, but operations are carried out in order and parentheses are not allowed.

`#prt A B` picks the atom range from A to B by atom number

`#mon A B` picks the residue range from A to B by residue number

`#cca A B` picks the residue range from A to B by residue number, but dividing the residue between CA and C; the CO for A is included, and the CO for monomer B is not. See Simmerling and Elber, 1994 for an example of where this can be useful.

`chem prtc A` picks all atoms named A, case sensitive

`chem mono A` picks all residues named A, case sensitive

Completion wildcards are acceptable for names: H* picks H, HA, etc. Note that H*2 will select all atoms starting with H and ignore the 2.

Boolean logic:

`|` *or* atoms in either group are selected

`&` *and* atoms must be in both groups to be selected

`!=` *not* A != B will pick all atoms in A that are NOT in B

The user should carefully check the output file to ensure that the proper atoms were selected.

Examples:

pick command	***atoms selected***
pick #mon 4 19 done	**all atoms in residues 4 through 19**
pick #mon 1 50 & chem mono GLY done	**only GLY in residues 1 to 50**
pick chem mono LYS | chem mono GLU done	**any GLU or LYS residue**
pick #mon 1 5 != #prt 1 3 done	**residues 1 to 5 but not atoms 1 to 3**

so, a full command to add a new subspace (LES region) with 4 copies of atoms 15 to 35 is:

```
spac numc=4 pick #prt 15 35 done
```

14.4. Using the new topology/coordinate files with SANDER

These topology files are ready to use in Sander with one exception: all of the FF parameters have been scaled by 1/N for N copies. This is done to provide the energy of the new system as an average of the energies of the individual copies (note that it is an average energy or force, not the energy or force from an average copy coordinate). However, one additional correction is required for interactions between pairs of atoms in the same LES region. Sander will make these corrections for you, and this information is just to explain what is being done. For example, consider a system where you make 2 copies of a sidechain in a protein. Each charge is scaled by 1/2. For these atoms interacting with the rest of the system, each interaction is scaled by 1/2 and there are 2 such interactions. For a pair of particles inside the sub-space, however, the interaction is scaled by 1/2*1/2=1/4, and since the copies do not interact, there are only 2 such interactions and the sum does not correspond to the correct average. Therefore, the interaction must be scaled up by a factor of N. When the PME technique is requested, this simple scaling cannot be used since the entire charge set is used in the construction of the PME grid and individual charges are not used in the reciprocal space calculation. Therefore, the intra-copy energies and forces are corrected in a separate step for PME calculations. Sander will print out the number of correction interactions that need to

be calculated, and very large amounts of these will make the calculation run more slowly. PME also needs to do a separate correction calculation for excluded atom pairs (atoms that should not have a nonbonded interaction, such as those that are connected by a bond). Large LES regions result in large numbers of excluded atoms, and these will result in a larger computational penalty for LES compared to non-LES simulations. For both of these reasons, it is more efficient computationally to use smaller LES regions- but see the discussion above for how region size affects simulation efficiency. These changes are included in the LES version of Sander (sander.LES). Each particle is assigned a LES 'type' (each new set of copies is a new type), and for each pair of types there is a scaling factor for the nonbond interactions between LES particles of those types. Most of the scaling factors are 1.0, but some are not - such as the diagonal terms which correspond to interactions inside a given subspace, and also off-diagonal terms where only some of the copies are in common. An example of this type is the side chain example given above- each of the 3 backbone copies has 2 sidechains, and while interactions inside the side chains need a factor of 6, interactions between the side chain and backbone need a factor of 3. This matrix of scaling factors is stored in the new topology file, along with the type for each atom, and the number of types. The changes made in sander relate to reading and using these scale factors.

14.5. Using LES with the Generalized Born solvation model

LES simulations can be performed using the GB solvent model, with some limitations. Compared to LES simulations in explicit water, using GB with LES provides several advantages. The most important is how each of the copies interacts with the solvent. With explicit water, the water is normally not copied and therefore interacts in an average way with all LES copies. This has important consequences for solvation of the copies. If the copies move apart, water cannot overlap any of them and therefore the water cavity will be that defined by the union of the space occupied by the copies. This has two consequences. First, moving the copies apart requires creation of a larger solvent cavity and therefore copies have a greater tendency to remain together, reducing the effectiveness of LES. Second, when the copies do move apart, each copy will not be individually solvated.

These effects arise because the water interacts with all of the copies; for each copy to be solvated independently of the other copies would require copying the water molecules. This is normally not a good idea, since copying all of the water would result in very significant computational expense. Copying only water near the solute would be tractable, but one would need to ensure that the copied waters did not exchange with non-LES bulk waters.

Using GB with LES largely overcomes these problems since each copy can be individually solvated with the continuum model. Thus when one copy moves, the solvation of the other copies are not affected. This results in a more reasonable solvation of each copy and also improves the independence of the copies. Of course the resulting simulations do retain all of the limitations that accompany the GB models.

The current code allows igb values of 1, 5 or 7 when using LES. Surface area calculations are not yet supported with LES. Only a single LES region is permitted for GB+LES simulations. A new namelist variable was introduced (RDT) in sander to control the compromise of speed and accuracy for GB+LES simulations. The article referenced below provides more detail on the function of this variable. RDT is the effective radii deviation threshold. When using GB+LES, non-LES atoms require multiple effective Born radii for an exact calculation. Using these multiple radii can significantly increase calculation time required for GB calculations. When the difference between the multiple radii for a non-LES atom is less than RDT, only a single effective radius will be used. A value of 0.01 has been found to provide a reasonable compromise between speed and accuracy, and is the default value. Before using this method, it is strongly recommended that the user read the article describing the derivation of the GB+LES approach.[305]

14.6. Case studies: Examples of application of LES

14.6.1. Enhanced sampling for individual functional groups: Glucose

The first example will deal with enhancing sampling for small parts of a molecule, such as individual functional groups or protein side chains. In this case we wanted to carry out separate simulations of α and β (not converting between anomers, only for conversions involving rotations about bonds) glucose, but the 5 hydroxyl groups and the strong hydrogen bonds between neighboring hydroxyls make conversion between different

rotamers slow relative to affordable simulation times. The eventual goal was to carry out free energy simulations converting between anomers, but we need to ensure that each window during the Gibbs calculation would be able to sample all relevant orientations of hydroxyl groups in their proper Boltzmann-weighted populations. We were initially unsure how many different types of structures should be populated and carried out non-LES simulations starting from different conformations. We found that transitions between different conformations were separated by several hundred picoseconds, far too long to expect converged populations during each window of the free energy calculation. We therefore decided to enhance conformational sampling for each hydroxyl group by making 5 copies of each hydroxyl hydrogen and also 5 copies of the entire hydroxymethyl group. Since the hydroxyl rotamer for each copy should be relatively independent, we decided to place each group in a different LES region. This meant that each hydroxyl copy interacted with all copies of the neighboring groups, with a total of 5*5*5*5*5 or 3125 structural combinations contributing to the LES average energy at each point in time. The input file is given below.

```
file rprm name=(parm.solv.top) read
file rcvb name=(glucose.solv.equ.crd) read
file wprm name=(les.prmtop) wovr
file wcrd name=(glucose.les.crd) wovr
action
omas
~ 5 copies of each hydroxyl hydrogen- copying oxygen will make no difference
~ since they will not be able to move significantly apart anyway
spac numc=5 pick chem prtc HO1 done
spac numc=5 pick chem prtc HO2 done
spac numc=5 pick chem prtc HO3 done
spac numc=5 pick chem prtc HO4 done
~ take the entire hydroxy methyl group
spac numc=5 pick #prt 20 24 done
*EOD
```

This worked quite well, with transitions now occurring every few ps and populations that were essentially independent of initial conformation.[302]

14.6.2. Enhanced sampling for a small region: Application of LES to a nucleic acid loop

In this example, we consider a biomolecule (in this case a single RNA strand) for which part of the structure is reliable and another part is potentially less accurate. This can be the case in a number of different modeling situations, such as with homologous proteins or when the experimental data is incomplete. In this case two different structures were available for the same RNA sequence. While both structures were hairpins with a tetraloop, the loop conformations differed, and one was more accurate. We tested whether MD would be able to show that one structure was not stable and would convert to the other on an affordable timescale.

Standard MD simulations of several ns were not able to undergo any conversion between these two structures (the initial structure was always retained). Since the stem portion of the RNA was considered to be accurate, LES was only applied to the tetraloop region. In this case, both of the ends of the LES region would be attached to the same locations in space, and there was no concern about copies diffusing too far apart to re-converge to the same positions after optimization. The issues that need to be addressed once again are the number of copies to use, and how to place the LES region(s). I usually start with the simplest choices and used 5 LES copies and only a single LES region consisting of the entire loop. If each half of the loop was copied, then it might become too *crowded* with copies near the base-pair hydrogen bonds and conformational changes that required moving a base through this regions could become even more difficult (see the background section for details). Therefore, one region was chosen, and the RNA stem, counterions and solvent were not copied. The ADDLES input file is given below.

```
file rprm name=(prm.top) read
file rcvb name=(rna.crd) read
file wprm name=(les.parm) wovr
```

```
file wcrd name=(les.crd) wovr
action
omas
~ copy the UUCG loop region- residues 5 to 8.
~ pick by atom number, though #mon 5 8 would work the same way
spac numc=5 pick #prt 131 255 done
*EOD
```

Subsequent LES simulations were able to reproducibly convert from what was known to be the incorrect structure to the correct one, and stay in the correct structure in simulations that started there. Different numbers of LES copies as well as slightly changing the size of the LES region (from 4 residues to 6, extending 1 residue beyond the loop on either side) were not found to affect the results. Fewer copies still converted between structures, but on a slower timescale, consistent with the barrier heights being reduced roughly proportional to the number of copies used. See Simmerling, Miller and Kollman, 1998, for further details.

14.6.3. Improving conformational sampling in a small peptide

In this example, we were interested not just in improving sampling of small functional groups or even individual atoms, but in the entire structure of a peptide. The peptide sequence is AVPA, with ACE and NME terminal groups. Copying just the side chains might be helpful, but would not dramatically reduce the barriers to backbone conformational changes, especially in this case with so little conformational variety inherent in the Ala and Pro residues. We therefore apply LES to all atoms. If we copied the entire peptide in 1 LES regions, the copies could float apart. While this would not be a disaster, it would make it difficult to bring all of the copies back together if we were searching for the global energy minimum, as described above. We therefore use more than one LES region, and need to decide where to place the boundaries between regions. A useful rule of thumb is that regions should be at least two amino acids in size, so we pick our two regions as Ace-Ala-Val and Pro-Ala-Nme. If we make five LES copies of each region and each copy does not interact with other copies of the same regions, each half the peptide will be represented by five potentially different conformations at each point in time. In addition, since each copy interacts with all copies of the rest of the system, there are 25 different combinations of the two halves of the peptide that contribute at each point in time. This statistical improvement alone is valuable, but the corresponding barriers are also reduced by approximately the same factors. When we place the peptide in a solvent box the solvent interacts in an average way with each of the copies. The input file is given below, and all of the related files can be found in the test directory for LES.

```
~ all file names are specified at the beginning, before "action"
~ specify input prmtop
file rprm name=(prmtop) read
~ specify input coordinates, velocities and box (this is a restart file)
file rcvb name=(md.solv.crd) read
~ specify LES prmtop
file wprm name=(LES.prmtop) wovr
~ specify LES coordinates (and velocities and box since they were input)
file wcrd name=(LES.crd) wovr
~ now the action command reads the files and tells addles to
~ process commands
action
~ do not scale masses of copied particles
omas
~ divide the peptide into 2 regions.
~ use the CCA option to place the division between carbonyl and
~ alpha carbon
~ use the "or" to make sure all atoms in the terminal residues
~ are included since the CCA option places the region division at C/CA
~ and we want all of the terminal residue included on each end
```

```
~
~ make 5 copies of each half
~ "spac" defines a LES subspace (or region)
spac numc=5 pick #cca 1 3 | #mon 1 1 done
spac numc=5 pick #cca 4 6 | #mon 6 6 done
~ the following line is required at the end
*EOD
```

This example brings up several important questions:

1. Should I make LES copies before or after adding solvent? Since LEaP is used to add solvent, and LEaP will not be able to load and understand a LES structure, you must run ADDLES after you have solvated the peptide in LEaP. ADDLES should be the last step before running SANDER.

2. Which structure should be used as input to ADDLES? If you will also be carrying out non-LES simulations, then you can equilibrate the non-LES simulation and carry out any amount of production simulation desired before taking the structure and running ADDLES. At the point you may switch to only LES simulations, or continue both LES and non-LES from the same point (using different versions of SANDER). Typically I equilibrate my system without LES to ensure that it has initial stability and that everything looks OK, then switch to LES afterward. This way I separate any potential problems from incorrect LES setup from those arising from problems with the non-LES setup, such as in initial coordinates, LEaP setup, solvent box dimensions and equilibration protocols.

3. How can I analyze the resulting LES simulation? This is probably the most difficult part of using LES. With all of the extra atoms, most programs will have difficulty. For example, a given amino acid with LES will have multiple phi and psi backbone dihedral angles. There are basically two options: first, you can process your trajectory such that you obtain a single structure (non-LES). This might be just extracting one of the copies, or it might be one by taking the average of the LES copies. After that, you can proceed to traditional analysis but must keep in mind that the average structure may be non-physical and may not represent any actual structure being sampled by the copies, especially if they move apart significantly. A better way is to use LES-friendly analysis tools, such as those developed in the group of Carlos Simmerling. The visualization program MOIL-View (*http://morita.chem.sunysb.edu/carlos/moil-view.html*) is one example of these programs, and has many analysis tools that are fully LES compatible. Read the program web page or manual for more details.

1.7. Unresolved issues with LES in Amber

1. Sander can't currently maintain groups of particles at different temperatures (important for dynamics, less so for optimization.)[301, 306] Users can set *temp0les* to maintain all LES atoms at a temperature that is different from that for the system as a whole, but all LES atoms are then coupled to the same bath.

2. Initial velocity issues as mentioned above- works properly, user must be careful.

3. Analysis programs may not be compatible. See *http://morita.chem.sunysb.edu/carlos/moil-view.html* for an LES-friendly analysis and visualization program.

4. Visualization can be difficult, especially with programs that use distance-based algorithms to determine bonds. See #3 above.

5. Water should not be copied- the fast water routines have not been modified. For most users this won't matter.

6. Copies should not span different 'molecules' for pressure coupling and periodic imaging issues. Copies of an entire 'molecule' should result in the copies being placed in new, separate molecules- currently this is not done. This would include copying things such as counterions and entire protein or nucleic acid chains.

7. Copies are placed into the same residue as the original atoms- this can make some residues much larger than others, and may result in less efficient parallelization with algorithms that assign nonbond workload based on residue numbers.

Bibliography

[1] L. David; R. Luo; M. K. Gilson. Comparison of generalized born and poisson models: Energetics and dynamics of hiv protease. *J. Comput. Chem.*, **2000**, *21*, 295–309.

[2] M. Feig. Kinetics from Implicit Solvent Simulations of Biomolecules as a Function of Viscosity. *J. Chem. Theory Comput.*, **2007**, *3*, 1734–1748.

[3] R. E. Amaro; X. Cheng; I. Ivanov; D. Xu; A. J. Mccammon. Characterizing Loop Dynamics and Ligand Recognition in Human- and Avian-Type Influenza Neuraminidases via Generalized Born Molecular Dynamics and End-Point Free Energy Calculations. *J. Am. Chem. Soc.*, **2009**, *131*, 4702–4709.

[4] B. Zagrovic; V. Pande. Solvent viscosity dependence of the folding rate of a small protein: Distributed computing study. *J. Comput. Chem.*, **2003**, *24*, 1432–1436.

[5] R. Anandakrishnan; A. Drozdetski; R. C. Walker; A. V. Onufriev. Speed of Conformational Change: Comparing Explicit and Implicit Solvent Molecular Dynamics Simulations. *Biophysical Journal*, **2015**, *108*, 1153–1164.

[6] A. V. Onufriev; D. A. Case. Generalized Born implicit solvent models for biomolecules. *Annu. Rev. Biophys.*, **2019**, *48*, 275–296.

[7] J. Weiser; P. S. Shenkin; W. C. Still. Approximate Atomic Surfaces from Linear Combinations of Pairwise Overlaps (LCPO). *J. Comput. Chem.*, **1999**, *20*, 217–230.

[8] W. C. Still; A. Tempczyk; R. C. Hawley; T. Hendrickson. Semianalytical treatment of solvation for molecular mechanics and dynamics. *J. Am. Chem. Soc.*, **1990**, *112*, 6127–6129.

[9] M. Schaefer; M. Karplus. A comprehensive analytical treatment of continuum electrostatics. *J. Phys. Chem.*, **1996**, *100*, 1578–1599.

[10] S. R. Edinger; C. Cortis; P. S. Shenkin; R. A. Friesner. Solvation free energies of peptides: Comparison of approximate continuum solvation models with accurate solution of the Poisson-Boltzmann equation. *J. Phys. Chem. B*, **1997**, *101*, 1190–1197.

[11] B. Jayaram; D. Sprous; D. L. Beveridge. Solvation free energy of biomacromolecules: Parameters for a modified generalized Born model consistent with the AMBER force field. *J. Phys. Chem. B*, **1998**, *102*, 9571–9576.

[12] C. J. Cramer; D. G. Truhlar. Implicit solvation models: Equilibria, structure, spectra, and dynamics. *Chem. Rev.*, **1999**, *99*, 2161–2200.

[13] D. Bashford; D. A. Case. Generalized Born models of macromolecular solvation effects. *Annu. Rev. Phys. Chem.*, **2000**, *51*, 129–152.

[14] A. Onufriev; D. Bashford; D. A. Case. Modification of the generalized Born model suitable for macromolecules. *J. Phys. Chem. B*, **2000**, *104*, 3712–3720.

[15] M. S. Lee; F. R. Salsbury, Jr.; C. L. Brooks, III. Novel generalized Born methods. *J. Chem. Phys.*, **2002**, *116*, 10606–10614.

[16] B. N. Dominy; C. L. Brooks, III. Development of a generalized Born model parameterization for proteins and nucleic acids. *J. Phys. Chem. B*, **1999**, *103*, 3765–3773.

[17] V. Tsui; D. A. Case. Molecular dynamics simulations of nucleic acids using a generalized Born solvation model. *J. Am. Chem. Soc.*, **2000**, *122*, 2489–2498.

[18] N. Calimet; M. Schaefer; T. Simonson. Protein molecular dynamics with the generalized Born/ACE solvent model. *Proteins*, **2001**, *45*, 144–158.

[19] A. Onufriev; D. Bashford; D. A. Case. Exploring protein native states and large-scale conformational changes with a modified generalized Born model. *Proteins*, **2004**, *55*, 383–394.

[20] J. Srinivasan; M. W. Trevathan; P. Beroza; D. A. Case. Application of a pairwise generalized Born model to proteins and nucleic acids: inclusion of salt effects. *Theor. Chem. Acc.*, **1999**, *101*, 426–434.

[21] A. Onufriev; D. A. Case; D. Bashford. Effective Born radii in the generalized Born approximation: The importance of being perfect. *J. Comput. Chem.*, **2002**, *23*, 1297–1304.

[22] G. D. Hawkins; C. J. Cramer; D. G. Truhlar. Parametrized models of aqueous free energies of solvation based on pairwise descreening of solute atomic charges from a dielectric medium. *J. Phys. Chem.*, **1996**, *100*, 19824–19839.

[23] F. M. Richards. Areas, volumes, packing, and protein structure. *Ann. Rev. Biophys. Bioeng.*, **1977**, *6*, 151–176.

[24] M. Schaefer; C. Froemmel. A precise analytical method for calculating the electrostatic energy of macromolecules in aqueous solution. *J. Mol. Biol.*, **1990**, *216*, 1045–1066.

[25] M. Feig; A. Onufriev; M. Lee; W. Im; D. A. Case; C. L. Brooks, III. Performance comparison of the generalized Born and Poisson methods in the calculation of the electrostatic solvation energies for protein structures. *J. Comput. Chem.*, **2004**, *25*, 265–284.

[26] R. Geney; M. Layten; R. Gomperts; C. Simmerling. Investigation of salt bridge stability in a generalized Born solvent model. *J. Chem. Theory Comput.*, **2006**, *2*, 115–127.

[27] A. Okur; L. Wickstrom; C. Simmerling. Evaluation of salt bridge structure and energetics in peptides using explicit, implicit and hybrid solvation models. *J. Chem. Theory Comput.*, **2008**, *4*, 488–498.

[28] A. Okur; L. Wickstrom; M. Layten; R. Geney; K. Song; V. Hornak; C. Simmerling. Improved efficiency of replica exchange simulations through use of a hybrid explicit/implicit solvation model. *J. Chem. Theory Comput.*, **2006**, *2*, 420–433.

[29] D. R. Roe; A. Okur; L. Wickstrom; V. Hornak; C. Simmerling. Secondary Structure Bias in Generalized Born Solvent Models: Comparison of Conformational Ensembles and Free Energy of Solvent Polarization from Explicit and Implicit Solvation. *J. Phys. Chem. B*, **2007**, *111*, 1846–1857.

[30] H. Nguyen; D. R. Roe; C. Simmerling. Improved Generalized Born Solvent Model Parameters for Protein Simulations. *J. Chem. Theory Comput.*, **2013**, *9*, 2020–2034.

[31] H. Nguyen; J. Maier; H. Huang; V. Perrone; C. Simmerling. Folding simulations for proteins with diverse topologies are accessible in days with a physics-based force field and implicit solvent. *J. Am. Chem. Soc.*, **2014**, *136*, 13959–13962. PMID: 25255057.

[32] H. Nguyen; A. Pérez; S. Bermeo; C. Simmerling. Refinement of Generalized Born Implicit Solvation Parameters for Nucleic Acids and Their Complexes with Proteins. *J. Chem. Theory Comput.*, **2015**, *11*, 3714–3728.

[33] A. Onufriev. in *Modeling Solvent Environments*, M. Feig, Ed., (Wiley, USA). pp 127–165. 2010.

[34] G. D. Hawkins; C. J. Cramer; D. G. Truhlar. Pairwise solute descreening of solute charges from a dielectric medium. *Chem. Phys. Lett.*, **1995**, *246*, 122–129.

[35] M. Schaefer; H. W. T. Van Vlijmen; M. Karplus. Electrostatic contributions to molecular free energies in solution. *Adv. Protein Chem.*, **1998**, *51*, 1–57.

[36] V. Tsui; D. A. Case. Theory and applications of the generalized Born solvation model in macromolecular simulations. *Biopolymers (Nucl. Acid. Sci.)*, **2001**, *56*, 275–291.

[37] C. P. Sosa; T. Hewitt; M. S. Lee; D. A. Case. Vectorization of the generalized Born model for molecular dynamics on shared-memory computers. *J. Mol. Struct. (Theochem)*, **2001**, *549*, 193–201.

[38] J. Mongan; C. Simmerling; J. A. McCammon; D. A. Case; A. Onufriev. Generalized Born with a simple, robust molecular volume correction. *J. Chem. Theory Comput.*, **2007**, *3*, 156–169.

[39] H. Huang; C. Simmerling. Fast Pairwise Approximation of Solvent Accessible Surface Area for Implicit Solvent Simulations of Proteins on CPUs and GPUs. *J. Chem. Theory Comput.*, **2018**, *14*, 5797–5814.

[40] D. Sitkoff; K. A. Sharp; B. Honig. Accurate calculation of hydration free energies using macroscopic solvent models. *J. Phys. Chem.*, **1994**, *98*, 1978–1988.

[41] T. Luchko; S. Gusarov; D. R. Roe; C. Simmerling; D. A. Case; J. Tuszynski; A. Kovalenko. Three-dimensional molecular theory of solvation coupled with molecular dynamics in Amber. *J. Chem. Theory Comput.*, **2010**, *6*, 607–624.

[42] D. Chandler; H. C. Andersen. Optimized cluster expansions for classical fluids. ii. theory of molecular liquids. *J. Chem. Phys.*, **1972**, *57*, 1930–1937.

[43] F. Hirata; P. J. Rossky. An extended RISM equation for molecular polar fluids. *Chem. Phys. Lett.*, **1981**, pp 329–334.

[44] F. Hirata; B. M. Pettitt; P. J. Rossky. Application of an extended rism equation to dipolar and quadrupolar fluids. *J. Chem. Phys.*, **1982**, *77*, 509–520.

[45] F. Hirata; P. J. Rossky; B. M. Pettitt. The interionic potential of mean force in a molecular polar solvent from an extended rism equation. *J. Chem. Phys.*, **1983**, *78*, 4133–4144.

[46] D. Chandler; J. McCoy; S. Singer. Density functional theory of nonuniform polyatomic systems. i. general formulation. *J. Chem. Phys.*, **1986**, *85*, 5971–5976.

[47] D. Chandler; J. McCoy; S. Singer. Density functional theory of nonuniform polyatomic systems. ii. rational closures for integral equations. *J. Chem. Phys.*, **1986**, *85*, 5977–5982.

[48] D. Beglov; B. Roux. Numerical solution of the hypernetted chain equation for a solute of arbitrary geometry in three dimensions. *J. Chem. Phys.*, **1995**, *103*, 360–364.

[49] D. Beglov; B. Roux. An integral equation to describe the solvation of polar molecules in liquid water. *J. Phys. Chem. B*, **1997**, *101*, 7821–7826.

[50] A. Kovalenko; F. Hirata. Three-dimensional density profiles of water in contact with a solute of arbitrary shape: a RISM approach. *Chem. Phys. Lett.*, **1998**, *290*, 237–244.

[51] A. Kovalenko; F. Hirata. Self-consistent description of a metal–water interface by the Kohn–Sham density functional theory and the three-dimensional reference interaction site model. *J. Chem. Phys.*, **1999**, *110*, 10095–10112.

[52] A. Kovalenko. In Hirata [670], chapter 4.

[53] A. Kovalenko; F. Hirata. Potentials of mean force of simple ions in ambient aqueous solution. i: Three-dimensional reference interaction site model approach. *J. Chem. Phys.*, **2000**, *112*, 10391–10402.

[54] A. Kovalenko; F. Hirata. Potentials of mean force of simple ions in ambient aqueous solution. ii: Solvation structure from the three-dimensional reference interaction site model approach, and comparison with simulations. *J. Chem. Phys.*, **2000**, *112*, 10403–10417.

[55] J.-P. Hansen; I. R. McDonald. *Theory of simple liquids*. Academic Press, London, 1990.

[56] F. Hirata. In *Molecular Theory of Solvation* [670], chapter 1.

[57] J. S. Perkyns; B. M. Pettitt. A site-site theory for finite concentration saline solutions. *J. Chem. Phys.*, **1992**, *97*, 7656–7666.

[58] A. Kovalenko; S. Ten-No; F. Hirata. Solution of three-dimensional reference interaction site model and hypernetted chain equations for simple point charge water by modified method of direct inversion in iterative subspace. *J. Comput. Chem*, **1999**, *20*, 928–936.

[59] I. S. Joung; T. Luchko; D. A. Case. Simple electrolyte solutions: Comparison of DRISM and molecular dynamics results for alkali halide solutions. *J Chem Phys*, **2013**, *138*, 044103.

[60] G. M. Giambasu; T. Luchko; D. Herschlag; D. M. York; D. A. Case. Ion counting from explicit-solvent simulations and 3d-RISM. *Biophys J*, **2014**, *106*, 883–894.

[61] J. W. Kaminski; S. Gusarov; T. A. Wesolowski; A. Kovalenko. Modeling solvatochromic shifts using the orbital-free embedding potential at statistically mechanically averaged solvent density. *J. Phys. Chem. A*, **2010**, *114*, 6082–6096.

[62] M. Frigo; S. G. Johnson. FFTW: An adaptive software architecture for the FFT. in *Proc. 1998 IEEE Intl. Conf. Acoustics Speech and Signal Processing*, volume 3, pp 1381–1384. IEEE, 1998.

[63] M. Frigo. A fast Fourier transform compiler. in *Proc. 1999 ACM SIGPLAN Conf. on Programming Language Design and Implementation*, volume 34, pp 169–180. ACM, 1999.

[64] S. J. Singer; D. Chandler. Free energy functions in the extended RISM approximation. *Mol. Phys.*, **1985**, *55*, 621–625.

[65] B. M. Pettitt; P. J. Rossky. Alkali halides in water: Ion-solvent correlations and ion-ion potentials of mean force at infinite dilution. *J. Chem. Phys.*, **1986**, *15*, 5836–5844.

[66] S. M. Kast. Free energies from integral equation theories: Enforcing path independence. *Phys. Rev. E*, **2003**, *67*, 041203.

[67] G. Schmeer; A. Maurer. Development of thermodynamic properties of electrolyte solutions with the help of RISM-calculations at the Born-Oppenheimer level. *Phys. Chem. Chem. Phys.*, **2010**, *12*, 2407–2417.

[68] S. Gusarov; T. Ziegler; A. Kovalenko. Self-consistent combination of the three-dimensional RISM theory of molecular solvation with analytical gradients and the amsterdam density functional package. *J. Phys. Chem. A*, **2006**, *110*, 6083–6090.

[69] T. Miyata; F. Hirata. Combination of molecular dynamics method and 3D-RISM theory for conformational sampling of large flexible molecules in solution. *J. Comput. Chem.*, **2007**, *29*, 871–882.

[70] S. M. Kast; T. Kloss. Closed-form expressions of the chemical potential for integral equation closures with certain bridge functions. *J. Chem. Phys.*, **2008**, *129*, 236101.

[71] D. Chandler; Y. Singh; D. M. Richardson. Excess electrons in simple fluids. I. General equilibrium theory for classical hard sphere solvents. *J. Chem. Phys.*, **1984**, *81*, 1975–1982.

[72] T. Ichiye; D. Chandler. Hypernetted chain closure reference interaction site method theory of structure and thermodynamics for alkanes in water. *J. Phys. Chem.*, **1988**, *92*, 5257–5261.

[73] P. H. Lee; G. M. Maggiora. Solvation thermodynamics of polar molecules in aqueous solution by the XRISM method. *J. Phys. Chem.*, **1993**, *97*, 10175–10185.

[74] S. Genheden; T. Luchko; S. Gusarov; A. Kovalenko; U. Ryde. An MM/3D-RISM approach for ligand-binding affinities. *J. Phys. Chem.*, **2010**. Accepted.

[75] D. S. Palmer; A. I. Frolov; E. L. Ratkova; M. V. Fedorov. Towards a universal method for calculating hydration free energies: a 3D reference interaction site model with partial molar volume correction. *J. Phys.: Condens. Matter*, **2010**, *22*, 492101.

[76] J.-F. Truchon; B. M. Pettitt; P. Labute. A cavity corrected 3D-RISM functional for accurate solvation free energies. *J. Chem. Theory Comput.*, **2014**, *10*, 934–941.

[77] V. Sergiievskyi; G. Jeanmairet; M. Levesque; D. Borgis. Solvation free-energy pressure corrections in the three dimensional reference interaction site model. *J. Chem. Phys.*, **2015**, *143*, 184116.

[78] J. Johnson; D. A. Case; T. Yamazaki; S. Gusarov; A. Kovalenko; T. Luchko. Small molecule solvation energy and entropy from 3D-RISM. *J. Phys. Condens. Mat.*, **2016**.

[79] T. Luchko; N. Blinov; G. C. Linon; K. P. Joyce; A. Kovalenko. SAMPL5: 3D-RISM partition coefficient calculations with partial molar volume corrections and solute conformational sampling. *J. Comput. Aided Mol. Design*, **2016**, *30*, 1115–1127.

[80] R. C. Walker; M. F. Crowley; D. A. Case. The implementation of a fast and accurate QM/MM potential method in Amber. *J. Comput. Chem.*, **2008**, *29*, 1019–1031.

[81] G. M. Seabra; R. C. Walker; M. Elstner; D. A. Case; A. E. Roitberg. Implementation of the SCC-DFTB Method for Hybrid QM/MM Simulations within the Amber Molecular Dynamics Package. *J. Phys. Chem. A.*, **2007**, *20*, 5655–5664.

[82] M. Elstner; D. Porezag; G. Jungnickel; J. Elsner; M. Haugk; T. Frauenheim; S. Suhai; G. Seifert. Self-consistent charge density functional tight-binding method for simulation of complex material properties. *Phys. Rev. B*, **1998**, *58*, 7260.

[83] T. Kruger; M. Elstner; P. Schiffels; T. Frauenheim. Validation of the density-functional based tight-binding approximation. *J. Chem. Phys.*, **2005**, *122*, 114110.

[84] T. J. Giese; D. M. York. Charge-dependent model for many-body polarization, exchange, and dispersion interactions in hybrid quantum mechanical/molecular mechanical calculations. *J. Chem. Phys.*, **2007**, *127*, 194101–194111.

[85] J. J. P. Stewart. Optimization of parameters for semiempirical methods I. Method. *J. Comput. Chem.*, **1989**, *10*, 209–220.

[86] M. J. S. Dewar; E. G. Zoebisch; E. F. Healy; J. J. P. Stewart. AM1: A new general purpose quantum mechanical molecular model. *J. Am. Chem. Soc.*, **1985**, *107*, 3902–3909.

[87] G. B. Rocha; R. O. Freire; A. M. Simas; J. J. P. Stewart. RM1: A Reparameterization of AM1 for H, C, N, O, P, S, F, Cl, Br and I. *J. Comp. Chem.*, **2006**, *27*, 1101–1111.

[88] M. J. S. Dewar; W. Thiel. Ground states of molecules. 38. The MNDO method, approximations and parameters. *J. Am. Chem. Soc.*, **1977**, *99*, 4899–4907.

[89] M. P. Repasky; J. Chandrasekhar; W. L. Jorgensen. PDDG/PM3 and PDDG/MNDO: Improved semiempirical methods. *J. Comput. Chem.*, **2002**, *23*, 1601–1622.

[90] J. P. McNamara; A. M. Muslim; H. Abdel-Aal; H. Wang; M. Mohr; I. H. Hillier; R. A. Bryce. Towards a quantum mechanical force field for carbohydrates: A reparameterized semiempirical MO approach. *Chem. Phys. Lett.*, **2004**, *394*, 429–436.

[91] M. I. Bernal-Uruchurtu; M. F. Ruiz-López. Basic ideas for the correction of semiempirical methods describing H-bonded systems. *Chem. Phys. Lett.*, **2000**, *330*, 118–124.

[92] O. I. Arillo-Flores; M. F. Ruiz-López; M. I. Bernal-Uruchurtu. Can semi-empirical models describe HCl dissociation in water? *Theoret. Chem. Acc.*, **2007**, *118*, 425–435.

[93] W. Thiel; A. A. Voityuk. Extension of the MNDO formalism to d orbitals: Integral approximations and preliminary numerical results. *Theoret. Chim. Acta*, **1992**, *81*, 391–404.

[94] W. Thiel; A. A. Voityuk. Extension of the MNDO formalism to d orbitals: Integral approximations and preliminary numerical results. *Theoret. Chim. Acta*, **1996**, *93*, 315.

[95] W. Thiel; A. A. Voityuk. Erratum: Extension of MNDO to d orbitals: Parameters and results for the second-row elements and for the zinc group. *J. Phys. Chem.*, **1996**, *100*, 616–626.

[96] P. Imhof; F. Noé; S. Fischer; J. C. Smith. AM1/d Parameters for Magnesium in Metalloenzymes. *J. Chem. Theory Comput.*, **2006**, *2*, 1050–1056.

[97] K. Nam; Q. Cui; J. Gao; D. M. York. Specific Reaction Parametrization of the AM1/d Hamiltonian for Phosphoryl Transfer Reactions: H, O, and P Atoms. *J. Chem. Theory Comput.*, **2007**, *3*, 486–504.

[98] J. J. P. Stewart. Optimization of parameters for semiempirical methods V: Modification of NDDO approximations and application to 70 elements. *J. Mol. Mod.*, **2007**, *13*, 1173–1213.

[99] D. Porezag; T. Frauenheim; T. Kohler; G. Seifert; R. Kaschner. Construction of tight-binding-like potentials on the basis of density-functional-theory: Applications to carbon. *Phys. Rev. B*, **1995**, *51*, 12947.

[100] G. Seifert; D. Porezag; T. Frauenheim. Calculations of molecules, clusters and solids with a simplified LCAO-DFT-LDA scheme. *Int. J. Quantum Chem.*, **1996**, *58*, 185.

[101] M. Gaus; Q. Cui; M. Elstner. DFTB3: Extension of the self-consistent-charge density-functional tight-binding method (SCC-DFTB). *J. Chem. Theory Comput.*, **2011**, *7*, 931–948.

[102] M. Elstner; P. Hobza; T. Frauenheim; S. Suhai; E. Kaxiras. Hydrogen bonding and stacking interactions of nucleic acid base pairs: a density-functional-theory based treatment. *J. Chem. Phys.*, **2001**, *114*, 5149.

[103] J. A. Kalinowski; B. Lesyng; J. D. Thompson; C. J. Cramer; D. G. Truhlar. Class IV charge model for the self-consistent charge density-functional tight-binding method. *J. Phys. Chem. A*, **2004**, *108*, 2545–2549.

[104] Y. Yang; H. Yu; D. M. York; Q. Cui; M. Elstner. Extension of the self-consistent charge density-functional tight-binding method: Third-order expansion of the density functional theory total energy and introduction of a modified effective Coulomb interaction. *J. Phys. Chem. A*, **2007**, *111*, 10861–10873.

[105] M. Korth. Third-generation hydrogen-bonding corrections for semiempirical qm methods and force fields. *J. Chem. Theory Comput.*, **2010**, *6*, 3808.

[106] P. Jurecka; J. Cerný; P. Hobza; D. R. Salahub. Density functional theory augmented with an empirical dispersion term. Interaction energies and geometries of 80 noncovalent complexes compared with ab initio quantum mechanics calculations. *J. Comp. Chem.*, **2007**, *28*, 555–569.

[107] A. Bondi. van der Waals volumes and radii. *J. Phys. Chem.*, **1964**, *68*, 441–451.

[108] M. Korth; M. Pitonak; J. Rezac; P. Hobza. A transferable h-bonding correction for semiempirical quantum-chemical methods. *J. Chem. Theory Comput.*, **2010**, *6*, 344–352.

[109] M. Manathunga; C. Jin; V. W. D. Cruzeiro; J. Smith; K. Keipert; D. Pekurovsky; D. Mu; Y. Miao; X. He; K. Ayers; E. Brothers; A. W. Goetz; K. M. Merz. QUICK, version 21.03. University of California San Diego, CA and Michigan State University, East Lansing, MI. 2021.

[110] Y. Miao; K. M. Merz. Acceleration of Electron Repulsion Integral Evaluation on Graphics Processing Units via Use of Recurrence Relations. *J. Chem. Theory Comput.*, **2013**, *9*, 965–976.

[111] Y. Miao; K. M. Merz. Acceleration of High Angular Momentum Electron Repulsion Integrals and Integral Derivatives on Graphics Processing Units. *J. Chem. Theory Comput.*, **2015**, *11*, 1449–1462.

[112] M. Manathunga; Y. Miao; D. Mu; A. W. Götz; K. M. Merz. Parallel Implementation of Density Functional Theory Methods in the Quantum Interaction Computational Kernel Program. *J. Chem. Theory Comput.*, **2020**, *16*, 4315–4326.

[113] M. Manathunga; C. Jin; V. W. D. Cruzeiro; Y. Miao; D. Mu; K. Arumugam; K. Keipert; H. M. Aktulga; J. Merz, Kenneth M.; A. W. Götz. Harnessing the Power of Multi-GPU Acceleration into the Quantum Interaction Computational Kernel Program. *ChemRxiv*, **2021**. https://doi.org/10.26434/chemrxiv.13769209.v1.

[114] E. Pellegrini; M. J. Field. A generalized-Born solvation model for macromolecular hybrid-potential calculations. *J. Phys. Chem. A.*, **2002**, *106*, 1316–1326.

[115] K. Nam; J. Gao; D. York. An efficient linear-scaling Ewald method for long-range electrostatic interactions in combined QM/MM calculations. *J. Chem. Theory Comput.*, **2005**, *1*, 2–13.

[116] A. W. Götz; M. A. Clark; R. C. Walker. An extensible interface for QM/MM molecular dynamics simulations with AMBER. *J. Comput. Chem.*, **2014**, *35*, 95–108.

[117] V. W. D. Cruzeiro; M. Manathunga; J. Merz, Kenneth M.; A. W. Götz. Open-Source Multi-GPU-Accelerated QM/MM Simulations with AMBER and QUICK. *ChemRxiv*, **2021**. https://doi.org/10.26434/chemrxiv.13984028.v1.

[118] Q. T. Wang; R. A. Bryce. Improved hydrogen bonding at the NDDO-type semiempirical quantum mechanical/molecular mechanical interface. *J. Chem. Theory Comput.*, **2009**, *5*, 2206–2211.

[119] G. te Velde; F. M. Bickelhaupt; E. J. Baerends; C. F. Guerra; S. J. A. van Gisbergen; J. G. Snijders; T. Ziegler. Chemistry with ADF. *J. Comp. Chem.*, **2001**, *22*, 931–967.

[120] ADF2011, SCM, Theoretical Chemistry, Vrije Universiteit, Amsterdam, The Netherlands, http://www.scm.com, 2012.

[121] M. W. Schmidt; K. K. Baldridge; J. A. Boatz; S. T. Elbert; M. S. Gordon; J. H. Jensen; S. Koseki; N. Matsunaga; K. A. Nguyen; S. Su; T. L. Windus; M. Dupuis; J. A. Montgomery, Jr. General atomic and molecular electronic structure system. *J. Comp. Chem.*, **1993**, *14*, 1347–1363.

[122] M. S. Gordon; M. W. Schmidt. in *Theory and Applications of Computational Chemistry, the first forty years*, C. E. Dykstra; G. Frenking; K. S. Kim; G. E. Scuseria, Eds., chapter 41, pp 1167–1189. Elsevier, Asterdam, 2005.

[123] M. Valiev; E. J. Bylaska; N. Govind; K. Kowalski; T. P. Straatsma; H. J. J. van Dam; D. Wang; J. Niepolcha; E. Apra; T. L. Windus; W. A. de Jong. Nwchem: a comprehensive and scalable open-source solution for large scale molecular simulations. *Comput. Phys. Commun.*, **2010**, *181*, 1477.

[124] M. J. Frisch; G. W. Trucks; H. B. Schlegel; G. E. Scuseria; M. A. Robb; J. R. Cheeseman; G. Scalmani; V. Barone; B. Mennucci; G. A. Petersson; H. Nakatsuji; M. Caricato; X. Li; H. P. Hratchian; A. F. Izmaylov; J. Bloino; G. Zheng; J. L. Sonnenberg; M. Hada; M. Ehara; K. Toyota; R. Fukuda; J. Hasegawa; M. Ishida; T. Nakajima; Y. Honda; O. Kitao; H. Nakai; T. Vreven; J. A. Montgomery, Jr.; J. E. Peralta; F. Ogliaro; M. Bearpark; J. J. Heyd; E. Brothers; K. N. Kudin; V. N. Staroverov; R. Kobayashi; J. Normand; K. Raghavachari; A. Rendell; J. C. Burant; S. S. Iyengar; J. Tomasi; M. Cossi; N. Rega; J. M. Millam; M. Klene; J. E. Knox; J. B. Cross; V. Bakken; C. Adamo; J. Jaramillo; R. Gomperts; R. E. Stratmann; O. Yazyev; A. J. Austin; R. Cammi; C. Pomelli; J. W. Ochterski; R. L. Martin; K. Morokuma; V. G. Zakrzewski; G. A. Voth; P. Salvador; J. J. Dannenberg; S. Dapprich; A. D. Daniels; O. Farkas; J. B.

Foresman; J. V. Ortiz; J. Cioslowski; D. J. Fox. Gaussian 09 Revision A.1. Gaussian Inc. Wallingford CT 2009.

[125] F. Neese. ORCA - an ab initio, Density Functional and Semiempirical program package, Version 2.8.0, University of Bonn, 2010.

[126] Y. Shao; L. Fusti-Molnar; Y. Jung; J. Kussmann; C. Ochsenfeld; S. T. Brown; A. T. B. Gilbert; L. V. Slipchenko; S. V. Levchenko; D. P. O'Neill; R. A. DiStasio, Jr.; R. C. Lochan; T. Wang; G. J. O. Beran; N. A. Besley; J. M. Herbert; C. Y. Lin; T. V. Voorhis; S. H. Chien; A. Sodt; R. P. Steele; V. A. Rassolov; P. E. Maslen; P. P. Korambath; R. D. Adamson; B. Austin; J. Baker; E. F. C. Byrd; H. Daschel; R. J. Doerksen; A. Dreuw; B. D. Dunietz; A. D. Dutoi; T. R. Furlani; S. R. Gwaltney; A. Heyden; S. Hirata; C.-P. Hsu; G. Kedziora; R. Z. Khaliullin; P. Klunzinger; A. M. Lee; M. S. Lee; W. Liang; I. Lotan; N. Nair; B. Peters; E. I. Proynov; P. A. Pieniazek; Y. M. Rhee; J. Ritchie; E. Rosta; C. D. Sherrill; A. C. Simmonett; J. E. Subotnik; H. L. Woodcock, III; W. Zhang; A. T. Bell; A. K. Chakraborty; D. M. Chipman; F. J. Keil; A. Warshel; W. J. Hehre; H. F. Schaefer, III; J. Kong; A. I. Krylov; P. M. W. Gill; M. Head-Gordon. Advances in methods and algorithms in a modern quantum chemistry program package. *Phys. Chem. Chem. Phys.*, **2006**, *8*, 3172–3191.

[127] I. S. Ufimtsev; T. J. Martinez. Quantum chemistry on graphical processing units. 3. Analytical energy gradients, geometry optimization, and first principles molecular dynamics. *J. Chem. Theory Comput.*, **2009**, *5*, 2619–2628.

[128] M. Kállay; Z. Rolik; J. Csontos; I. Ladjánszki; L. Szegedy; B. Ladoćzki; G. Samu; K. Petrov; M. Farkas; P. Nagy; D. Mester; B. Hegely. Mrcc, a quantum chemical program suite. www.mrcc.hu.

[129] Z. Rolik; L. Szegedy; I. Ladjánszki; B. Ladóczki; M. Kállay. An efficient linear-scaling CCSD(T) method based on local natural orbitals. *J. Chem. Phys.*, **2013**, *139*, 094105.

[130] J. P. Lewis; P. Jelínek; J. Ortega; A. A. Demkov; D. G. Trabada; B. Haycock; H. Wang; G. Adams; J. K. Tomfohr; E. Abad; H. Wang; D. A. Drabold. Advances and applications in the FIREBALL ab initio tight-binding molecular-dynamics formalism. *Phys. Status Solidi B*, **2011**, *248*, 1989–2007.

[131] J. Torras; Y. He; C. Cao; K. Muralidharan; E. Deumens; H. Cheng; S. Trickey. PUPIL: A systematic approach to software integration in multi-scale simulations. *Comput. Phys. Comm.*, **2007**, *177*, 265–279.

[132] J. Torras; G. Seabra; E. Deumens; S. B. Trickey; A. E. Roitberg. A versatile AMBER-Gaussian QM/MM interface through PUPIL. *J. Comput. Chem.*, **2008**, *29*, 1564–1573.

[133] B. Hégely; P. R. Nagy; G. G. Ferenczy; M. Kállay. Exact density functional and wave function embedding schemes based on orbital localization. *J. Chem. Phys.*, **2016**, *145*, 064107.

[134] D. S. Cerutti; D. A. Case. Molecular dynamics simulations of macromolecular crystals. *Wires Comput. Mol. Sci.*, **2018**, *8*, e1402.

[135] H. Kopitz; A. Zivkovic; J. W. Engels; H. Gohlke. Determinants of the unexpected stability of RNA fluorobenzene self pairs. *ChemBioChem*, **2008**, *9*, 2619–2622.

[136] C. W. Hopkins; S. Le Grand; R. C. Walker; A. E. Roitberg. Long-Time-Step Molecular Dynamics through Hydrogen Mass Repartitioning. *J. Chem. Theory Comput.*, **2015**, *11*, 1864–1874.

[137] T. Morishita. Fluctuation formulas in molecular-dynamics simulations with the weak coupling heat bath. *J. Chem. Phys.*, **2000**, *113*, 2976.

[138] A. Mudi; C. Chakravarty. Effect of the Berendsen thermostat on the dynamical properties of water. *Mol. Phys.*, **2004**, *102*, 681–685.

[139] H. J. C. Berendsen; J. P. M. Postma; W. F. van Gunsteren; A. DiNola; J. R. Haak. Molecular dynamics with coupling to an external bath. *J. Chem. Phys.*, **1984**, *81*, 3684–3690.

[140] S. C. Harvey; R. K. Tan; T. E. Cheatham, III. The flying ice cube: Velocity rescaling in molecular dynamics leads to violation of energy equipartition. *J. Comput. Chem.*, **1998**, *19*, 726–740.

[141] T. A. Andrea; W. C. Swope; H. C. Andersen. The role of long ranged forces in determining the structure and properties of liquid water. *J. Chem. Phys.*, **1983**, *79*, 4576–4584.

[142] H. C. Andersen. Molecular dynamics simulations at constant pressure and/or temperature. *J. Chem. Phys.*, **1980**, *72*, 2384–2393.

[143] B. P. Uberuaga; M. Anghel; A. F. Voter. Synchronization of trajectories in canonical molecular-dynamics simulations: Observation, explanation, and exploitation. *J. Chem. Phys.*, **2004**, *120*, 6363–6374.

[144] D. J. Sindhikara; S. Kim; A. F. Voter; A. E. Roitberg. Bad seeds sprout perilous dynamics: Stochastic thermostat induced trajectory synchronization in biomolecules. *J. Chem. Theory Comput.*, **2009**, *5*, 1624–1631.

[145] I. Omelyan; A. Kovalenko. Generalized canonical-isokinetic ensemble: Speeding up multiscale molecular dynamics and coupling with 3d molecular theory of solvation. *Mol. Sim.*, **2013**, *39*, 25–48.

[146] I. Omelyan; A. Kovalenko. Multiple time step molecular dynamics in the optimized isokinetic ensemble steered with the molecular theory of solvation: Accelerating with advanced extrapolation of effective solvation forces. *J. Chem. Phys.*, **2013**, *139*, 244106.

[147] B. Leimkuhler; D. T. Margul; M. E. Tuckerman. Stochastic, Resonance-Free Multiple Time-Step Algorithm for Molecular Dynamics with Very Large Time Steps. *Mol. Phys.*, **2013**, *111*, 3579–3594.

[148] G. Bussi; D. Donadio; M. Parrinello. Canonical sampling through velocity rescaling. *The Journal of chemical physics*, **2007**, *126*, 014101.

[149] R. W. Pastor; B. R. Brooks; A. Szabo. An analysis of the accuracy of Langevin and molecular dynamics algorithms. *Mol. Phys.*, **1988**, *65*, 1409–1419.

[150] R. J. Loncharich; B. R. Brooks; R. W. Pastor. Langevin dynamics of peptides: The frictional dependence of isomerization rates of N-actylananyl-N'-methylamide. *Biopolymers*, **1992**, *32*, 523–535.

[151] Y. M. Rhee; V. S. Pande. Solvent viscosity dependence of the protein folding dynamics. *J. Phys. Chem. B*, **2008**, *112*, 6221–6227. PMID: 18229911.

[152] J. A. Izaguirre; D. P. Catarello; J. M. Wozniak; R. D. Skeel. Langevin stabilization of molecular dynamics. *J. Chem. Phys.*, **2001**, *114*, 2090–2098.

[153] R. G. FernÃ¡ndez; J. L. F. Abascal; C. Vega. The melting point of ice Ih for common water models calculated from direct coexistence of the solid-liquid interface. *The Journal of Chemical Physics*, **2006**, *124*, 144506.

[154] Y. Zhang; S. E. Feller; B. R. Brooks; R. W. Pastor. Computer simulation of liquid/liquid interfaces. I. Theory and application to octane/water. *J. Chem. Phys.*, **1995**, *103*, 10252–10266.

[155] J.-P. Ryckaert; G. Ciccotti; H. J. C. Berendsen. Numerical integration of the cartesian equations of motion of a system with constraints: Molecular dynamics of n-alkanes. *J. Comput. Phys.*, **1977**, *23*, 327–341.

[156] S. Miyamoto; P. A. Kollman. SETTLE: An analytical version of the SHAKE and RATTLE algorithm for rigid water models. *J. Comput. Chem.*, **1992**, *13*, 952–962.

[157] Z. Zhang; X. Liu; Z. Chen; H. Zheng; K. Yan; J. Liu. A unified thermostat scheme for efficient configurational sampling for classical/quantum canonical ensembles via molecular dynamics. *J. Chem. Phys.*, **2017**, *147*, 034109.

[158] J. Liu; D. Li; X. Liu. A simple and accurate algorithm for path integral molecular dynamics with the Langevin thermostat. *J. Chem. Phys.*, **2016**, *145*, 024103.

[159] D. Li; X. Han; Y. Chai; C. Wang; Z. Zhang; Z. Chen; J. Liu; J. Shao. Stationary state distribution and efficiency analysis of the Langevin equation via real or virtual dynamics. *J. Chem. Phys.*, **2017**, *147*, 184104.

[160] D. Li; Z. Chen; Z. Zhang; J. Liu. Understanding molecular dynamics with stochastic processes via real or virtual dynamics. *Chinese Journal of Chemical Physics*, **2017**, *30*, 735–760.

[161] Z. Zhang; K. Yan; X. Liu; J. Liu. A leap-frog algorithm-based efficient unified thermostat scheme for molecular dynamics. *Chinese Science Bulletin*, **2018**, *63*, 3467–3483.

[162] B. Leimkuhler; C. Matthews. Rational construction of stochastic numerical methods for molecular sampling. *Appl. Math. Res. eXpress*, **2013**, *2013*, 34–56.

[163] N. Gr"ønbech-Jensen; O. Farago. A simple and effective Verlet-type algorithm for simulating Langevin dynamics. *Mol. Phys.*, **2013**, *111*, 983–991.

[164] X. Liu; J. Liu. Critical role of quantum dynamical effects in the Raman spectroscopy of liquid water. *Mol. Phys.*, **2018**, *116*, 755–779.

[165] H. C. Andersen. RATTLE: A "velocity" version of the shake algorithm for molecular dynamics calculations. *J. Computat. Phys.*, **1983**, *52*, 24 – 34.

[166] Z. Zhang; X. Liu; K. Yan; M. Tuckerman; J. Liu. Unified efficient thermostat scheme for the canonical ensemble with holonomic or isokinetic constraints via molecular dynamics. *J. Phys. Chem. A*, **2019**, *123*, 6056–6079.

[167] X. Wu; S. Subramaniam; D. A. Case; K. Wu; B. R. Brooks. Targeted conformational search with map-restrained self-guided langevin dynamics: application to flexible fitting into electron microscopic density maps. *J. Struct. Biology*, **2013**, *183*, 429–440.

[168] P. Ren; J. W. Ponder. Consistent treatment of inter- and intramolecular polarization in molecular mechanics calculations. *J. Comput. Chem.*, **2002**, *23*, 1497–1506.

[169] P. Ren; J. W. Ponder. Temperature and pressure dependence of the AMOEBA water model. *J. Phys. Chem. B*, **2004**, *108*, 13427–13437.

[170] P. Li; K. M. Merz, Jr. Taking into Account the Ion-Induced Dipole Interaction in the Nonbonded Model of Ions. *J. Chem. Theory Comput.*, **2014**, *10*, 289–297.

[171] T. Darden; D. York; L. Pedersen. Particle mesh Ewald–an Nlog(N) method for Ewald sums in large systems. *J. Chem. Phys.*, **1993**, *98*, 10089–10092.

[172] U. Essmann; L. Perera; M. L. Berkowitz; T. Darden; H. Lee; L. G. Pedersen. A smooth particle mesh Ewald method. *J. Chem. Phys.*, **1995**, *103*, 8577–8593.

[173] M. F. Crowley; T. A. Darden; T. E. Cheatham, III; D. W. Deerfield, II. Adventures in improving the scaling and accuracy of a parallel molecular dynamics program. *J. Supercomput.*, **1997**, *11*, 255–278.

[174] C. Sagui; T. A. Darden. in *Simulation and Theory of Electrostatic Interactions in Solution*, L. R. Pratt; G. Hummer, Eds., pp 104–113. American Institute of Physics, Melville, NY, 1999.

[175] A. Toukmaji; C. Sagui; J. Board; T. Darden. Efficient particle-mesh Ewald based approach to fixed and induced dipolar interactions. *J. Chem. Phys.*, **2000**, *113*, 10913–10927.

[176] C. Sagui; L. G. Pedersen; T. A. Darden. Towards an accurate representation of electrostatics in classical force fields: Efficient implementation of multipolar interactions in biomolecular simulations. *J. Chem. Phys.*, **2004**, *120*, 73–87.

[177] X. Wu; B. R. Brooks. Isotropic periodic sum: A method for the calculation of long-range interactions. *J. Chem. Phys.*, **2005**, *122*, 044107.

[178] J. B. Klauda; X. Wu; R. W. Pastor; B. R. Brooks. Long-Range Lennard-Jones and Electrostatic Interactions in Interfaces. *J. Phys. Chem. B*, **2007**, *111*, 4393–4400.

[179] K. Takahashi; K. Yasuoka; T. Narumi. Cutoff radius effect of isotropic periodic sum method for transport. *J. Chem. Phys.*, **2007**, *127*, 114511.

[180] X. Wu; B. R. Brooks. Using the Isotropic Periodic Sum Method to Calculate Long-Range Interactions of Heterogeneous Systems. *J. Chem. Phys.*, **2008**, *129*, 154115.

[181] X. Wu; B. R. Brooks. Isotropic periodic sum of electrostatic interactions for polar systems. *J. Chem. Phys.*, **2009**, *131*, 024107.

[182] R. M. Venable; L. E. Chen; R. W. Pastor. Comparison of the Extended Isotropic Periodic Sum and Particle Mesh Ewald Methods for Simulations of Lipid Bilayers and Monolayers. *J. Phys. Chem. B*, **2009**, *113*, 5855–5862.

[183] H. Wei; R. Qi; J. Wang; P. Cieplak; Y. Duan; R. Luo. Efficient Formulation of polarizable Gaussian Multipole Electrostatics for Biomolecular Simulations. *J. Chem. Phys.*, **2020**, *153*, 114116.

[184] X. Wu; B. R. Brooks. Self-guided Langevin dynamics simulation method. *Chem. Phys. Lett.*, **2003**, *381*, 512–518.

[185] X. Wu; A. Damjanovic; B. R. Brooks. Efficient and unbiased sampling of biomolecular systems in the canonical ensemble: a review of self-guided langevin dynamics. *Adv. Chem. Phys.*, **2012**, *150*, 255–326.

[186] X. Yu; X. Wu; G. A. Bermejo; B. R. Brooks; J. W. Taraska. Accurate high-throughput structure mapping and prediction with transition metal ion fret. *Structure*, **2013**, *21*, 9–19.

[187] X. Wu; B. R. Brooks. Self-guided langevin dynamics via generalized langevin equation. *J. Comput. Chem.*, **2016**, *37*, 595–601.

[188] X. Wu; B. R. Brooks. Toward canonical ensemble distribution from self-guided Langevin dynamics simulation. *J. Chem. Phys.*, **2011**, *134*, 134108.

[189] X. Wu; B. R. Brooks. Force-momentum-based self-guided Langevin dynamics: a rapid sampling method that approaches the canonical ensemble . *J. Chem. Phys.*, **2011**, *135*, 204101.

[190] X. Wu; M. Hodoscek; B. R. Brooks. Replica exchanging self-guided Langevin dynamics for efficient and accurate conformational sampling. *J. Chem. Phys.*, **2012**, *137*, 044106.

[191] D. Hamelberg; J. Mongan; J. A. McCammon. Accelerated molecular dynamics: A promising and efficient simulation method for biomolecules. *J. Chem. Phys.*, **2004**, *120*, 11919–11929.

[192] D. Hamelberg; C. A. F. de Oliveira; J. McCammon. Sampling of slow diffusive conformational transitions with accelerated molecular dynamics. *J. Chem. Phys.*, **2007**, *127*, 155102–155109.

[193] B. J. Grant; A. A. Gorfe; J. A. McCammon. Ras conformational switching: Simulating nucleotide-dependent conformational transitions with accelerated molecular dynamics. *PLoS Computat. Biol.*, **2009**, *5*, e1000325.

[194] C. A. F. de Oliveira; B. J. Grant; M. Zhou; J. A. McCammon. Large-scale conformational changes of trypanosoma cruzi proline racemase predicted by accelerated molecular dynamics simulation. *PLoS Computat. Biol.*, **2011**, *7*, e1002178.

[195] L. C. T. Pierce; R. Salomon-Ferrer; C. A. F. de Oliveira; J. A. McCammon; R. C. Walker. Routine access to milli-second time scales with accelerated molecular dynamics. *J. Chem. Theory Comput.*, **2012**, *8*, 2997–3002.

[196] U. Doshi; D. Hamelberg. Reoptimization of the amber forcefield for peptide bond (omega) torsions using accelerated molecular dynamics. *J. Chem. Phys. B*, **2009**, *113*, 16590–16595.

[197] Y. Miao; V. A. Feher; J. A. McCammon. Gaussian Accelerated Molecular Dynamics: Unconstrained Enhanced Sampling and Free Energy Calculation. *J. Chem. Theory Comput.*, **2015**, *11*, 3584–3595.

[198] Y. Miao; W. Sinko; L. Pierce; D. Bucher; R. C. Walker; J. A. McCammon. Improved Reweighting of Accelerated Molecular Dynamics Simulations for Free Energy Calculation. *J. Chem. Theory Comput.*, **2014**, *10*, 2677–2689.

[199] Y. Miao; A. Bhattarai; J. Wang. Ligand Gaussian accelerated molecular dynamics (LiGaMD): Characterization of ligand binding thermodynamics and kinetics. *bioRxiv*, **2020**.

[200] I. Kolossváry; W. C. Guida. Low mode search. An efficient, automated computational method for conformational analysis: Application to cyclic and acyclic alkanes and cyclic peptides. *J. Am. Chem. Soc.*, **1996**, *118*, 5011–5019.

[201] I. Kolossváry; W. C. Guida. Low-mode conformational search elucidated: Application to $C_{39}H_{80}$ and flexible docking of 9-deazaguanine inhibitors into PNP. *J. Comput. Chem.*, **1999**, *20*, 1671–1684.

[202] I. Kolossváry; G. M. Keserü. Hessian-free low-mode conformational search for large-scale protein loop optimization: Application to c-jun N-terminal kinase JNK3. *J. Comput. Chem.*, **2001**, *22*, 21–30.

[203] G. M. Keserü; I. Kolossváry. Fully flexible low-mode docking: Application to induced fit in HIV integrase. *J. Am. Chem. Soc.*, **2001**, *123*, 12708–12709.

[204] W. H. Press; B. P. Flannery; S. A. Teukolsky; W. T. Vetterling. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, New York, 1989.

[205] D. C. Liu; J. Nocedal. On the limited memory method for large scale optimization. *Math. Programming B*, **1989**, *45*, 503–528.

[206] J. Nocedal; J. L. Morales. Automatic preconditioning by limited memory quasi-Newton updating. *SIAM J. Opt.*, **2000**, *10*, 1079–1096.

[207] P. Kollman. Free energy calculations: Applications to chemical and biochemical phenomena. *Chem. Rev.*, **1993**, *93*, 2395–2417.

[208] T. Simonson. in *Computational Biochemistry and Biophysics*, O. Becker; A. D. MacKerell; B. Roux; M. Watanabe, Eds. Marcel Dekker, New York, 2001.

[209] D. Frenkel; B. Smit. *Understanding Molecular Simulation: From Algorithms to Applications. Second edition*. Academic Press, San Diego, 2002.

[210] T. Steinbrecher; D. A. Case; A. Labahn. A multistep approach to structure-based drug design: Studying ligand binding at the human neutrophil elastase. *J. Med.. Chem.*, **2006**, *49*, 1837–1844.

[211] T. Steinbrecher; A. Hrenn; K. Dormann; I. Merfort; A. Labahn. Bornyl (3,4,5-trihydroxy)-cinnamate - An optimized human neutrophil elastase inhibitor designed by free energy calculations. *Bioorg. Med. Chem.*, **2008**, *16*, 2385–2390.

[212] G. Hummer; A. Szabo. Calculation of free-energy differences from computer simulations of initial and final states. *J. Chem. Phys.*, **1996**, *105*, 2004–2010.

[213] T. Steinbrecher; D. L. Mobley; D. A. Case. Non-linear scaling schemes for Lennard-Jones interactions in free energy calculations. *J. Chem. Phys.*, **2007**, *127*, 214108.

[214] J. Kaus; L. C. T. Pierce; R. C. Walker; J. A. McCammon. PMEMD TI: Placeholder. *J. Chem. Theory Comput.*, **2013**.

[215] T. Steinbrecher; I. Joung; D. A. Case. Soft-core potentials in thermodynamic integration: Comparing one- and two-step transformations. *J. Comp. Chem.*, **2011**, *32*, 3253–3263.

[216] V. Babin; C. Roland; C. Sagui. Adaptively biased molecular dynamics for free energy calculations. *J. Chem. Phys.*, **2008**, *128*, 134101.

[217] T. Huber; A. E. Torda; W. F. van Gunsteren. Local elevation: a method for improving the searching properties of molecular dynamics simulation. *J. Comput. Aided. Mol. Des.*, **1994**, *8*, 695–708.

[218] F. Wang; D. P. Landau. Efficient, multiple-range random walk algorithm to calculate the density of states. *Phys. Rev. Lett.*, **2001**, *86*, 2050–2053.

[219] A. Laio; M. Parrinello. Escaping free-energy minima. *Proc. Natl. Acad. Sci.*, **2002**, *99*, 12562–12566.

[220] M. Iannuzzi; A. Laio; M. Parrinello. Efficient exploration of reactive potential energy surfaces using car-parrinello molecular dynamics. *Phys. Rev. Lett.*, **2003**, *90*, 238302–1.

[221] T. Lelièvre; M. Rousset; G. Stoltz. Computation of free energy profiles with parallel adaptive dynamics. *J. Chem. Phys.*, **2007**, *126*, 134111.

[222] P. Raiteri; A. Laio; F. L. Gervasio; C. Micheletti; M. Parrinello. Efficient reconstruction of complex free energy landscapes by multiple walkers metadynamics. *J. Phys. Chem.*, **2006**, *110*, 3533–3539.

[223] Y. Sugita; A. Kitao; Y. Okamoto. Multidimensional replica-exchange method for free-energy calculations. *J. Chem. Phys.*, **2000**, *113*, 6042–6051.

[224] G. Bussi; F. L. Gervasio; A. Laio; M. Parrinello. Free-energy landscape for β hairpin folding from combined parallel tempering and metadynamics. *J. Am. Chem. Soc.*, **2006**, *128*, 13435–13441.

[225] S. Piana; A. Laio. A bias-exchange approach to protein folding. *J. Phys. Chem. B*, **2007**, *111*, 4553–4559.

[226] V. Babin; C. Roland; T. A. Darden; C. Sagui. The free energy landscape of small peptides as obtained from metadynamics with umbrella sampling corrections. *J. Chem. Phys.*, **2006**, *125*, 2049096.

[227] V. Babin; V. Karpusenka; M. Moradi; C. Roland; C. Sagui. Adaptively biased molecular dynamics: an umbrella sampling method with a time dependent potential. *Inter. J. Quantum Chem.*, **2009**, *109*, 3666–3678.

[228] V. Babin; C. Sagui. Conformational free energies of methyl-alpha-l-iduronic and methyl-beta-d-glucuronic acids in water. *J. Chem. Phys.*, **2010**, *132*, 104108.

[229] M. Moradi; V. Babin; C. Roland; T. Darden; C. Sagui. Conformations and free energy landscapes of polyproline peptides. *Proc. Natl. Aca. Sci. USA*, **2009**, *106*, 20746.

[230] M. Moradi; V. Babin; C. Roland; C. Sagui. A classical molecular dynamics investigation of the free energy and structure of short polyproline conformers. *J. Chem. Phys.*, **2010**, *133*, 125104.

[231] M. Moradi; V. Babin; C. Sagui; C. Roland. in *Proline: Biosynthesis, Regulation and Health Benefits*, B. Nedjimi, Ed., pp 67–110. Nova Publishers, 2013.

[232] M. Moradi; V. Babin; C. Sagui; C. Roland. A statistical analysis of the PPII propensity of amino acid guests in proline-rich peptides. *Biophysical J.*, **2011**, *100*, 1083 – 1093.

[233] M. Moradi; V. Babin; C. Sagui; C. Roland. PPII propensity of multiple-guest amino acids in a proline-rich environment. *J. Phys. Chem. B.*, **2011**, *115*, 8645–8656.

[234] V. Babin; C. Roland; C. Sagui. The alpha-sheet: A missing-in-action secondary structure? *Proteins –Structure Function and Bioinformatics*, **2011**, *79*, 937–946.

[235] M. Moradi; V. Babin; C. Roland; C. Sagui. Are long-range structural correlations behind the aggregation phenomena of polyglutamine diseases? *PLoS Comput. Biol.*, **2012**, *8*, e1002501.

[236] M. Moradi; V. Babin; C. Roland; C. Sagui. Reaction path ensemble of the B-Z-DNA transition: a comprehensive atomistic study. *Nucleic Acids Research*, **2013**, *41*, 33–43.

[237] F. Pan; V. H. Man; C. Roland; C. Sagui. Structure and Dynamics of DNA and RNA Double Helices of CAG and GAC Trinucleotide Repeats. *Biophys. J.*, **2017**, *113*, 19–36.

[238] F. Pan; Y. Zhang; V. H. Man; C. Roland; C. Sagui. E-motif formed by extrahelical cytosine bases in DNA homoduplexes of trinucleotide and hexanucleotide repeats. *Nucl. Acids Res.*, **2018**, *46*, 942–955.

[239] F. Pan; V. H. Man; C. Roland; C. Sagui. Structure and Dynamics of DNA and RNA Double Helices Obtained From the CCG and GGC Trinucleotide Repeats. *J. Phys. Chem. B*, **2018**, *122*, 4491–4512.

[240] P. Xu; F. Pan; C. Roland; C. Sagui; K. Weninger. Dynamics of strand slippage in DNA hairpins formed by CAG repeats: roles of sequence parity and trinucleotide interrupts. *Nucl. Acids Res.*, **2020**, *48*, 2232–2245.

[241] M. Moradi; J.-G. Lee; V. Babin; C. Roland; C. Sagui. Free energy and structure of polyproline peptides: an ab initio and classical molecular dynamics investigation. *Int. J. Quantum Chem.*, **2010**, *110*, 2865–2879.

[242] M. Moradi; C. Sagui; C. Roland. Calculating relative transition rates with driven nonequilibrium simulations. *Chem. Phys. Lett.*, **2011**, *518*, 109.

[243] M. Moradi; C. Sagui; C. Roland. Investigating rare events with nonequilibrium work measurements: I. Nonequilibrium transition paths. *J. Chem. Phys.*, **2014**, *140*, 034114.

[244] M. Moradi; C. Sagui; C. Roland. Investigating rare events with nonequilibrium work measurements: II. Transition and reaction rates. *J. Chem. Phys.*, **2014**, *140*, 034115.

[245] M. Moradi; E. Tajkorshid. Driven Metadynamics: Reconstructing equilibrium free energies from driven adaptive-bias simulations. *J. Phys. Chem. Lett.*, **2013**, *4*, 1882.

[246] A. C. Pan; D. Sezer; B. Roux. Finding transition pathways using the string method with swarms of trajectories. *J. Phys. Chem. B*, **2008**, *112*, 3432–3440.

[247] A. Barducci and G. Bussi and M. Parrinello. Well-tempered metadynamics: a smoothly converging and tunable free energy method. *Phys. Rev. Lett.*, **2008**, *100*, 020603.

[248] K. Minoukadeh and Ch. Chipot and T. Lelievre. Potential of Mean Force Calculations: A multiple-walker adaptive biasing force technique. *J. Chem. Theor. and Comput.*, **2010**, *6*, 1008.

[249] E. A. Coutsias; C. Seok; K. A. Dill. Using quaternions to calculate RMSD. *J. Comput. Chem.*, **2004**, *25*, 1849–1857.

[250] D. K. Coutsias EA, Seok C. Using quaternions to calculate rmsd. *J Comput Chem*, **2004 Nov 30**, *25*, 1849–57.

[251] G. Fiorin; M. L. Klein; J. Hénin. Using collective variables to drive molecular dynamics simulations. *Molecular Physics*, **2013**, *111*, 3345–3362.

[252] M. Moradi; E. Tajkhorshid. Mechanistic picture for conformational transition of a membrane transporter at atomic resolution. *Proc. Natl. Acad. Sci. U.S.A.*, **2013**, *110*, 18916–18921.

[253] M. Moradi; E. Tajkhorshid. Computational Recipe for Efficient Description of Large-Scale Conformational Changes in Biomolecular Systems. *J Chem Theory Comput*, **2014**, *10*, 2866–2880.

[254] S. Park; F. Khalili-Araghi; E. Tajkhorshid; K. Schulten. Free energy calculation from steered molecular dynamics simulations using Jarzynski's equality. *J. Chem. Phys.*, **2003**, *119*, 3559–3566.

[255] M. Matsumoto; T. Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, **1998**, *8*, 3–30.

[256] L. Maragliano; A. Fischer; E. Vanden-Eijnden; G. Ciccotti. String method in collective variables: Minimum free energy paths and isocommittor surfaces. *J. Chem. Phys.*, **2006**, *125*, 024106.

[257] B. M. Duggan; G. B. Legge; H. J. Dyson; P. E. Wright. SANE (Structure Assisted NOE Evaluation): An automated model-based approach for NOE assignment. *J. Biomol. NMR*, **2001**, *19*, 321–329.

[258] A. Kalk; H. J. C. Berendsen. Proton magnetic relaxation and spin diffusion in proteins. *J. Magn. Reson.*, **1976**, *24*, 343–366.

[259] E. T. Olejniczak; M. A. Weiss. Are methyl groups relaxation sinks in small proteins? *J. Magn. Reson.*, **1990**, *86*, 148–155.

[260] K. J. Cross; P. E. Wright. Calibration of ring-current models for the heme ring. *J. Magn. Reson.*, **1985**, *64*, 220–231.

[261] K. Ösapay; D. A. Case. A new analysis of proton chemical shifts in proteins. *J. Am. Chem. Soc.*, **1991**, *113*, 9436–9444.

[262] D. A. Case. Calibration of ring-current effects in proteins and nucleic acids. *J. Biomol. NMR*, **1995**, *6*, 341–346.

[263] L. Banci; I. Bertini; G. Gori-Savellini; A. Romagnoli; P. Turano; M. A. Cremonini; C. Luchinat; H. B. Gray. Pseudocontact shifts as constraints for energy minimization and molecular dynamics calculations on solution structures of paramagnetic metalloproteins. *Proteins*, **1997**, *29*, 68.

[264] C. R. Sanders, II; B. J. Hare; K. P. Howard; J. H. Prestegard. Magnetically-oriented phospholipid micelles as a tool for the study of membrane-associated molecules. *Prog. NMR Spectr.*, **1994**, *26*, 421–444.

[265] V. Tsui; L. Zhu; T. H. Huang; P. E. Wright; D. A. Case. Assessment of zinc finger orientations by residual dipolar coupling constants. *J. Biomol. NMR*, **2000**, *16*, 9–21.

[266] D. A. Case. Calculations of NMR dipolar coupling strengths in model peptides. *J. Biomol. NMR*, **1999**, *15*, 95–102.

[267] G. P. Gippert; P. F. Yip; P. E. Wright; D. A. Case. Computational methods for determining protein structures from NMR data. *Biochem. Pharm.*, **1990**, *40*, 15–22.

[268] D. A. Case; P. E. Wright. in *NMR in Proteins*, G. M. Clore; A. Gronenborn, Eds., pp 53–91. MacMillan, New York, 1993.

[269] D. A. Case; H. J. Dyson; P. E. Wright. Use of chemical shifts and coupling constants in nuclear magnetic resonance structural studies on peptides and proteins. *Meth. Enzymol.*, **1994**, *239*, 392–416.

[270] R. Brüschweiler; D. A. Case. Characterization of biomolecular structure and dynamics by NMR cross-relaxation. *Prog. NMR Spectr.*, **1994**, *26*, 27–58.

[271] D. A. Case. The use of chemical shifts and their anisotropies in biomolecular structure determination. *Curr. Opin. Struct. Biol.*, **1998**, *8*, 624–630.

[272] A. E. Torda; R. M. Scheek; W. F. VanGunsteren. Time-dependent distance restraints in molecular dynamics simulations. *Chem. Phys. Lett.*, **1989**, *157*, 289–294.

[273] D. A. Pearlman; P. A. Kollman. Are time-averaged restraints necessary for nuclear magnetic resonance refinement? A model study for DNA. *J. Mol. Biol.*, **1991**, *220*, 457–479.

[274] A. E. Torda; R. M. Brunne; T. Huber; H. Kessler; W. F. van Gunsteren. Structure refinement using time-averaged J-coupling constant restraints. *J. Biomol. NMR*, **1993**, *3*, 55–66.

[275] D. A. Pearlman. How well to time-averaged J-coupling restraints work? *J. Biomol. NMR*, **1994**, *4*, 279–299.

[276] D. A. Pearlman. How is an NMR structure best defined? An analysis of molecular dynamics distance-based approaches. *J. Biomol. NMR*, **1994**, *4*, 1–16.

[277] X. Wu; J. L. Milne; M. J. Borgnia; A. V. Rostapshov; S. Subramaniam; B. R. Brooks. A core-weighted fitting method for docking atomic structures into low-resolution maps: application to cryo-electron microscopy. *J Struct Biol*, **2003**, *141*, 63–76.

[278] J. L. Milne; X. Wu; M. J. Borgnia; J. S. Lengyel; B. R. Brooks; D. Shi; R. N. Perham; S. Subramaniam. Molecular structure of a 9-MDa icosahedral pyruvate dehydrogenase subcomplex containing the E2 and E3 enzymes using cryoelectron microscopy. *J Biol Chem*, **2006**, *281*, 4364–70.

[279] X. Wu; B. R. Brooks. Modeling of Macromolecular assemblies with map objects. *Proc. 2007 Int. Conf. Bioinform. Comput. Biol.*, **2007**, *II*, 411–417.

[280] C. M. Khursigara; X. Wu; P. Zhang; J. Lefman; S. Subramaniam. Role of HAMP domains in chemotaxis signaling by bacterial chemoreceptors. *PNAS*, **2008**, *105*, 16555–60.

[281] J. S. Lengyel; K. M. Stott; X. Wu; A. Brooks, B. R.and Balbo; P. Schuck; R. N. Perham; S. Subramaniam; J. L. Milne. Extended polypeptide linkers establish the spatial architecture of a pyruvate dehydrogenase multienzyme complex. *Structure*, **2008**, *16*, 93–103.

[282] C. M. Khursigara; X. Wu; ; S. Subramaniam. Chemoreceptors in Caulobacter crescentus: trimers of receptor dimers in a partially ordered hexagonally packed array. *J Bacteriol*, **2008**, *190*, 6805–10.

[283] J. Elegheert; A. Desfosses; A. V. Shkumatov; X. Wu; N. Bracke; K. Verstraete; K. Van Craenenbroeck; B. R. Brooks; D. I. Svergun; B. Vergauwen; I. Gutsche; S. N. Savvides. Extracellular complexes of the hematopoietic human and mouse CSF-1 receptor are driven by common assembly principles . *Structure*, **2011**, *19*, 1762–72.

[284] C. M. Khursigara; G. Lan; S. Neumann; X. Wu; S. Ravindran; M. J. Borgnia; V. Sourjik; J. Milne; Y. Tu; S. Subramaniam. Lateral density of receptor arrays in the membrane plane influences sensitivity of the E. coli chemotaxis response. *Embo J*, **2011**, *30*, 1719–29.

[285] X. Wu; B. R. Brooks. in *Microscopy: advances in scientific research and education*, A. Mendez-Vilas, Ed., pp 39–47. Formatex Research Center, Spain, 2014.

[286] X. Wu; B. R. Brooks. in *Modern electron microscopy in physical and life science*, M. Janecek, Ed., chapter 12, pp 243–262. InTech, 2016.

[287] A. Bartesaghi; A. Merk; S. Banerjee; D. Matthies; X. Wu; J. L. S. Milne; S. Subramaniam. 2.2 a resolution cryo-em structure of beta-galactosidase in complex with a cell-permeant inhibitor. *Science*, **2015**, *348*, 1147–1151.

[288] P. A. Janowski; D. S. Cerutti; J. M. Holton; D. A. Case. Peptide crystal simulations reveal hidden dynamics. *J. Am. Chem. Soc.*, **2013**, *135*, 7938–7948.

[289] N. Moriarty; P. Janowski; J. Swails; H. Nguyen; J. Richardson; D. Case; P. Adams. Improved chemistry restraints for crystallographic refinement by integrating the Amber force field into Phenix. *Acta Cryst. D*, **2020**, *76*, 51–62.

[290] P. Afonine; A. Urzhumtsev. On a fast calculation of structure factors at a subatomic resolution. *Acta Cryst. A*, **2004**, *60*, 19–32.

[291] J. Jiang; A. Brünger. Protein hydration observed by X-ray diffraction: solvation properties of penicillopepsin and neuraminidase crystal structures. *J. Mol. Biol.*, **1994**, *243*, 100–115.

[292] A. Fokine; A. Urzhumtsev. Flat bulk-solvent model: obtaining optimal parameters. *Acta Cryst. D*, **2002**, *58*, 1387–1392.

[293] R. Grosse-Kunstleve; N. Sauter; N. Moriarty; P. Adams. The Computational Crystallography Toolbox: crystallographic algorithms in a reusable software framework. *J. Appl. Crystallogr.*, **2002**, *35*, 126–136.

[294] P. Afonine; R. Grosse-Kunstleve; P. Adams; A. Urzhumtsev. Bulk-solvent and overall scaling revisited: faster calculations, improved results. *Acta Cryst. D*, **2013**, *69*, 625–634.

[295] V. Lunin; T. Skovoroda. R-free likelihood-based estimates of errors for phases calculated from atomic models. *Acta Cryst. A*, **1995**, *51*, 880–887.

[296] P. V. Afonine; R. W. Grosse-Kunstleve; P. D. Adams. A robust bulk-solvent correction and anisotropic scaling procedure. *Acta Cryst. D*, **2005**, *61*, 850–855.

[297] Y. Xue; N. Skrynnikov. Ensemble MD simulations restrained via crystallographic data: accurate structure leads to accurate dynamics. *Prot. Sci.*, **2014**, *23*, 488–507.

[298] A. Raval; S. Piana; M. Eastwood; R. Dror; D. Shaw. Refinement of protein structure homology models via long, all-atom molecular dynamics simulations. *Proteins*, **2012**, *80*, 2071–2079.

[299] O. Mikhailovskii; Y. Xue; N. R. Skrynnikov. Modeling a unit cell: crystallographic refinement procedure using the biomolecular MD simulation platform *Amber*. *IUCrJ*, **2022**, *9*, 114–133.

[300] A. Roitberg; R. Elber. Modeling side chains in peptides and proteins: Application of the locally enhanced sampling and the simulated annealing methods to find minimum energy conformations. *J. Chem. Phys.*, **1991**, *95*, 9277.

[301] A. Ulitsky; R. Elber. The thermal equilibrium aspects of the time-dependent Hartree and the locally enhanced sampling approximations: Formal properties, a correction, and computational examples for rare gas clusters. *J. Chem. Phys.*, **1993**, *98*, 3380.

[302] C. Simmerling; T. Fox; P. A. Kollman. Use of Locally Enhanced Sampling in Free Energy Calculations: Testing and Application of the alpha to beta Anomerization of Glucose. *J. Am. Chem. Soc.*, **1998**, *120*, 5771–5782.

[303] C. Simmerling; J. L. Miller; P. A. Kollman. Combined locally enhanced sampling and particle mesh Ewald as a strategy to locate the experimental structure of a nonhelical nucleic acid. *J. Am. Chem. Soc.*, **1998**, *120*, 7149–7155.

[304] A. Miranker; M. Karplus. Functionality maps of binding sites: A multiple copy simultaneous search method. *Proteins: Str. Funct. Gen.*, **1991**, *11*, 29–34.

[305] X. Cheng; V. Hornak; C. Simmerling. Improved conformational sampling through an efficient combination of mean-field simulation approaches. *J. Phys. Chem. B*, **2004**, *108*.

[306] J. E. Straub; M. Karplus. Enery partitioning in the classical time-dependent Hartree approximation. *J. Chem. Phys.*, **1991**, *94*, 6737.

[307] J. I. Mendieta-Moreno; R. C. Walker; J. P. Lewis; P. Gómez-Puertas; J. Mendieta; J. Ortega. FIREBALL/AMBER: An efficient local-orbital DFT QM/MM method for biomolecular systems. *J. Chem. Theory Comput.*, **2014**, *10*, 2185–2193.

[308] S. N. Steinmann; R. Ferreira de Morais; A. W. Götz; P. Fleurat-Lessard; M. Iannuzzi; P. Sautet; C. Michel. *J. Chem. Theory Comput.*, **2018**.

[309] S. N. Steinmann; P. Fleurat-Lessard; A. W. Götz; C. Michel; R. Ferreira de Morais; P. Sautet. Molecular mechanics models for the image charge, a comment on "including image charge effects in the molecular dynamics simulations of molecules on metal surfaces". *J. Comput. Chem.*, **2017**, *38*, 2127–2129.

[310] Å. Skjevik; B. D. Madej; C. J. Dickson; C. Lin; K. Teigen; R. C. Walker; I. R. Gould. Simulations of lipid bilayer self-assembly using all-atom lipid force fields. *Phys. Chem. Chem. Phys.*, **2016**, *18*, 10573–10584.

[311] Å. Skjevik; B. D. Madej; C. J. Dickson; K. Teigen; R. C. Walker; I. R. Gould. All-atom lipid bilayer self-assembly with the amber and charmm lipid force fields. *Chem. Commun.*, **2015**, *51*, 4402–4405.

[312] H. T. Nguyen; S. A. Pabit; S. P. Meisburger; L. Pollack; D. A. Case. Accurate small and wide angle X-ray scattering profiles from atomic models of proteins and nucleic acids. *J. Chem. Phys.*, **2014**, *141*, 22D508.

[313] S. Park; J. P. Bardhan; B. Roux; L. Makowski. Simulated X-ray scattering of protein solutions using explicit-solvent models. *J. Chem. Phys.*, **2009**, *130*, 134114.

[314] R. Assaraf; M. Caffarel; A. C. Kollias. Chaotic versus nonchaotic stochastic dynamics in monte carlo simulations: A route for accurate energy differences in n-body systems. *Phys. Rev. Lett.*, **2011**, *106*, 150601.

[315] K. Park; A. W. Götz; R. C. Walker; F. Paesani. Application of adaptive qm/mm methods to molecular dynamics simulations of aqueous systems. *J. Chem. Theory Comput.*, **2012**, *8*, 2868–2877.

[316] R. E. Bulo; C. Michel; P. Fleurat-Lessard; P. Sautet. Multiscale modeling of chemistry in water: Are we there yet? *J. Chem. Theory Comput.*, **2013**, *9*, 5567–5577.

[317] R. E. Bulo; B. Ensing; J. Sikkema; L. Visscher. Toward a practical method for adaptive qm/mm simulations. *J. Chem. Theory Comput.*, **2009**, *9*, 2212–2221.

[318] B. E. Hingerty; S. Figueroa; T. L. Hayden; S. Broyde. Prediction of DNA Structure from Sequence: A Build-up Technique. *Biopolymers*, **1989**, *28*, 1195–1222.

[319] D. A. Erie; K. J. Breslauer; W. K. Olson. A Monte Carlo Method for Generating Structures of Short Single-Stranded DNA Sequences. *Biopolymers*, **1993**, *33*, 75–105.

[320] D. A. Pearlman; S.-H. Kim. Conformational Studies of Nucleic Acids I. A Rapid and Direct Method for Generating Coordinates from the Pseudorotation Angle. *J. Biomol. Struct. Dyn.*, **1985**, *3*, 85–98.

[321] D. A. Pearlman; S.-H. Kim. Conformational Studies of Nucleic Acids II. The Conformational Energetics of Commonly Occurring Nucleosides. *J. Biomol. Struct. Dyn.*, **1985**, *3*, 99–125.

[322] D. A. Pearlman; S.-H. Kim. Conformational Studies of Nucleic Acids: III. Empirical Multiple Correlation Functions for Nucleic Acid Torsion Angles. *J. Biomol. Struct. Dyn.*, **1986**, *4*, 49–67.

[323] D. A. Pearlman; S.-H. Kim. Conformational Studies of Nucleic Acids: IV. The Conformational Energetics of Oligonucleotides: d(ApApApA) and ApApApA. *J. Biomol. Struct. Dyn.*, **1986**, *4*, 69–98.

[324] D. A. Pearlman; S.-H. Kim. Conformational Studies of Nueleic Acids. V. Sequence Specificities of in the Conformational Energetics of Oligonucleotides: The Homo-Tetramers. *Biopolymers*, **1988**, *27*, 59–77.

[325] T. Schlick. A Modular Strategy for Generating Starting Conformations and Data Structures of Polynucleotide Helices for Potential Energy Calculations. *J. Comput. Chem.*, **1988**, *9*, 861–889.

[326] J. Gabarro-Arpa; J. A. H. Cognet; M. Le Bret. Object Command Language: a formalism to build molecule models and to analyze structural parameters in macromolecules, with applications to nucleic acids. *J. Mol. Graph.*, **1992**, *10*, 166–173.

[327] R. Lavery. in *Unusual DNA Structures*, R. D. Wells; S. C. Harvey, Eds. Springer-Verlag, New York, 1988.

[328] L. Shen; I. Tinoco. The Structure of an RNA Pseudoknot that Causes Efficient Frameshift in Mouse Mammary Tumor Virus. *J. Mol. Biol.*, **1995**, *247*, 963–978.

[329] J. M. Hubbard; J. E. Hearst. Predicting the Three-Dimensional Folding of Transfer RNA with a Computer Modeling Protocol. *Biochemistry*, **1991**, *30*, 5458–5465.

[330] S.-H. Chou; L. Zhu; B. R. Reid. The Unusual Structure of the Human Centromere (GGA)2 Motif. *J. Mol. Biol.*, **1994**, *244*, 259–268.

[331] M. Levitt. Detailed Molecular Model for Transfer Ribonucleic Acid. *Nature*, **1969**, *224*, 759–763.

[332] M. S. Babcock; E. P. D. Pednault; W. K. Olson. Nucleic Acid Structure Analysis. *J. Mol. Biol.*, **1994**, *237*, 125–156.

[333] J. M. Hubbard; J. E. Hearst. Computer Modeling 16S Ribosomal RNA. *J. Mol. Biol.*, **1991**, *221*, 889–907.

[334] F. Major; M. Turcotte; D. Gautheret; G. Lapalme; E. Fillon; R. Cedergren. The Combination of Symbolic and Numerical Computation for Three-Dimensional Modeling of RNA. *Science*, **1991**, *253*, 1255–1260.

[335] T. Schlick; W. K. Olson. Supercoiled DNA Energetics and Dynamics by Computer Simulation. *J. Mol. Biol.*, **1992**, *223*, 1089–1119.

[336] R. E. Dickerson. Definitions and Nomenclature of Nucleic Acid Structure Parameters. *J. Biomol. Struct. Dyn.*, **1989**, *6*, 627–634.

[337] S. R. Holbrook; J. L. Sussman; R. W. Warrant; S.-H. Kim. Crystal Structure of Yeast Phenylalanine Transfer RNA II. Structural Features and and Functional Implications. *J. Mol. Biol.*, **1978**, *123*, 631–660.

[338] B. N. Conner; C. Yoon; J. Dickerson; R. E. Dickerson. Helix Geometry and Hydration in an A-DNA Tetramer: C-C-G-G. *J. Mol. Biol.*, **1984**, *174*, 663–695.

[339] M. Le Bret; J. Gabarro-Arpa; J. C. Gilbert; C. Lemarechal. MORCAD an object-oriented molecular modeling package. *J. Chim. Phys.*, **1991**, *88*, 2489–2496.

[340] V. B. Zhurkin; Y. P. Lysov; V. I. Ivanov. Different Families of Double Stranded Conformations of DNA as Revealed by Computer Calculations. *Biopolymers*, **1978**, *17*, 277–312.

[341] A. T. Brünger. *X-PLOR: A System for Crystallography and NMR, Version 3.1*. Yale University, New Haven, CT, 1992.

[342] J. R. Wyatt; J. D. Puglisi; I. Tinoco Jr. RNA Pseudoknots. Stability and Loop Size Requirements. *J. Mol. Biol.*, **1990**, *214*, 455–470.

[343] J. D. Puglisi; J. R. Wyatt; I. Tinoco Jr. Conformation of an RNA Pseudoknot. *J. Mol. Biol.*, **1990**, *214*, 437–453.

[344] G. Kuila; J. A. Fee; J. R. Schoonover; W. H. Woodruff. Resonance Raman Spectra of the [2Fe-2S] Clusters of the Rieske Protein from Thermus and Phthalate Dioxygenase from Pseudomonas. *J. Am. Chem. Soc.*, **1987**, *109*, 1559–1561.

[345] B. Lewin. in *Genes IV*, pp 409–425. Cell Press, Cambridge, Mass., 1990.

[346] W. H. Press; S. A. Teukolsky; W. T. Vettering; B. P. Flannery. in *Numerical Recipes in C*, pp 113–117. Cambridge, New York, 1992.

[347] V. B. Zhurkin; G. Raghunathan; N. B. Ulynaov; R. D. Camerini-Otero; R. L. Jernigan. A Parallel DNA Triplex as a Model for the Intermediate in Homologous Recombination. *J. Mol. Biol.*, **1994**, *239*, 181–200.

[348] W. Saenger. in *Principles of Nucleic Acid Structure*, p 120. Springer-Verlag, New York, 1984.

[349] M. Turcotte; G. Lapalme; F. Major. Exploring the conformations of nucleic acids. *J. Funct. Program.*, **1995**, *5*, 443–460.

[350] C.-S. Tung; E. S. Carter, II. Nucleic acid modeling tool (NAMOT): an interactive graphic tool for modeling nucleic acid structures. *CABIOS*, **1994**, *10*, 427–433.

[351] E. S. Carter, II; C.-S. Tung. NAMOT2–a redesigned nucleic acid modeling tool: construction of non-canonical DNA structures. *CABIOS*, **1996**, *12*, 25–30.

[352] R. Lavery; K. Zakrzewska; H. Skelnar. JUMNA (junction minimisation of nucleic acids). *Comp. Phys. Commun.*, **1995**, *91*, 135–158.

[353] G. M. Crippen; T. F. Havel. *Distance Geometry and Molecular Conformation*. Research Studies Press, Taunton, England, 1988.

[354] D. C. Spellmeyer; A. K. Wong; M. J. Bower; J. M. Blaney. Conformational analysis using distance geometry methods. *J. Mol. Graph. Model.*, **1997**, *15*, 18–36.

[355] M. E. Hodsdon; J. W. Ponder; D. P. Cistola. The NMR solution structure of intestinal fatty acid-binding protein complexed with palmitate: Application of a novel distance geometry algorithm. *J. Mol. Biol.*, **1996**, *264*, 585–602.

[356] T. Macke; S.-M. Chen; W. J. Chazin. in *Structure and Function, Volume 1: Nucleic Acids*, R. H. Sarma; M. H. Sarma, Eds., pp 213–227. Adenine Press, Albany, 1992.

[357] B. C. M. Potts; J. Smith; M. Akke; T. J. Macke; K. Okazaki; H. Hidaka; D. A. Case; W. J. Chazin. The structure of calcyclin reveals a novel homodimeric fold S100 Ca^{2+}-binding proteins. *Nature Struct. Biol.*, **1995**, *2*, 790–796.

[358] J. J. Love; X. Li; D. A. Case; K. Giese; R. Grosschedl; P. E. Wright. DNA recognition and bending by the architectural transcription factor LEF-1: NMR structure of the HMG domain complexed with DNA. *Nature*, **1995**, *376*, 791–795.

[359] R. J. Gurbiel; P. E. Doan; G. T. Gassner; T. J. Macke; D. A. Case; T. Ohnishi; J. A. Fee; D. P. Ballou; B. M. Hoffman. Active site structure of Rieske-type proteins: Electron nuclear double resonance studies of isotopically labeled phthalate dioxygenase from *Pseudomonas cepacia and Rieske protein from Rhodobacter capsulatus* and molecular modeling studies of a Rieske center. *Biochemistry*, **1996**, *35*, 7834–7845.

[360] T. J. Macke. *NAB, a Language for Molecular Manipulation*. Ph.D. thesis, The Scripps Research Institute, 1996.

[361] D. Gautheret; F. Major; R. Cedergren. Modeling the three-dimensional structure of RNA using discrete nucleotide conformational sets. *J. Mol. Biol.*, **1993**, *229*, 1049–1064.

[362] R. Tan; S. Harvey. Molecular Mechanics Model of Supercoiled DNA. *J. Mol. Biol.*, **1989**, *205*, 573–591.

[363] T. F. Havel. An evaluation of computational strategies for use in the determination of protein structure from distance constraints obtained by nuclear magnetic resonance. *Prog. Biophys. Mol. Biol.*, **1991**, *56*, 43–78.

[364] J. Kuszewski; M. Nilges; A. T. Brünger. Sampling and efficiency of metric matrix distance geometry: A novel partial metrization algorithm. *J. Biomolec. NMR*, **1992**, *2*, 33–56.

[365] B. L. deGroot; D. M. F. van Aalten; R. M. Scheek; A. Amadei; G. Vriend; H. J. C. Berendsen. Prediction of protein conformational freedom from distance constraints. *Proteins*, **1997**, *29*, 240–251.

[366] T. F. Havel; I. D. Kuntz; G. M. Crippen. The theory and practice of distance geometry. *Bull. Math. Biol.*, **1983**, *45*, 665–720.

[367] D. K. Agrafiotis. Stochastic Proximity Embedding. *J. Comput. Chem.*, **2003**, *24*, 1215–1221.

[368] C. Brooks; A. Brünger; M. Karplus. Active site dynamics in protein molecules: A stochastic boundary molecular-dynamics approach. *Biopolymers*, **1985**, *24*, 843–865.

[369] D. T. Nguyen; D. A. Case. On finding stationary states on large-molecule potential energy surfaces. *J. Phys. Chem.*, **1985**, *89*, 4020–4026.

[370] D. A. Pearlman; D. A. Case; J. W. Caldwell; W. S. Ross; T. E. Cheatham, III; S. DeBolt; D. Ferguson; G. Seibel; P. Kollman. AMBER, a package of computer programs for applying molecular mechanics, normal mode analysis, molecular dynamics and free energy calculations to simulate the structural and energetic properties of molecules. *Comp. Phys. Commun.*, **1995**, *91*, 1–41.

[371] S. Harvey; J. A. McCammon. *Dynamics of Proteins and Nucleic Acids*. Cambridge University Press, Cambridge, 1987.

[372] M. P. Allen; D. J. Tildesley. *Computer Simulation of Liquids*. Clarendon Press, Oxford, 1987.

[373] W. F. van Gunsteren; P. K. Weiner; A. J. Wilkinson, eds. *Computer Simulations of Biomolecular Systems, Vol. 3.* ESCOM Science Publishers, Leiden, 1997.

[374] W. F. van Gunsteren; P. K. Weiner; A. J. Wilkinson, eds. *Computer Simulations of Biomolecular Systems, Vol. 2.* ESCOM Science Publishers, Leiden, 1993.

[375] L. R. Pratt; G. Hummer, eds. *Simulation and Theory of Electrostatic Interactions in Solution*. American Institute of Physics, Melville, NY, 1999.

[376] J. Wang; P. Cieplak; P. A. Kollman. How well does a restrained electrostatic potential (RESP) model perform in calculating conformational energies of organic and biological molecules? *J. Comput. Chem.*, **2000**, *21*, 1049–1074.

[377] R. W. Dixon; P. A. Kollman. Advancing beyond the atom-centered model in additive and nonadditive molecular mechanics. *J. Comput. Chem.*, **1997**, *18*, 1632–1646.

[378] W. D. Cornell; P. Cieplak; C. I. Bayly; I. R. Gould; K. M. Merz, Jr.; D. M. Ferguson; D. C. Spellmeyer; T. Fox; J. W. Caldwell; P. A. Kollman. A second generation force field for the simulation of proteins, nucleic acids, and organic molecules. *J. Am. Chem. Soc.*, **1995**, *117*, 5179–5197.

[379] M. D. Beachy; R. A. Friesner. Accurate ab intio quantum chemical determination of the relative energies of peptide conformations and assessment of empirical force fields. *J. Am. Chem. Soc.*, **1997**, *119*, 5908–5920.

[380] P. A. Kollman; R. Dixon; W. Cornell; T. Fox; C. Chipot; A. Pohorille. in *Computer Simulation of Biomolecular Systems, Vol. 3*, A. Wilkinson; P. Weiner; W. F. van Gunsteren, Eds., pp 83–96. Elsevier, 1997.

[381] J. Higo; N. Ito; M. Kuroda; S. Ono; N. Nakajima; H. Nakamura. Energy landscape of a peptide consisting of α-helix, 3_{10} helix, β-turn, β-hairpin and other disordered conformations. *Prot. Sci.*, **2001**, *10*, 1160–1171.

[382] L. Wang; Y. Duan; R. Shortle; B. Imperiali; P. A. Kollman. Study of the stability and unfolding mechanism of BBA1 by molecular dynamics simulations at different temperatures. *Prot. Sci.*, **1999**, *8*, 1292–1304.

[383] T. E. Cheatham, III; P. Cieplak; P. A. Kollman. A modified version of the Cornell et al. force field with improved sugar pucker phases and helical repeat. *J. Biomol. Struct. Dyn.*, **1999**, *16*, 845–862.

[384] S. J. Weiner; P. A. Kollman; D. A. Case; U. C. Singh; C. Ghio; G. Alagona; S. Profeta, Jr.; P. Weiner. A new force field for molecular mechanical simulation of nucleic acids and proteins. *J. Am. Chem. Soc.*, **1984**, *106*, 765–784.

[385] S. J. Weiner; P. A. Kollman; D. T. Nguyen; D. A. Case. An all-atom force field for simulations of proteins and nucleic acids. *J. Comput. Chem.*, **1986**, *7*, 230–252.

[386] J. Åqvist. Ion-water interaction potentials derived from free energy perturbation simulations. *J. Phys. Chem.*, **1990**, *94*, 8021–8024.

[387] J. Åqvist; A. Warshel. Computer simulation of the initial proton-transfer step in human carbonic anhydrase-I. *J. Mol. Biol.*, **1992**, *224*, 7–14.

[388] W. L. Jorgensen; J. Chandrasekhar; J. Madura; M. L. Klein. Comparison of simple potential functions for simulating liquid water. *J. Chem. Phys.*, **1983**, *79*, 926–935.

[389] W. L. Jorgensen; J. D. Madura. Temperature and size dependence for Monte Carlo simulations of TIP4P water. *Mol. Phys.*, **1985**, *56*, 1381–1392.

[390] M. W. Mahoney; W. L. Jorgensen. A five-site model for liquid water and the reproduction of the density anomaly by rigid, nonpolarizable potential functions. *J. Chem. Phys.*, **2000**, *112*, 8910–8922.

[391] H. J. C. Berendsen; J. R. Grigera; T. P. Straatsma. The missing term in effective pair potentials. *J. Phys. Chem.*, **1987**, *91*, 6269–6271.

[392] H. J. C. Berendsen; J. P. M. Postma; W. F. van Gunsteren; A. DiNola; J. R. Haak. Molecular dynamics with coupling to an external bath. *J. Chem. Phys.*, **1984**, *81*, 3684–3690.

[393] D. S. Wishart; D. A. Case. Use of chemical shifts in macromolecular structure determination. *Meth. Enzymol.*, **2001**, *338*, 3–34.

[394] J. W. Caldwell; P. A. Kollman. Structure and properties of neat liquids using nonadditive molecular dynamics: Water, methanol and N-methylacetamide. *J. Phys. Chem.*, **1995**, *99*, 6208–6219.

[395] B. Honig; A. Nicholls. Classical electrostatics in biology and chemistry. *Science*, **1995**, *268*, 1144–1149.

[396] J. Srinivasan; T. E. Cheatham, III; P. Cieplak; P. Kollman; D. A. Case. Continuum solvent studies of the stability of DNA, RNA, and phosphoramidate–DNA helices. *J. Am. Chem. Soc.*, **1998**, *120*, 9401–9409.

[397] L. T. Chong; Y. Duan; L. Wang; I. Massova; P. A. Kollman. Molecular dynamics and free-energy calculations applied to affinity maturation in antibody 48G7. *Proc. Natl. Acad. Sci. USA*, **1999**, *96*, 14330–14335.

[398] P. A. Kollman; I. Massova; C. Reyes; B. Kuhn; S. Huo; L. Chong; M. Lee; T. Lee; Y. Duan; W. Wang; O. Donini; P. Cieplak; J. Srinivasan; D. A. Case; T. E. Cheatham, III. Calculating structures and free energies of complex molecules: Combining molecular mechanics and continuum models. *Accts. Chem. Res.*, **2000**, *33*, 889–897.

[399] M. L. Connolly. Analytical molecular surface calculation. *J. Appl. Cryst.*, **1983**, *16*, 548–558.

[400] R. Elber; M. Karplus. Enhanced sampling in molecular dynamics. Use of the time-dependent Hartree approximation for a simulation of carbon monoxide diffusion through myoglobin. *J. Am. Chem. Soc.*, **1990**, *112*, 9161–9175.

[401] C. Simmerling; M. R. Lee; A. R. Ortiz; A. Kolinski; J. Skolnick; P. A. Kollman. Combining MONSSTER and LES/PME to Predict Protein Structure from Amino Acid Sequence: Application to the Small Protein CMTI-1. *J. Am. Chem. Soc.*, **2000**, *122*, 8392–8402.

[402] C. Simmerling; R. Elber. Hydrophobic "collapse" in a cyclic hexapeptide: Computer simulations of CHDLFC and CAAAAC in water. *J. Am. Chem. Soc.*, **1994**, *116*, 2534–2547.

[403] W. S. Ross; C. C. Hardin. Ion-induced stabilization of the G-DNA quadruplex: Free energy perturbation studies. *J. Am. Chem. Soc.*, **1994**, *116*, 6070–6080.

[404] A. Vedani; D. W. Huhta. A new force field for modeling metalloproteins. *J. Am. Chem. Soc.*, **1990**, *112*, 4759–4767.

[405] D. L. Veenstra; D. M. Ferguson; P. A. Kollman. How transferable are hydrogen parameters in molecular mechanics calculations? *J. Comput. Chem.*, **1992**, *13*, 971–978.

[406] F. H. Allen; O. Kennard; D. G. Watson; L. Brammer; A. G. Orpen; R. Taylor. *J. Chem. Soc. Perkin Trans. II*, **1987**, pp S1–S19.

[407] M. D. Harmony; R. W. Laurie; R. L. Kuczkowski; R. H. Schwendemann; D. A. Ramsay; F. J. Lovas; W. J. Lafferty; A. G. Maki. *J. Phys. Chem. Ref. Data*, **1979**, *8*, 619.

[408] A. J. Hopfinger; R. A. Pearlstein. Molecular mechanics force-field parameterization procedures. *J. Comput. Chem.*, **1985**, *5*, 486–499.

[409] J. F. Cannon. AMBER force-field parameters for guanosine triphosphate and its imido and methylene analogs. *J. Comput. Chem.*, **1993**, *14*, 995–1005.

[410] P. Cieplak; W. D. Cornell; C. Bayly; P. A. Kollman. Application of the multimolecule and multiconformational RESP methodology to biopolymers: Charge derivation for DNA, RNA and proteins. *J. Comput. Chem.*, **1995**, *16*, 1357–1377.

[411] W. D. Cornell; P. Cieplak; C. I. Bayly; P. A. Kollman. Application of RESP charges to calculate conformational energies, hydrogen bond energies and free energies of solvation. *J. Am. Chem. Soc.*, **1993**, *115*, 9620–9631.

[412] C. I. Bayly; P. Cieplak; W. D. Cornell; P. A. Kollman. A well-Behaved electrostatic potential based method using charge restraints for determining atom-centered charges: The RESP model. *J. Phys. Chem.*, **1993**, *97*, 10269–10280.

[413] A. E. Howard; P. Cieplak; P. A. Kollman. A molecular mechanical model that reproduces the relative energies for chair and twist-boat conformations of 1,3-dioxanes. *J. Comp. Chem.*, **1995**, *16*, 243–261.

[414] A. St.-Amant; W. D. Cornell; P. A. Kollman; T. A. Halgren. Calculation of molecular geometries, relative conformational energies, dipole moments, and molecular electrostatic potential fitted charges of small organic molecules of biochemical interest by density functional theory. *J. Comput. Chem.*, **1995**, *16*, 1483–1506.

[415] T. A. Halgren. Merck Molecular Force Field (MMFF94). Part I-V. *J. Comput. Chem.*, **1996**, *17*, 490–641.

[416] J. Wang; P. Morin; W. Wang; P. A. Kollman. Use of MM-PBSA in reproducing the binding free energies to HIV-1 RT of TIBO derivatives and predicting the binding mode to HIV-1 RT of efavirenz by docking and MM-PBSA. *J. Am. Chem. Soc.*, **2001**, *123*, 5221–5230.

[417] W. Wang; P. Kollman. Free energy calculations on dimer stability of the HIV protease using molecular dynamics and a continuum solvent model. *J. Mol. Biol.*, **2000**, *303*, 567.

[418] C. Reyes; P. Kollman. Structure and thermodynamics of RNA-protein binding: Using molecular dynamics and free energy analyses to calculate the free energies of binding and conformational change. *J. Mol. Biol.*, **2000**, *297*, 1145–1158.

[419] M. R. Lee; Y. Duan; P. A. Kollman. Use of MM-PB/SA in estimating the free energies of proteins: Application to native, intermediates, and unfolded vilin headpiece. *Proteins*, **2000**, *39*, 309–316.

[420] P. Cieplak; J. Caldwell; P. Kollman. Molecular mechanical models for organic and biological systems going beyond the atom centered two body additive approximation: Aqueous solution free energies of methanol and N-methyl acetamide, nucleic acid base, and amide hydrogen bonding and chloroform/water partition coefficients of the nucleic acid bases. *J. Comput. Chem.*, **2001**, *22*, 1048–1057.

[421] E. Meng; P. Cieplak; J. W. Caldwell; P. A. Kollman. Accurate solvation free energies of acetate and methylammonium ions calculated with a polarizable water model. *J. Am. Chem. Soc.*, **1994**, *116*, 12061–12062.

[422] J. Wang; P. A. Kollman. Automatic parameterization of force field by systematic search and genetic algorithms. *J. Comput. Chem.*, **2001**, *22*, 1219–1228.

[423] D. L. Beveridge; F. M. DiCapua. Free energy simulation via molecular simulations: Applications to chemical and biomolecular systems. *Annu. Rev. Biophys. Biophys. Chem.*, **1989**, *18*, 431–492.

[424] C. Chipot; P. A. Kollman; D. A. Pearlman. Alternative approaches to potential of mean force calculations: free energy perturbation versus thermodynamics integration. Case study of some representative nonpolar interactions. *J. Comput. Chem*, **1996**, *17*, 1112–1131.

[425] D. A. Pearlman; P. A. Kollman. The overlooked bond-stretching contribution in free energy perturbation calculations. *J. Chem. Phys.*, **1991**, *94*, 4532–4545.

[426] D. A. Pearlman. Determining the contributions of constraints in free energy calculations: Development, characterization, and recommendations. *J. Chem. Phys.*, **1993**, *98*, 8946–8957.

[427] D. A. Pearlman. Free energy derivatives: A new method for probing the convergence problem in free energy calculations. *J. Comput. Chem.*, **1994**, *15*, 105–123.

[428] D. A. Pearlman. A comparison of alternative approaches to free energy calculations. *J. Phys. Chem.*, **1994**, *98*, 1487–1493.

[429] D. A. Pearlman; B. G. Rao. in *Encyclopedia of Computational Chemistry*, P. von R. Schleyer; N. L. Allinger; T. Clark; J. Gasteiger; P. A. Kollman; I. H. F. Schaefer, Eds., pp 1036–1061. John Wiley, Chichester, 1998.

[430] R. J. Radmer; P. A. Kollman. Free energy calculation methods: A theoretical and empirical comparison of numerical errors and a new method for qualitative estimates of free energy changes. *J. Comput. Chem.*, **1997**, *18*, 902–919.

[431] D. A. Pearlman; P. A. Kollman. A new method for carrying out free energy perturbation calculations: dynamically modified windows. *J. Chem. PHys.*, **1989**, *90*, 2460–2470.

[432] H.-A. Yu; M. Karplus. A thermodynamic analysis of solvation. *J. Chem. Phys.*, **1988**, *89*, 2366–2379.

[433] G. Hummer. Fast-growth thermodynamic integration: Error and efficiency analysis. *J. Chem. Phys.*, **2001**, *114*, 7330–7337.

[434] S. H. Fleischman; C. L. Brooks, III. Thermodynamic calculations on biological systems: Solution properties of alcohols and alkanes. *J. Chem. Phys.*, **1988**, *87*, 221–234.

[435] A. Jakalian; B. L. Bush; D. B. Jack; C. I. Bayly. Fast, efficient generation of high-quality atomic charges. AM1-BCC model: I. Method. *J. Comput. Chem.*, **2000**, *21*, 132–146.

[436] A. Jakalian; D. B. Jack; C. I. Bayly. Fast, efficient generation of high-quality atomic charges. AM1-BCC model: II. Parameterization and Validation. *J. Comput. Chem.*, **2002**, *23*, 1623–1641.

[437] J. J. Vincent; K. M. Merz, Jr. A highly portable parallel implementation of AMBER4 using the message passing interface standard. *J. Comput. Chem.*, **1995**, *16*, 1420–1427.

[438] R. Radmer; P. Kollman. The application of three approximate free energy calculations methods to structure based ligand design: Trypsin and its complex with inhibitors. *J. Comput.-Aided Mol. Design*, **1998**, *12*, 215–228.

[439] S. R. Niketic; K. Rasmussen. *The Consistent Force Field: A Documentation*. Springer-Verlag, New York, 1977.

[440] C. Cerjan; W. H. Miller. On finding transition states. *J. Chem. Phys.*, **1981**, *75*, 2800.

[441] D. T. Nguyen; D. A. Case. On finding stationary states on large-molecule potential energy surfaces. *J. Phys. Chem.*, **1985**, *89*, 4020–4026.

[442] G. Lamm; A. Szabo. Langevin modes of macromolecules. *J. Chem. Phys.*, **1986**, *85*, 7334–7348.

[443] J. Kottalam; D. A. Case. Langevin modes of macromolecules: application to crambin and DNA hexamers. *Biopolymers*, **1990**, *29*, 1409–1421.

[444] S. Huo; I. Massova; P. A. Kollman. Computational Alanine Scanning of the 1:1 Human Growth Hormone-Receptor Complex. *J. Comput. Chem.*, **2002**, *23*, 15–27.

[445] T. Darden; D. Pearlman; L. G. Pedersen. Ionic charging free energies: Spherical versus periodic boundary conditions. *J. Chem. Phys.*, **1998**, *109*, 10921–10935.

[446] R. M. Levy; M. Karplus; J. Kushick; D. Perahia. Evaluation of the configurational entropy for proteins: Application to molecular dynamics simulations of an α-helix. *Macromolecules*, **1984**, *17*, 1370–1374.

[447] E. Gallicchio; M. M. Kubo; R. M. Levy. Enthalpy-entropy and cavity decomposition of alkane hydration free energies: Numerical results and implications for theories of hydrophobic solvation. *J. Phys. Chem.*, **2000**, *104*, 6271–6285.

[448] S. Arnott; P. J. Campbell-Smith; R. Chandrasekaran. in *Handbook of Biochemistry and Molecular Biology, 3rd ed. Nucleic*, G. P. Fasman, Ed., pp 411–422. CRC Press, Cleveland, 1976.

[449] O. Becker; A. D. MacKerell; B. Roux; M. Watanabe, eds. *Computational Biochemistry and Biophysics*. Marcel Dekker, New York, 2001.

[450] A. R. Leach. *Molecular Modelling. Principles and Applications, Second Edition*. Prentice-Hall, Harlow, England, 2001.

[451] T. E. Cheatham, III; B. R. Brooks; P. A. Kollman. in *Current Protocols in Nucleic Acid Chemistry*, pp Sections 7.5, 7.8, 7.9, 7.10. Wiley, New York, 1999.

[452] G. Sigalov; P. Scheffel; A. Onufriev. Incorporating variable environments into the generalized Born model. *J. Chem. Phys.*, **2005**, *122*, 094511.

[453] G. Sigalov; A. Fenley; A. Onufriev. Analytical electrostatics for biomolecules: Beyond the generalized Born approximation . *J. Chem. Phys.*, **2006**, *124*, 124902.

[454] A. Mitsutake; Y. Sugita; Y. Okamoto. Generalized-ensemble algorithms for molecular simulations of biopolymers. *Biopolymers*, **2001**, *60*, 96–123.

[455] J. J. Prompers; R. Brüschweiler. Dynamic and structural analysis of isotropically distributed molecular ensembles. *Proteins*, **2002**, *46*, 177–189.

[456] J. J. Prompers; R. Brüschweiler. General framework for studying the dynamics of folded and nonfolded proteins by NMR relaxation spectroscopy and MD simulation. *J. Am. Chem. Soc.*, **2002**, *124*, 4522–4534.

[457] J. P. Valleau; G. M. Torrie. in *Modern Theoretical Chemistry, Vol. 5: Statistical Mechanics, Part A,*, B. J. Berne, Ed. Plenum Press, New York, 1977.

[458] S. Kumar; D. Bouzida; R. H. Swendsen; P. A. Kollman; J. M. Rosenberg. The weighted histogram analysis method for free-energy calculations on biomolecules. I. The method. *J. Comput. Chem.*, **1992**, *13*, 1011–1021.

[459] S. Kumar; J. M. Rosenberg; D. Bouzida; R. H. Swendsen; P. A. Kollman. Multidimensional free-energy calculations using the weighted histogram analysis method. *J. Comput. Chem.*, **1995**, *16*, 1339–1350.

[460] J. Kottalam; D. A. Case. Dynamics of ligand escape from the heme pocket of myoglobin. *J. Am. Chem. Soc.*, **1988**, *110*, 7690–7697.

[461] B. Roux. The calculation of the potential of mean force using computer simulations. *Comput. Phys. Comm.*, **1995**, *91*, 275–282.

[462] J. W. Ponder; D. A. Case. Force fields for protein simulations. *Adv. Prot. Chem.*, **2003**, *66*, 27–85.

[463] W. H. Press; B. P. Flannery; S. A. Teukolsky; W. T. Vetterling. in *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, New York, 1989.

[464] P. Beroza; D. A. Case. Calculations of proton-binding thermodynamics in proteins. *Meth. Enzymol.*, **1998**, *295*, 170–189.

[465] J. D. Madura; M. E. Davis; M. K. Gilson; R. C. Wade; B. A. Luty; J. A. McCammon. Biological applications of electrostatic calculations and brownian dynamics simulations. *Rev. Computat. Chem.*, **1994**, *5*, 229–267.

[466] M. K. Gilson. Theory of electrostatic interactions in macromolecules. *Curr. Opin. Struct. Biol.*, **1995**, *5*, 216–23.

[467] M. Scarsi; J. Apostolakis; A. Caflisch. Continuum electrostatic energies of macromolecules in aqueous solutions. *J. Phys. Chem. A*, **1997**, *101*, 8098–8106.

[468] R. Luo; L. David; M. K. Gilson. Accelerated Poisson-Boltzmann calculations for static and dynamic systems. *J. Comput. Chem.*, **2002**, *23*, 1244–1253.

[469] H. Wei; A. Luo; T. Qiu; R. Luo; R. Qi. Improved Poisson-Boltzmann Methods for High-Performance Computing. *J. Chem. Theory Comput.*, **2019**, *15*, 6190.

[470] H. Wei; R. Luo; R. Qi. An Efficient Second-Order Poisson-Boltzmann Method. *J. Comput. Chem.*, **2019**, *40*, 1257.

[471] D. Greene; R. Qi; R. Nguyen; T. Qiu; R. Luo. Heterogeneous dielectric implicit membrane model for the calculation of mmpbsa binding free energies. *J. Chem. Infom. Model.*, **2019**, *59*, 3041.

[472] L. Xiao; J. Diao; D. Greene; J. Wang; R. Luo. A Continuum Poisson-Boltzmann Model for Membrane Channel Proteins. *J. Chem. Theory Comput.*, **2017**, *13*, 3398.

[473] C. Wang; P. Ren; R. Luo. Ionic solution: What goes right and wrong with continuum solvation modeling. *J. Phys. Chem. B*, **2017**, *121*, 11169.

[474] E. King; R. Qi; H. Li; R. Luo; E. Aitchison. Estimating the roles of protonation and electronic polarization in absolute binding affinity simulations. *J. Chem. Theory Comput.*, **2021**, *17*, 0000.

[475] Z. Li; I. K. *The Immersed Interface Method: Numerical Solutions of PDEs Involving Interfaces and Irregular Domains*. SIAM Frontiers in Applied Mathematics, Philadelphia, 2006.

[476] J. Wang; Q. Cai; Z. Li; H. Zhao; R. Luo. Achieving Energy Conservation in Poisson-Boltzmann Molecular Dynamics: Accuracy and Precision with Finite-difference Algorithms. *Chem. Phys. Lett.*, **2009**, *468*, 112.

[477] J. Wang; R. Luo. Assessment of Linear Finite-Difference Poisson-Boltzmann solvers. *J. Comput. Chem.*, **2010**, *31*, 1689–1698.

[478] Q. Cai; J. Wang; H. Zhao; R. Luo. On removal of charge singularity in Poisson-Boltzmann equation. *J. Chem. Phys.*, **2009**, *130*, 145101.

[479] Q. Cai; M.-J. Hsieh; J. Wang; R. Luo. Performance of Nonlinear Finite-Difference Poisson-Boltzmann Solvers. *J. Chem. Theory Comput.*, **2010**, *6*, 203.

[480] Q. Cai; X. Ye; J. Wang; R. Luo. Dielectric boundary force in numerical Poisson-Boltzmann methods: Theory and numerical strategies. *Chem. Phys. Lett.*, **2011**, *514*, 368.

[481] Q. Cai; X. Ye; J. Wang; R. Luo. On-the-Fly Numerical Surface Integration for Finite-Difference Poisson-Boltzmann Methods. *J. Chem. Theory Comput.*, **2011**, *7*, 3608–3619.

[482] Q. Cai; X. Ye; R. Luo. Dielectric Pressure in Continuum Electrostatic Solvation of Biomolecules. *Phys. Chem. Chem. Phys.*, **2012**, *14*, 15917–15925.

[483] W. M. Botello-Smith; X. Liu; Q. Cai; Z. Li; H. Zhao; R. Luo. Numerical Poisson-Boltzmann Model for Continuum Membrane Systems. *Chem. Phys. Lett.*, **2013**, *555*, 274.

[484] X. Ye; J. Wang; R. Luo. A Revised Density Function for Molecular Surface Calculation in Continuum Solvent Models. *J. Chem. Theory Comput.*, **2010**, *6*, 1157–1169.

[485] C. Wang; P. Nguyen; K. Pham; D. Huynh; T. Le; H. Wang; P. Ren; R. Luo. Calculating protein-ligand binding affinities with MMPBSA: Method and error analysis. *J. Comput. Chem.*, **2016**, *37*, 2436–2446.

[486] R. Qi; W. Botello-Smith; R. Luo. Acceleration of Linear Finite-Difference Poisson-Boltzmann Methods on Graphics Processing Units. *J. Chem. Theory Comput.*, **2017**, *13*, 3378–3387.

[487] R. Qi; R. Luo. Robustness and Efficiency of Poisson-Boltzmann Modeling on Graphics Processing Units. *J. Chem. Inf. Model.*, **2019**, *59*, 409–420.

[488] T. Simonson. Electrostatics and dynamics of proteins. *Rep. Prog. Phys.*, **2003**, *66*, 737–787.

[489] D. Bashford; M. Karplus. p‘K sub a‘’s of ionizable groups in proteins: Atomic detail from a continuum electrostatic model. *Biochemistry*, **1990**, *29*, 10219–10225.

[490] A. Ghosh; C. S. Rapp; R. A. Friesner. Generalized Born model based on a surface integral formulation. *J. Phys. Chem. B*, **1998**, *102*, 10983–10990.

[491] Y. Duan; C. Wu; S. Chowdhury; M. C. Lee; G. Xiong; W. Zhang; R. Yang; P. Cieplak; R. Luo; T. Lee. A point-charge force field for molecular mechanics simulations of proteins based on condensed-phase quantum mechanical calculations. *J. Comput. Chem.*, **2003**, *24*, 1999–2012.

[492] J. D. Jackson. *Classical Electrodynamics*. Wiley and Sons, New York, 1975.

[493] Q. Lu; R. Luo. A Poisson-Boltzmann dynamics method with nonperiodic boundary condition. *J. Chem. Phys.*, **2003**, *119*, 11035–11047.

[494] C. H. Tan; L. J. Yang; R. Luo. How well does Poisson-Boltzmann implicit solvent agree with explicit solvent? A quantitative analysis. *J. Phys. Chem. B*, **2006**, *110*, 18680–18687.

[495] C. H. Tan; Y. H. Tan; R. Luo. Implicit nonpolar solvent models. *J. Phys. Chem. B*, **2007**, *111*, 12263–12274.

[496] M. Feig; J. Karanicolas; C. L. Brooks, III. MMTSB Tool Set: Enhanced sampling and multiscale modeling methods for application in structural biology. *J. Mol. Graphics Mod.*, **2004**, *22*, 377–395.

[497] C. Simmerling; B. Strockbine; A. E. Roitberg. All-atom structure prediction and folding simulations of a stable protein. *J. Am. Chem. Soc.*, **2002**, *124*, 11258–11259.

[498] A. E. García; K. Y. Sanbonmatsu. α-helical stabilization by side chain shielding of backbone hydrogen bonds. *Proc. Natl. Acad. Sci. USA*, **2002**, *99*, 2782–2787.

[499] J. Wang; R. M. Wolf; J. W. Caldwell; P. A. Kollamn; D. A. Case. Development and testing of a general Amber force field. *J. Comput. Chem.*, **2004**, *25*, 1157–1174.

[500] K. N. Kirschner; R. J. Woods. Solvent interactions determine carbohydrate conformation. *Proc. Natl. Acad. Sci. USA*, **2001**, *98*, 10541–10545.

[501] M. Basma; S. Sundara; D. Calgan; T. Venali; R. J. Woods. Solvated ensemble averaging in the calculation of partial atomic charges. *J. Comput. Chem.*, **2001**, *22*, 1125–1137.

[502] K. N. Kirschner; R. J. Woods. Quantum mechanical study of the nonbonded forces in water-methanol complexes. *J. Phys. Chem. A*, **2001**, *105*, 4150–4155.

[503] K. A. Sharp; B. Honig. Electrostatic interactions in macromolecules: Theory and experiment. *Annu. Rev. Biophys. Biophys. Chem.*, **1990**, *19*, 301–332.

[504] M. K. Gilson; K. A. Sharp; B. H. Honig. Calculating the electrostatic potential of molecules in solution: method. *J. Comput. Chem.*, **1988**, *9*, 327–35.

[505] J. Warwicker; H. C. Watson. Calculation of the electric potential in the active site cleft due to. *J. Mol. Biol.*, **1982**, *157*, 671–679.

[506] I. Klapper; R. Hagstrom; R. Fine; K. Sharp; B. Honig. Focussing of electric fields in the active stie of Cu, Zn superoxide dismutase. *Proteins*, **1986**, *1*, 47–59.

[507] A. Nicholls; B. Honig. A rapid finite difference algorithm, utilizing successive over-relaxation to solve the Poisson-Boltzmann equation. *J. Comput. Chem.*, **1991**, *12*, 435–445.

[508] M. E. Davis; J. A. McCammon. Dielectric boundary smoothing in finite difference solutions of the Poisson equation: An approach to improve accuracy and convergence. *J. Comput. Chem.*, **1991**, *12*, 909–912.

[509] M. E. Davis; J. A. McCammon. Electrostatics in biomolecular structure and dynamics. *Chem. Rev.*, **1990**, *90*, 509–521.

[510] M. E. Davis; J. A. McCammon. Solving the finite-difference linearized Poisson-Boltzmann equation – a comparison of relaxation and conjugate gradient methods. *J. Comput. Chem.*, **1989**, *10*, 386–391.

[511] D. Bashford. An object-oriented programming suite for electrostatic effects in biological molecules. *Lect. Notes Comput. Sci.*, **1997**, *1343*, 233–240.

[512] B. A. Luty; M. E. Davis; J. A. McCammon. Electrostatic energy calculations by a finite-difference method: Rapid calculation of charge-solvent interaction energies. *J. Comput. Chem.*, **1992**, *13*, 768–771.

[513] U. C. Singh; S. J. Weiner; P. A. Kollman. Molecular dynamics simulations of d(C-G-C-G-A).d(T-C-G-C-G) with and without "hydrated" counterions. *Proc. Nat. Acad. Sci.*, **1985**, *82*, 755–759.

[514] J. Gao. Absolute free energy of solvation from Monte Carlo simulations using combined quantum and molecular mechanical potentials. *J. Phys. Chem.*, **1992**, *96*, 537–540.

[515] A. Warshel; M. Levitt. Theoretical studies of enzymic reactions: Dielectric, electrostatic and steric stabilization of the carbonium ion in the reaction of lysozyme. *J. Mol. Biol.*, **1976**, *103*, 227–249.

[516] M. J. Field; P. A. Bash; M. Karplus. A combined quantum mechanical and molecular mechanical potential for molecular dynamics simulations. *J. Comput. Chem.*, **1990**, *11*, 700–733.

[517] R. V. Stanton; D. S. Hartsough; K. M. Merz, Jr. An examination of a density functional/molecular mechanical coupled potential. *J. Comput. Chem.*, **1994**, *16*, 113–128.

[518] R. V. Stanton; L. R. Little; K. M. Merz, Jr. An examination of a Hartree-Fock/molecular mechanical coupled potential. *J. Phys. Chem.*, **1995**, *99*, 17344–17348.

[519] R. V. Stanton; D. S. Hartsough; K. M. Merz, Jr. Calculations of solvation free energies using a density functional/molecular dynamics coupled potential. *J. Phys. Chem.*, **1993**, *97*, 11868–11870.

[520] W. Yang; T.-S. Lee. A density-matrix divide-and-conquer approach for electronic structure calculations of large molecules. *J. Chem. Phys.*, **1995**, *103*, 5674–5678.

[521] S. L. Dixon; K. M. Merz, Jr. Semiempirical molecular orbital calculations with linear system size scaling. *J. Chem. Phys.*, **1996**, *104*, 6643–6649.

[522] S. L. Dixon; K. M. Merz, Jr. Fast, accurate semiempirical molecular orbital calculations for macromolecules. *J. Chem. Phys.*, **1997**, *107*, 879–893.

[523] J. Nocedal; S. J. Wright. *Numerical Optimization*. Springer-Verlag, New York, 1999.

[524] S. G. Nash. A survey of truncated-Newton methods. *J. of Computational and Applied Mathematics*, **2000**, *124*, 45–59.

[525] M. C. Lee; Y. Duan. Distinguish protein decoys by using a scoring function based on a new Amber force field, short molecular dynamics simulations, and the generalized Born solvent model. *Proteins*, **2004**, *55*, 620–634.

[526] J. Chu; B. L. Trout; B. R. Brooks. A super-linear minimization scheme for the nudged elastic band method. *J. Chem. Phys.*, **2003**, *119*, 12708–12717.

[527] R. Elber; M. Karplus M. A method for determining reaction paths in large molecules: Application to myoglobin. *Chem. Phys. Lett.*, **1987**, *139*, 375–380.

[528] G. Henkelman; H. Jönsson. Improved tangent estimate in the nudged elastic band method for finding minimum energy paths and saddle points. *J. Chem. Phys.*, **2000**, *113*, 9978–9985.

[529] G. Henkelman; B. P. Uberuaga; H. Jönsson. A climbing image nudged elastic band method for finding saddle points and minimum energy paths. *J. Chem. Phys.*, **2000**, *113*, 9901–9904.

[530] H. Jönsson; G. Mills; K. W. Jacobsen. in *Classical and Quantum Dynamics in Condensed Phase Simulations*, B. J. Berne; G. Ciccoti; D. F. Coker, Eds., pp 385–404. World Scientific, Singapore, 1998.

[531] G. Mills; H. Jönsson. Quantum and thermal effects in H_2 dissociative adsorption: Evaluation of free energy barriers in multidimensional quantum systems. *Phys. Rev. Lett.*, **1994**, *72*, 1124–1127.

[532] J. Mongan; D. A. Case; J. A. McCammon. Constant pH molecular dynamics in generalized Born implicit solvent. *J. Comput. Chem.*, **2004**, *25*, 2038–2048.

[533] D. A. Case; T. Cheatham; T. Darden; H. Gohlke; R. Luo; K. M. Merz, Jr.; A. Onufriev; C. Simmerling; B. Wang; R. Woods. The Amber biomolecular simulation programs. *J. Computat. Chem.*, **2005**, *26*, 1668–1688.

[534] E. J. Sorin; V. S. Pande. Exploring the helix-coil transition via all-atom equilibrium ensemble simulations. *Biophys. J.*, **2005**, *88*, 2472–2493.

[535] L. Yang; C. Tan; M.-J. Hsieh; J. Wang; Y. Duan; P. Cieplak; J. Caldwell; P. A. Kollman; R. Luo. New-generation Amber united-atom force field. *J. Phys. Chem. B*, **2006**, *110*, 13166–13176.

[536] B. Wang; K. M. Merz, Jr. A fast QM/MM (quantum mechanical/molecular mechanical) approach to calculate nuclear magnetic resonance chemical shifts for macromolecules. *J. Chem. Theory Comput.*, **2006**, *2*, 209–215.

[537] R. C. Rizzo; T. Aynechi; D. A. Case; I. D. Kuntz. Estimation of absolute free energies of hydration using continuum methods: Accuracy of partial charge models and optimization of nonpolar contributions. *J. Chem. Theory Comput.*, **2006**, *2*, 128–139.

[538] Z.-X. Wang; W. Zhang; C. Wu; H. Lei; P. Cieplak; Y. Duan. Strike a Balance: Optimization of backbone torsion parameters of AMBER polarizable force field for simulations of proteins and peptides. *J. Comput. Chem.*, **2006**, *27*, 781–790.

[539] R. P. Feynman; A. R. Hibbs. *Quantum Mechanics and Path Integrals*. McGraw-Hill, New York, 1965.

[540] R. P. Feynman. *Statistcal Mechanics*. Benjamin, Reading, MA, 1972.

[541] H. Kleinert. *Path Integrals in Quantum Mechanics, Statistics, and Polymer Physics*. World Scientific, Singapore, 1995.

[542] L. S. Schulman. *Techniques and Applications of Path Integration*. Wiley & Sons, New York, 1996.

[543] A. Messiah. *Quantum Mechanics*. Wiley & Sons, New York, 1958.

[544] D. Chandler; P. G. Wolynes. Exploiting the isomorphism between quantum theory and classical statistical mechanics of polyatomic fluids. *J. Chem. Phys.*, **1981**, *74*, 4078–4095.

[545] D. M. Ceperley. Path integrals in the theory of condensed helium. *Rev. Mod. Phys.*, **1995**, *67*, 279–355.

[546] J. Cao; B. J. Berne. On energy estimators in path integral Monte Carlo simulations: Dependence of accuracy on algorithm. *J. Chem. Phys.*, **1989**, *91*, 6359–6366.

[547] J. W. Storer; D. J. Giesen; C. J. Cramer; D. G. Truhlar. Class IV charge models: A new semiempirical approach in quantum chemistry. *J. Comput.-Aided Mol. Design*, **1995**, *9*, 87–110.

[548] J. Li; C. J. Cramer; D. G. Truhlar. New class IV charge model for extracting accurate partial charges from Wave Functions. *J. Phys. Chem. A.*, **1998**, *102*, 1820–1831.

[549] A. van der Vaart; K. M. Merz, Jr. Divide and conquer interaction energy decomposition. *J. Phys. Chem. A*, **1999**, *103*, 3321–3329.

[550] A. V. Mitin. The dynamic level shift method for improving the convergence of the SCF procedure. *J. Comput. Chem.*, **1988**, *9*, 107–110.

[551] M. D. Ermolaeva; A. van der Vaart; K. M. Merz, Jr. Implementation and testing of a frozen density matrix - divide and conquer algorithm. *J. Phys. Chem.*, **1999**, *103*, 1868–1875.

[552] B. Wang; E. N. Brothers; A. van der Vaart; K. M. Merz Jr. Fast semiempirical calculations for nuclear magnetic resonance chemical shifts: A divide-and-conquer approach. *J. Chem. Phys.*, **2004**, *120*, 11392–11400.

[553] B. Wang; K. Raha; K. M. Merz Jr. Pose scoring by NMR. *J. Am. Chem. Soc.*, **2004**, *126*, 11430–11431.

[554] K. Raha; A. van der Vaart; K. E. Riley; M. B. Peters; L. M. Westerhoff; H. Kim; K. M. Merz Jr. Pairwise decomposition of residue interaction energies using semiempirical quantum mechanical methods in studies of protein-ligand interaction. *J. Am. Chem. Soc.*, **2005**, *127*, 6583–6594.

[555] A. Luzhkov; A. Warshel. Microscopic models for quantum-mechanical calculations of chemical processes in solutions - Ld/Ampac and Scaas/Ampac calculations of solvation energies. *J. Comp. Chem.*, **1992**, *13*, 199–213.

[556] U. C. Singh; P. A. Kollman. A combined Ab initio quantum-mechanical and molecular mechanical method for carrying out simulations on complex molecular systems - Applications to the Ch3Cl + Cl- exchange-reaction and gas-phase protonation of polyethers. *J. Comp. Chem.*, **1986**, *7*, 718–730.

[557] I. B. Bersuker; M. K. Leong; J. E. Boggs; R. S. Pearlman. A method of combined quantum mechanical (QM) molecular mechanics (MM) treatment of large polyatomic systems with charge transfer between the QM and MM fragments. *Int. J. Quant. Chem.*, **1997**, *63*, 1051–1063.

[558] F. Maseras; K. Morokuma. Imomm - a new integrated ab-initio plus molecular geometry optimization scheme of equilibrium structures and transition-states. *J. Comp. Chem.*, **1995**, *16*, 1170–1179.

[559] Y. K. Zhang; T. S. Lee; W. T. Yang. A pseudobond approach to combining quantum mechanical and molecular mechanical methods. *J. Chem. Phys.*, **1999**, *110*, 46–54.

[560] J. L. Gao; P. Amara; C. Alhambra; M. J. Field. A generalized hybrid orbital (GHO) method for the treatment of boundary atoms in combined QM/MM calculations. *J Phys Chem A*, **1998**, *102*, 4714–4721.

[561] D. M. Philipp; R. A. Friesner. Mixed ab initio QM/MM modeling using frozen orbitals and tests with alanine dipeptide and tetrapeptide. *J. Comp. Chem.*, **1999**, *20*, 1468–1494.

[562] M. J. Field; M. Albe; C. Bret; F. Proust-De Martin; A. Thomas. The Dynamo library for molecular simulations using hybrid quantum mechanical and molecular mechanical potentials. *J. Comp. Chem.*, **2000**, *21*, 1088–1100.

[563] V. Hornak; R. Abel; A. Okur; B. Strockbine; A. Roitberg; C. Simmerling. Comparison of multiple Amber force fields and development of improved protein backbone parameters. *Proteins*, **2006**, *65*, 712–725.

[564] F. Floris; J. Tomasi. Evaluation of the dispersion contribution to the solvation energy. A simple computational model in the continuum approximation. *J. Comput. Chem.*, **1989**, *10*, 616–627.

[565] R. M. Levy; E. Gallicchio. Computer simulations with explicit solvent: recent progress in the thermodynamic decomposition of free energies and in modeling electrostatic effects. *Annu. Rev. Phys. Chem.*, **1999**, *49*, 531–567.

[566] H. Nymeyer; S. Gnanakaran; A. García. Atomic simulations of protein folding using the replica exchange algorithm. *Meth. Enzymol.*, **2004**, *383*, 119–149.

[567] D. H. Mathews; D. A. Case. Nudged Elastic Band calculation of minimal energy pathways for the conformational change of a GG mismatch. *J. Mol. Biol.*, **2006**, *357*, 1683–1693.

[568] X. Cheng; G. Cui; V. Hornak; C. Simmerling. Modified replica exchange simulation methods for local structure refinement. *J. Phys. Chem. B*, **2005**, *109*, 8220–8230.

[569] J. Wang; W. Wang; P. A. Kollman; D. A. Case. Automatic atom type and bond type perception in molecular mechanical. *J. Mol. Graphics Model.*, **2006**, *25*, 247–260.

[570] V. Hornak; A. Okur; R. Rizzo; C. Simmerling. HIV-1 protease flaps spontaneously open and reclose in molecular dynamics simulations. *Proc. Nat. Acad. Sci. USA*, **2006**, *103*, 915–920.

[571] V. Hornak; A. Okur; R. Rizzo; C. Simmerling. HIV-1 protease flaps spontaneously close when an inhibitor binds to the open state. *J. Am. Chem. Soc.*, **2006**, *128*, 2812–2813.

[572] C. Jarzynski. Nonequilibrium equality for free energy differences. *Phys. Rev. Lett.*, **1997**, *78*, 2690–2693.

[573] G. Hummer; A. Szabo. Free energy reconstruction from nonequilibrium single-molecule pulling experiments. *Proc. Natl. Acad. Sci. USA*, **2001**, *98*, 3658.

[574] G. Hummer; A. Szabo. Kinetics from nonequilibrium single-molecule pulling experiments. *Biophys. J.*, **2003**, *85*, 5–15.

[575] M. O. Jensen; S. Park; E. d; K. Schulten. Energetics of glycerol conduction through aquaglyceroporin GlpF. *Proc. Natl. Acad. Sci. USA*, **2002**, *99*, 6731–6736.

[576] A. Crespo; M. A. Marti; D. A. Estrin; A. E. Roitberg. Multiple-steering QM-MM calculation of the free energy profile in chorismate mutase. *J. Am. Chem. Soc.*, **2005**, *127*, 6940–6941.

[577] P. Y. Ren; J. W. Ponder. Polarizable atomic multipole water model for molecular mechanics simulation. *J. Phys. Chem. B*, **2003**, *107*, 5933–5947.

[578] P. Y. Ren; J. W. Ponder. Tinker polarizable atomic multipole force field for proteins. *to be published.*, **2006**.

[579] J. Kästner; W. Thiel. Bridging the gap between thermodynamic integration and umbrella sampling provides a novel analysis method: "Umbrella integration". *J. Chem. Phys.*, **2005**, *123*, 144104.

[580] A. M. Wollacott; K. M. Merz, Jr. Development of a parameterized force field to reproduce semiempirical geometries. *J. Chem. Theory Comput.*, **2006**, *2*, 1070–1077.

[581] A. Warshel. *Computer Modeling of Chemical Reactions in Enzymes and Solutions*. John Wiley and Sons, New York, 1991.

[582] S. R. Billeter; S. P. Webb; T. Iordanov; P. K. Agarwal; S. Hammes-Schiffer. Hybrid approach for including electronic and nuclear quantum effects in molecular dynamics simulations of hydrogen transfer reactions in enzymes. *J. Chem. Phys.*, **2001**, *114*, 6925.

[583] C. Simmerling; R. Elber. Hydrophobic "collapse" in a cyclic hexapeptide: Computer simulations of CHDLFC and CAAAAC in water. *J. Am. Chem. Soc.*, **1994**, *116*, 2534–2547.

[584] W. Kabsch; C. Sander. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, **1983**, *22*, 2577–2637.

[585] T. E. Cheatham, III; M. A. Young. Molecular dynamics simulation of nucleic acids: Successes, limitations and promise. *Biopolymers*, **2001**, *56*, 232–256.

[586] Y. Deng; B. Roux. Calculation of standard binding free energies: Aromatic molecules in the T4 lysozyme L99A mutant. *J. Chem. Theor. Comput.*, **2006**, *2*, 1255–1273.

[587] L. Marinelli; S. Cosconati; T. Steinbrecher; V. Limongelli; A. Bertamino; E. Novellino; D. A. Case. Homology Modeling of NR2B Modulatory Domain of NMDA Receptor and Analysis of Ifenprodil Binding. *ChemMedChem*, **2007**, *2,*, 1498–1510.

[588] K. N. Kirschner; A. B. Yongye; S. M. Tschampel; J. González-Outeiriño; C. R. Daniels; B. L. Foley; R. J. Woods. GLYCAM06: A generalizable biomolecular force field. Carbohydrates. *J. Comput. Chem.*, **2008**, *29*, 622–655.

[589] S. M. Tschampel; M. R. Kennerty; R. J. Woods. TIP5P-consistent treatment of electrostatics for biomolecular simulations. *J. Chem. Theory Comput.*, **2007**, *3*, 1721–1733.

[590] M. B. Tessier; M. L. DeMarco; A. B. Yongye; R. J. Woods. Extension of the GLYCAM06 biomolecular force field to lipids, lipid bilayers and glycolipids. *Mol. Simul.*, **2008**, *34*, 349–363.

[591] F. Paesani; W. Zhang; D. A. Case; T. E. Cheatham; G. A. Voth. An accurate and simple quantum model for liquid water. *J. Chem. Phys.*, **2006**, *125*, 184507.

[592] A. Okur; D. R. Roe; G. Cui; V. Hornak; C. Simmerling. Improving convergence of replica-exchange simulations through coupling to a high-temperature structure reservoir. *J. Chem. Theory comput.*, **2007**, *3*, 557–568.

[593] A. E. Roitberg; A. Okur; C. Simmerling. Coupling of replica exchange simulations to a non-Boltzmann structure reservoir. *J. Phys. Chem. B*, **2007**, *111*, 2415–2418.

[594] H. B. Schlegel; J. L. Sonnenberg. Empirical valence-bond models for reactive potential energy surfaces using distributed Gaussians. *J. Chem. Theory Comput.*, **2006**, *2*, 905.

[595] J. L. Sonnenberg; H. B. Schlegel. Empirical valence bond models for reactive potential energy surfaces. II. Intramolecular proton transfer in pyridone and the Claisen reaction of allyl vinyl ether. *Mol. Phys.*, **2007**, *105*, 2719.

[596] Y. Saad; M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, **1986**, *7*, 856.

[597] P. Pulay. Convergence acceleration of iterative sequencies. The case of SCF iteration. *Chem. Phys. Lett.*, **1980**, *73*, 393.

[598] P. Pulay. Improved SCF convergence acceleration. *J. Comput. Chem.*, **1982**, *3*, 556.

[599] A. K. Rappe; C. J. Casewit; K. S. Colwell; W. A. Goddard III; W. M. Skiff. UFF, a Full Periodic Table Force Field for Molecular Mechanics and Molecular Dynamics Simulations. *J. Am. Chem. Soc.*, **1992**, *114*, 10024–10035.

[600] G. A. Voth; D. Chandler; W. H. Miller. Rigorous Formulation of Quantum Transition State Theory and Its Dynamical Corrections. *J. Chem. Phys.*, **1989**, *91*, 7749–7760.

[601] G. J. Martyna; M. L. Klein; M. Tuckerman. Nosé-Hoover chains: The canonical ensemble via continuous dynamics. *J. Chem. Phys.*, **1992**, *97*, 2635.

[602] G. J. Martyna; A. Hughes; M. E. Tuckerman. Molecular dynamics algorithms for path integrals at constant pressure. *J. Chem. Phys.*, **1999**, *110*, 3275.

[603] B. J. Berne; D. Thirumalai. On the simulation of quantum systems: path integral methods. *Annu. Rev. Phys. Chem.*, **1986**, *37*, 401.

[604] G. A. Voth. Path-integral centroid methods in quantum statistcal mechanics and dynamics. *Adv. Chem. Phys.*, **1996**, *93*, 135.

[605] I. R. Craig; D. E. Manolopoulos. Quantum statistics and classical mechnanics: Real time correlation functions from ring polymer molecular dynamics. *J. Chem. Phys.*, **2004**, *121*, 3368.

[606] T. F. Miller; D. E. Manolopoulos. Quantum diffusion in liquid water from ring polymer molecular dynamics. *J. Chem. Phys.*, **2005**, *123*, 154504.

[607] J. Cao; G. A. Voth. The formulation of quantum statistical mechanics based on the Feynman path centroid density. IV. Algorithms for centroid molecular dynamics. *J. Chem. Phys.*, **1994**, *101*, 6168.

[608] J. Vaníček; W. H. Miller; J. F. Castillo; F. J. Aoiz. Quantum-instanton evalution of the kinetic isotope effects. *J. Chem. Phys.*, **2005**, *123*, 054108.

[609] J. Vaníček; W. H. Miller. Efficient estimators for quantum instanton evaluation of the kinetic isotope effects: application to the intramolecular hydrogen transfer in pentadiene. *J. Chem. Phys.*, **2007**, *127*, 114309.

[610] W. H. Miller; Y. Zhao; M. Ceotto; S. Yang. Quantum instanton approximation for thermal rate constants of chemical. *J. Chem. Phys.*, **2003**, *119*, 1329–1342.

[611] W. H. Miller. Semiclassical limit of quantum mechanical transition state theory for nonseparable systems. *J. Chem. Phys.*, **1975**, *62*, 1899.

[612] T. Yamamoto; W. H. Miller. On the efficient path integral evaluation of thermal rate constants with the quantum instanton approximation. *J. Chem. Phys.*, **2004**, *120*, 3086–3099.

[613] T. Yamamoto; W. H. Miller. Path integral evaluation of the quantum instanton rate constant for proton transfer in a polar solvent. *J. Chem. Phys.*, **2005**, *122*, 044106.

[614] W. H. Miller; S. D. Schwartz; J. W. Tromp. Quantum mechanical rate constants for bimolecular reactions. *J. Chem. Phys.*, **1983**, *79*, 4889–4898.

[615] A. T. Brünger; P. D. Adams; G. M. Clore; W. L. Delano; P. Gros; R. W. Grosse-Kunstleve; J.-S. Jiang; J. Kuszewski; M. Nilges; N. S. Pannu; R. J. Read; L. M. Rice; T. Simonson; G. L. Warren. Crystallography and NMR system (CNS): A new software system for macromolecular structure determination. *Acta Cryst. D*, **1998**, *54*, 905–921.

[616] N. Yu; H. P. Yennawar; K. M. Merz, Jr. Refinement of protein crystal structures using energy restraints derived from linear-scaling quantum mechanics . *Acta Cryst. D*, **2005**, *61*, 322–332.

[617] N. Yu; X. Li; G. Cui; S. Hayik; K. M. Merz, Jr. Critical assessment of quantum mechanics based energy restraints in protein crystal structure refinement. *Prot. Sci.*, **2006**, *15*, 2773–2784.

[618] S. Fulle; H. Gohlke. Analyzing the flexibility of RNA structures by constraint counting. *Biophys. J.*, **2008**, *DOI:10.1529/biophysj.107.113415*.

[619] H. Gohlke; L. A. Kuhn; D. A. Case. Change in protein flexibility upon complex formation: Analysis of Ras-Raf using molecular dynamics and a molecular framework approach. *Proteins*, **2004**, *56*, 322–327.

[620] A. Ahmed; H. Gohlke. Multiscale modeling of macromolecular conformational changes combining concepts from rigidity and elastic network theory. *Proteins*, **2006**, *63*, 1038–1051.

[621] Y. Wu; H. L. Tepper; G. A. Voth. Flexible simple point-charge water model with improved liquid-state properties. *J. Chem. Phys.*, **2006**, *124*, 024503.

[622] D. J. Price; C. L. Brooks. A modified TIP3P water potential for simulation with Ewald summation. *J. Chem. Phys.*, **2004**, *121*, 10096–10103.

[623] H. W. Horn; W. C. Swope; J. W. Pitera; J. D. Madura; T. J. Dick; G. L. Hura; T. Head-Gordon. Development of an improved four-site water model for biomolecular simulations: TIP4P-Ew. *J. Chem. Phys.*, **2004**, *120*, 9665–9678.

[624] H. W. Horn; W. C. Swope; J. W. Pitera. Characterization of the TIP4P-Ew water model: Vapor pressure and boiling point. *J. Chem. Phys.*, **2005**, *123*, 194504.

[625] R. J. Woods. Derivation of net atomic charges from molecular electrostatic potentials. *J. Comput. Chem.*, **1990**, *11*, 29–310.

[626] M. L. DeMarco; R. J. Woods. Bridging computational biology and glycobiology: A game of snakes and ladders. *Glycobiology*, **2008**, *18*, 426–440.

[627] R. J. Woods. Restrained electrostatic potential charges for condensed phase simulations of carbohydrates. *J. Mol. Struct (Theochem)*, **2000**, *527*, 149–156.

[628] E. F. Pettersen; T. D. Goddard; C. C. Huang; G. S. Couch; D. M. Greenblatt; E. C. Meng; T. E. Ferrin. UCSF Chimera - A visualization system for exploratory research and analysis. *J. Comput. Chem.*, **2004**, *25*, 1605–1612.

[629] H. Sasaki; N. Ochi; A. Del; M. Fukuda. Site-specific glycosylation of human recombinant erythropoietin: Analysis of glycopeptides or peptides at each glycosylation site by fast atom bombardment mass spectrometry. *Biochemistry*, **1988**, *27*, 8618–8626.

[630] S. Dube; J. W. Fisher; J. S. Powell. Glycosylation at specific sites of erythropoietin is essential for biosynthesis, secretion, and biological function. *J. Biol. Chem.*, **1988**, *263*, 17516–17521.

[631] R. J. Darling; U. Kuchibhotla; W. Glaesner; R. Micanovic; D. R. Witcher; J. M. Beals. Glycosylation of erythropoietin effects receptor binding kinetics: Role of electrostatic interactions. *Biochemistry*, **2002**, *41*, 14524–14531.

[632] J. C. Cheetham; D. M. Smith; K. H. Aoki; J. L. Stevenson; T. J. Hoeffel; R. S. Syed; J. Egrie; T. S. Harvey. NMR structure of human erythropoietin and a comparison with its receptor bound conformation. *Nat. Struct. Biol.*, **1998**, *5*, 861–866.

[633] K. L. Dormann; R. Brueckner. Variable Synthesis of the Optically Active Thiotetronic Acid Antibiotics Thiolactomycin, Thiotetromycin, and 834-B1. *Angew. Chem. Int. Ed.*, **2007**, *46*, 1160–1163.

[634] B. Jojart; T. A. Martinek. Performance of the general amber force field in modeling aqueous POPC membrane bilayers. *J. Comput. Chem.*, **2007**, *28*, 2051–2058.

[635] L. Rosso; I. R. Gould. Structure and dynamics of phospholipid bilayers using recently developed general all-atom force fields. *J. Comput. Chem.*, **2008**, *29*, 24–37.

[636] A. P. Graves; D. M. Shivakumar; S. E. Boyce; M. P. Jacobson; D. A. Case; B. K. Shoichet. Rescoring docking hit lists for model cavity sites: Predictions and experimental testing. *J. Mol. Biol.*, **2008**, *377*, 914–934.

[637] E. Darve; A. Pohorille. Calculating free energies using average force. *J. Chem. Phys.*, **2001**, *115*, 9169–9183.

[638] A. Perez; I. Marchan; D. Svozil; J. Sponer; T. E. Cheatham; C. A. Laughton; M. Orozco. Refinement of the AMBER Force Field for Nucleic Acids: Improving the Description of alpha/gamma Conformers. *Biophys. J.*, **2007**, *92*, 3817–3829.

[639] L. Dang. Mechanism and thermodynamics of ion selectivity in aqueous solutions of 18-crown-6 ether: A molecular dynamics study. *J. Am. Chem. Soc.*, **1995**, *117*, 6954–6960.

[640] S. Joung; T. E. Cheatham, III. Determination of alkali and halide monovalent ion parameters for use in explicitly solvated biomolecular simulations. *J. Phys. Chem. B*, **2008**, *112*, 9020–9041.

[641] R. Aduri; B. T. Psciuk; P. Saro; H. Taniga; H. B. Schlegel; J. SantaLucia, Jr. AMBER force field parameters for the naturally occurring modified nucleosides in RNA. *J. Chem. Theory Comput.*, **2007**, *3*, 1465–1475.

[642] J. Shao; S. W. Tanner; N. Thompson; T. E. Cheatham, III. Clustering molecular dynamics trajectories: 1. Characterizing the performance of different clustering algorithms. *J. Chem. Theory Comput.*, **2007**, *3*, 2312–2334.

[643] P. Auffinger; T. E. Cheatham, III; A. C. Vaiana. Spontaneous formation of KCl aggregates in biomolecular simulations: a force field issue? *J. Chem. Theory Comput.*, **2007**, *3*, 1851–1859.

[644] P. Cieplak; F.-Y. Dupradeau; Y. Duan; J. Wang. Polarization effects in molecular mechanical force fields. *J. Phys.: Condens. Matter*, **2009**, *21*, 333102.

[645] Z. J. Shi; J. Shen. New inexact line search method for unconstrained optimization. *J. Optim. Theory Appl.*, **2005**, *127*, 425–446.

[646] A. D. MacKerell Jr.; D. Bashford; M. Bellott; R. L. Dunbrack; J. D. Evanseck; M. J. Field; S. Fischer; J. Gao; H. Guo; S. Ha; D. Joseph-McCarthy; L. Kuchnir; K. Kuczera; F. T. K. Lau; C. Mattos; S. Michnick; T. Ngo; D. T. Nguyen; B. Prodhom; W. E. Reiher; B. Roux; M. Schlenkrich; J. C. Smith; R. Stote; J. Straub; M. Watanabe; J. Wiorkiewicz-Kuczera; D. Yin; M. Karplus. All-Atom Empirical Potential for Molecular Modeling and Dynamics Studies of Proteins. *J. Phys. Chem. B*, **1998**, *102*, 3586–3616.

[647] A. D. MacKerell Jr.; N. Banavali; N. Foloppe. Development and current status of the CHARMM force field for nucleic acids. *Biopolymers*, **2000**, *56*, 257–265.

[648] B. R. Brooks; R. E. Bruccoleri; D. J. Olafson; D. J. States; S. Swaminathan; M. Karplus. CHARMM: A Program for Macromolecular Energy, Minimization, and Dynamics Calculations. *J. Computat. Chem.*, **1983**, *4*, 187–217.

[649] 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. CHARMM: the biomolecular simulation program. *J. Comput. Chem.*, **2009**, *30*, 1545–1614.

[650] A. D. MacKerell, Jr.; M. Feig; C. L. Brooks III. Improved Treatment of the Protein Backbone in Empirical Force Fields. *J. Am. Chem. Soc.*, **2004**, *126*, 698–699.

[651] A. D. MacKerell, Jr.; M. Feig; C. L. Brooks III. Extending the treatment of backbone energetics in protein force fields: Limitations of gas-phase quantum mechanics in reproducing protein conformational distributions in molecular dynamics simulations. *J. Computat. Chem.*, **2004**, *25*, 1400–1415.

[652] M. F. Crowley; M. J. Williamson; R. C. Walker. CHAMBER: Comprehensive support for CHARMM force fields within the AMBER software. *Int. J. Quant. Chem.*, **2009**, *109*, 3767–3772.

[653] B. J. Berne; G. D. Harp. *Adv. Chem. Phys.*, **1970**, *17*, 63.

[654] R. Kubo; M. Toda; N. Hashitsume. *Statistical Physics II: Nonequilibrium Statistical Mechanics, 2nd ed.* Springer-Verlag, Heidelberg, 1991.

[655] W. H. Miller. *Adv. Chem. Phys.*, **1974**, *25*, 69.

[656] W. H. Miller. Including quantum effects in the dynamics of complex (i.e., large) molecular systems. *J. Chem. Phys.*, **2006**, *125*, 132305.

[657] H. Wang; X. Sun; W. H. Miller. Semiclassical approximations for the calculation of thermal rate constants for chemical reactions in complex molecular systems. *J. Chem. Phys.*, **1998**, *108*, 9726.

[658] X. Sun; H. Wang; W. H. Miller. Semiclassical theory of electronically nonadiabatic dynamics: Results of a linearized approximation to the initial value representation. *J. Chem. Phys.*, **1998**, *109*, 7064.

[659] J. Liu; W. H. Miller. A simple model for the treatment of imaginary frequencies in chemical reaction rates and molecular liquids. *J. Chem. Phys.*, **2009**, *131*, 074113.

[660] J. Liu; W. H. Miller; F. Paesani; W. Zhang; D. A. Case. Quantum dynamical effects in liquid water: A semiclassical study on the diffusion and the infrared absorption spectrum. *J. Chem. Phys.*, **2009**, *131*, 164509.

[661] J. Liu. Recent advances in the linearized semiclassical initial value representation/classical wigner model for the thermal correlation function. *International Journal of Quantum Chemistry*, **2015**, *115*, 657–670.

[662] J. Liu; W. H. Miller. Real time correlation function in a single phase space integral beyond the linearized semiclassical initial value representation. *J. Chem. Phys.*, **2007**, *126*, 234110.

[663] J. Liu; W. H. Miller. Test of the consistency of various linearized semiclassical initial value time correlation functions in application to inelastic neutron scattering from liquid para-hydrogen. *J. Chem. Phys.*, **2008**, *128*, 144511.

[664] J. W. Ponder; C. Wu; P. Ren; V. S. Pande; J. D. Chodera; M. J. Schieders; I. Haque; D. L. Mobley; D. S. Lambrecht; R. A. DiStasio, Jr.; M. Head-Gordon; G. N. I. Clark; M. E. Johnson; T. Head-Gordon. Current status of the AMOEBA polarizable force field. *J. Phys. Chem. B*, **2010**, *114*, 2549–2564.

[665] L. Wickstrom; A. Okur; C. Simmerling. Evaluating the performance of the ff99SB force field based on NMR scalar coupling data. *Biophys. J.*, **2009**, *97*, 853–856.

[666] Q. Shi; E. Giva. *J. Chem. Phys. A*, **2003**, *107*, 9059.

[667] M.-J. Hsieh; R. Luo. Balancing simulation accuracy and efficiency with the Amber united atom force field. *J. Phys. Chem. B*, **2010**, *114*, 2886–2893.

[668] C. Bergonzo; A. J. Campbell; R. C. Walker; C. Simmerling. A Partial Nudged Elastic Band Implementation for Use with Large or Explicitly Solvated Systems. *Int J Quantum Chem*, **2009**, *109*, 3781–3790.

[669] I. Omelyan; A. Kovalenko. MTS-MD of biomolecules steered with 3D-RISM-KH mean solvation forces accelerated with generalized solvation force extrapolation. *J. Chem. Theory Comput.*, **2015**, *11*, 1875–1895.

[670] F. Hirata, Ed. *Molecular Theory of Solvation*. Kluwer Academic Publishers, 2003.

[671] M. E. Tuckerman; B. J. Berne; G. J. Martyna. Molecular dynamics algorithm for multiple time scales: Systems with long range forces. *J. Chem. Phys.*, **1991**, *94*, 6811–6815.

[672] M. Tuckerman; B. J. Berne; G. J. Martyna. Reversible multiple time scale molecular dynamics. *J. Chem. Phys.*, **1992**, *97*, 1990–2001.

[673] H. Grubmuumlller; H. Heller; A. Windemuth; K. Schulten. Generalized Verlet algorithm for efficient molecular dynamics simulations with long-range interactions. *Mol. Simulat.*, **1991**, *6*, 121–142.

[674] A. W. Goetz; M. J. Williamson; D. Xu; D. Poole; S. L. Grand; R. C. Walker. Routine microsecond molecular dynamics simulations with AMBER - Part I: Generalized Born. *J. Chem. Theory Comput.*, **2012**, *8*, 1542–1555.

[675] R. Salomon-Ferrer; A. W. Goetz; D. Poole; S. L. Grand; R. C. Walker. Routine microsecond molecular dynamics simulations with AMBER - Part 2: Explicit Solvent Particle Mesh Ewald . *J. Chem. Theory Comput.*, **2012**, *in review*.

[676] T. Schlick. *Molecular modeling and simulation: an interdisciplinary guide*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[677] M. K. Gilson; M. Davis; B. A. Luty; J. A. McCammon. Computation of electrostatic forces on solvated molecules using the Poisson-Boltzmann equation. *J Phys Chem*, **1993**, *97*, 3591–3600.

[678] D. W. Li; R. Brüschweiler. NMR-based protein potentials. *Angew. Chem. Int. Ed.*, **2010**, *49*, 6778–6780.

[679] K. Lindorff-Larsen; S. Piana; K. Palmo; P. Maragakis; J. Klepeis; R. O. Dror; D. E. Shaw. Improved side-chain torsion potentials for the Amber ff99SB protein force field. *Proteins*, **2010**, *78*, 1950–1958.

[680] P. Banáš; D. Hollas; M. Zgarbová; P. Jurecka; M. Orozco; T. E. Cheatham, III; J. Šponer; M. Otyepka. Performance of molecular mechanics force fields for RNA simulations: Stability of UUCG and GNRA hairpins. *J. Chem. Theory. Comput.*, **2010**, *6*, 3836–3849.

[681] R. Anandakrishnan; A. V. Onufriev. An N log N approximation based on the natural organization of biomolecules for speeding up the computation of long range interactions. *J. Comput. Chem.*, **2010**, *31*, 691–706.

[682] R. Anandakrishnan; M. Daga; A. V. Onufriev. An n log n Generalized Born Approximation. *J. Chem. Theory Comput.*, **2011**, *7*, 544–559.

[683] W. K. Olson; M. Bansal; S. K. Burley; R. E. Dickerson; M. Gerstein; S. C. Harvey; U. Heinemann; X.-J. Lu; S. Neidle; Z. Shakked; H. Sklenar; M. Suzuki; C.-S. Tung; E. Westhof; C. Wolberger; H. M. Berman. A standard reference frame for the description of nucleic acid base-pair geometry. *J. Mol. Biol.*, **2001**, *313*, 229–237.

[684] D. S. Cerutti; R. E. Duke; T. A. Darden; T. P. Lybrand. Staggered Mesh Ewald: An Extension of the Smooth Particle-Mesh Ewald Method Adding Great Versatility. *J. Chem. Theory Computat.*, **2009**, *5*, 2322–2338.

[685] D. S. Cerutti; D. A. Case. Multi-Level Ewald: A Hybrid Multigrid/Fast Fourier Transform Approach to the Electrostatic Particle-Mesh Problem. *J. Chem. Theory Comput.*, **2010**, *6*, 443–458.

[686] D. S. Cerutti; P. L. Freddolino; R. E. Duke, Jr.; D. A. Case. Simulations of a Protein Crystal with a High Resolution X-ray Structure: Evaluation of Force Fields and Water Models . *J. Phys. Chem. B*, **2010**, pp 12811–12824.

[687] M. B. Peters; Y. Yang; B. Wang; L. Fusti-Molnar; M. N. Weaver; K. M. Merz, Jr. Structural Survey of Zinc-Containing Proteins and Development of the Zinc AMBER Force Field (ZAFF). *J. Chem. Theor. Comput.*, **2010**, *6*, 2935–2947.

[688] M.-J. Hsieh; R. Luo. Exploring a coarse-grained distributive strategy for finite-difference poisson-boltzmann calculations. *J. Molec. Model.*, **2011**.

[689] I. Yildirim; H. A. Stern; S. D. Kennedy; J. D. Tubbs; D. H. Turner. Reparameterization of RNA chi Torsion Parameters for the AMBER Force Field and Comparison to NMR Spectra for Cytidine and Uridine . *J. Chem. Theory Comput.*, **2010**, *6*, 1520–1531.

[690] I. S. Joung; T. E. Cheatham, III. Molecular dynamics simulations of the dynamic and energetic properties of alkali and halide ions using water-model-specific ion parameters. *J. Phys. Chem. B*, **2009**, *113*, 13279–13290.

[691] I. Yildirim; H. A. Stern; J. D. Tubbs; S. D. Kennedy; D. H. Turner. Benchmarking AMBER Force Fields for RNA: Comparisons to NMR Spectra for Single-Stranded r(GACC) Are Improved by Revised chi Torsions. *J. Phys. Chem. B*, **2011**, *115*, 9261–9270.

[692] J. Wang; T. Hou. Application of Molecular Dynamics Simulations in Molecular Property Prediction. 1. Density and Heat of Vaporization. *J. Chem. Theory Comput.*, **2011**, *7*, 2151–2165.

[693] M. Zgarbova; M. Otyepka; J. Sponer; A. Mladek; P. Banas; T. E. Cheatham; P. Jurecka. Refinement of the Cornell et al. Nucleic Acids Force Field Based on Reference Quantum Chemical Calculations of Glycosidic Torsion Profiles. *J. Chem. Theory Comput.*, **2011**, *7*, 2886–2902.

[694] C. Perez; F. Lohr; H. Ruterjans; J. M. Schmidt. Self-Consistent Karplus Parameterization of (3)J couplings depending on the polypeptide side-chain torsion chi(1). *J. Am. Chem. Soc.*, **2001**, *123*, 7081–7093.

[695] J. J. Chou; D. A. Case; A. Bax. Insights into the mobility of methyl-bearing side chains in proteins. *J. Am. Chem. Soc.*, **2003**, *125*, 8959–8966.

[696] H.-A. Yu; B. Roux; M. Karplus. Solvation thermodynamics: An approach from analytic temperature derivatives. *J. Chem. Phys.*, **1990**, *92*, 5020–5033.

[697] T. Yamazaki; N. Blinov; D. Wishart; A. Kovalenko. Hydration effects on the HET-s prion and amyloid-β2 fibrillous aggregates, studied with three-dimensional molecular theory of solvation. *Biophys. J.*, **2008**, *95*, 4540–4548.

[698] T. Yamazaki; A. Kovalenko; V. V. Murashov; G. N. Patey. Ion solvation in a water-urea mixture. *J. Phys. Chem. B*, **2010**, *114*, 613–619.

[699] N. Homeyer; H. Gohlke. Free energy calculations by the molecular mechanics poisson-boltzmann surface area method. *Mol. Informatics*, **2012**, *DOI: 10.1002/minf.201100135*.

[700] V. Wong; D. A. Case. Evaluating rotational diffusion from protein md simulations. *J. Phys. Chem. B*, **2008**, *112*, 6013–6024.

[701] X. Lu; W. Olson. 3dna: a software package for the analysis, rebuilding and visualization of three-dimensional nucleic acid structures. *NUCLEIC ACIDS RESEARCH*, **2003**, *31*, 5108–5121.

[702] C. Altona; M. Sundaralingam. Conformational analysis of the sugar ring in nucleosides and nucleotides. a new description using the concept of pseudorotation. *J Am Chem Soc*, **1972**, *94*, 8205–8212.

[703] S. Harvey; M. Prabhakaran. Ribose puckering - structure, dynamics, energetics, and the pseudorotation cycle. *J Am Chem Soc*, **1986**, *108*, 6128–6136.

[704] D. Cremer; J. Pople. A general definition of ring puckering coordinates. *J Am Chem Soc*, **1975**, *97*, 1354–1358.

[705] T. Gaillard; D. A. Case. Evaluation of DNA Force Fields in Implicit Solvation. *J. Chem. Theory Comput.*, **2011**, *7*, 3181–3198.

[706] Y. Meng; A. E. Roitberg. Constant pH replica exchange molecular dynamics in biomolecules using a discrete protonation model. *J. Chem. Theory Comput.*, **2010**, *6*, 1401–1412.

[707] Y. Meng; D. Sabri Dashti; A. E. Roitberg. Computing Alchemical Free Energy Differences with Hamiltonian Replica Exchange Molecular Dynamics (H-REMD) Simulations. *J. Chem. Theory Comput.*, **2011**, *7*, 2721–2727.

[708] D. Sabri Dashti; A. E. Roitberg. Optimization of Umbrella Sampling Replica Exchange Molecular Dynamics by Replica Positioning. *J. Chem. Theory Comput.*, **2013**, *9*, 4692–4699.

[709] D. Sabri Dashti; A. E. Roitberg. Calculating the pKa Shift of Titratable Group at Position 66 of Staphylococcal Nuclease Mutant with the Replica Exchange Free Energy Perturbation method (REFEP). *In preparation*, **2012**.

[710] D. Sabri Dashti; Y. Meng; A. E. Roitberg. pH-Replica Exchange Molecular Dynamics in Proteins Using a Discrete Protonation Method. *J. Phys. Chem. B*, **2012**, *116*, 8805–8811.

[711] A. Pohorille; C. Jarzynski; C. Chipot. Good practices in Free-Energy calculations. *J. Phys. Chem. B*, **2010**, *114*, 10235–10253.

[712] M. R. Shirts; J. D. Chodera. Statistically optimal analysis of samples from multiple equilibrium states. *J. Chem. Phys.*, **2008**, *129*, 124105–124105–10.

[713] Å. Skjevik; B. D. Madej; R. C. Walker; K. Teigen. Lipid11: A modular framework for lipid simulations using amber. *J. Phys. Chem. B*, **2012**, *116*, 11124–11136.

[714] D. L. Mobley; C. I. Bayly; M. D. Cooper; M. R. Shirts; K. A. Dill. Small Molecule Hydration Free Energies in Explicit Solvent: An Extensive Test of Fixed-Charge Atomistic Simulations. *J. Chem. Theory Comput.*, **2009**, *5*, 350–358.

[715] X. Wu; B. R. Brooks. A virtual mixture approach to the study of multistate equilibrium: application to constant ph simulation in explicit water. *PLOS Computational Biology*, **2015**, *11*, e1004480.

[716] P. G. Karamertzanis; P. Raiteri; A. Galindo. The use of anisotropic potentials in modeling water and free energies of hydration. *J. Chem. Theory Comput.*, **2010**, *6*, 3153–3161.

[717] J. Wang; P. Cieplak; J. Li; T. Hou; R. Luo; Y. Duan. Development of Polarizable Models for Molecular Mechanical Calculations I: Parameterization of Atomic Polarizability. *J. Phys. Chem. B*, **2011**, *115*, 3091–3099.

[718] J. Wang; P. Cieplak; J. Li; J. Wang; Q. Cai; M. Hsieh; H. Lei; R. Luo; Y. Duan. Development of Polarizable Models for Molecular Mechanical Calculations II: Induced Dipole Models Significantly Improve Accuracy of Intermolecular Interaction Energies. *J. Phys. Chem. B*, **2011**, *115*, 3100–3111.

[719] J. Wang; P. Cieplak; J. Li; Q. Cai; M. Hsieh; R. Luo; Y. Duan. Development of Polarizable Models for Molecular Mechanical Calculations. 4. van der Waals Parametrization. *J. Phys. Chem. B*, **2012**, *116*, 7088–7101.

[720] J. Wang; P. Cieplak; Q. Cai; M. Hsieh; J. Wang; Y. Duan; R. Luo. Development of Polarizable Models for Molecular Mechanical Calculations. 3. Polarizable Water Models Conforming to Thole Polarization Screening Schemes. *J. Phys. Chem. B*, **2012**, *116*, 7999–8008.

[721] J. Wang; Q. Cai; Y. Xiang; R. Luo. Reducing Grid Dependence in Finite-Difference Poisson-Boltzmann Calculations. *J. Chem. Theory Comput.*, **2012**, *8*, 2741–2751.

[722] B. T. Thole. Molecular polarizabilities calculated with a modified dipole interaction. *Chem. Phys.*, **1981**, *59*, 341–350.

[723] R. Bosque; J. Sales. Polarizabilities of solvents from the chemical composition. *J. Chem. Inf. Comput. Sci.*, **2002**, *42*, 1154–1163.

[724] Z. X. Wang; C. Wu; H. X. Lei; Y. Duan. Accurate ab initio study on the hydrogen-bond pairs in protein secondary structures. *J. Chem. Theory Comput.*, **2007**, *3*, 1527–1537.

[725] J. Graf; P. H. Nguyen; G. Stock; H. Schwalbe. Structure and Dynamics of the Homologous Series of Alanine Peptides: A Joint Molecular Dynamics/NMR Study. *J. Am. Chem. Soc.*, **2007**, *129*, 1179–1189.

[726] T. E. Cheatham, III. Simulation and modeling of nucleic acid structure, dynamics and interactions. *Curr. Opin. Struct. Biol.*, **2004**, *14*, 360–367.

[727] I. Yildirim; S. D. Kennedy; H. A. Stern; J. M. Hart; R. Kierzek; D. H. Turner. Revision of AMBER Torsional Parameters for RNA Improves Free Energy Predictions for Tetramer Duplexes with GC and iGiC Base Pairs. *J. Chem. Theory Comput.*, **2012**, *8*, 172–181.

[728] M. Krepl; M. Zgarbova; P. Stadlbauer; M. Otyepka; P. Banas; J. Koca; T. E. Cheatham, III; J. Sponer. Reference simulations of noncanonical nucleic acids with different chi variants of the AMBER force field: Quadruplex DNA, quadruplex RNA, and Z-DNA. *J. Chem. Theory Comp.*, **2012**, *8*, 2506–2520.

[729] J. Wang; T. Hou. Application of Molecular Dynamics Simulations in Molecular Property Prediction. II. Diffusion coefficient. *J. Comput. Chem.*, **2011**, *32*, 3509–3519.

[730] C. F. Fu; S. X. Tian. A Comparative Study for Molecular Dynamics Simulations of Liquid Benzene. *J. Chem. Theory Comput.*, **2011**, *7*, 2240–2252.

[731] S. Tsuzuki; T. Uchimaru; K. Tanabe; S. Kuwajima. Refinement of Nonbonding Interaction Potential Parameters for Methane on the Basis of the Pair Potential Obtained by Mp3/6-311g(3d,3p)-Level Ab-Initio Molecular-Orbital Calculations - the Anisotropy of H/H Interaction. *J. Phys. Chem.*, **1994**, *98*, 1830–1833.

[732] G. A. Kaminski; R. A. Friesner; J. Tirado-Rives; W. L. Jorgensen. Evaluation and Reparametrization of the OPLS-AA Force Field for Proteins via Comparison with Accurate Quantum Chemical Calculations on Peptides. *J. Phys. Chem. B*, **2001**, *105*, 6474–6487.

[733] I. J. Chen; D. Yin; A. D. MacKerell. Combined Ab initio/Empirical Approach for Optimization of Lennard-Jones Parameters for Polar-Neutral Compounds. *J. Comput. Chem.*, **2002**, *23*, 199–213.

[734] M. L. DeMarco; R. J. Woods. Atomic-resolution conformational analysis of the G(M3) ganglioside in a lipid bilayer and its implications for ganglioside-protein recognition at membrane surfaces. *Glycobiology*, **2009**, *19*, 344–355.

[735] M. L. DeMarco; R. J. Woods; J. H. Prestegard; F. Tian. Presentation of Membrane-Anchored Glycosphingolipids Determined from Molecular Dynamics Simulations and NMR Paramagnetic Relaxation Rate Enhancement. *J. Am. Chem. Soc.*, **2010**, *132*, 1334–1338.

[736] R. Kadirvelraj; O. C. Grant; I. J. Goldstein; H. C. Winter; H. Tateno; E. Fadda; R. J. Woods. Structure and binding analysis of Polyporus squamosus lectin in complex with the Neu5Acα2-6Galβ1-4GlcNAc human-type influenza receptor. *Glycobiology*, **2011**, *21*, 973–984.

[737] B. L. Foley; M. B. Tessier; R. J. Woods. Carbohydrate force fields. *WIREs Comput. Mol. Sci.*, **2012**, *2*, 652–697.

[738] E. Ficko-Blean; C. P. Stuart; M. D. Suits; M. Cid; M. Tessier; R. J. Woods; A. B. Boraston. Carbohydrate Recognition by an Architecturally Complex α-N-Acetylglucosaminidase from Clostridium perfringens. *PLoS ONE*, **2012**, *7*, e33524.

[739] M. L. DeMarco; R. J. Woods. From agonist to antagonist: Structure and dynamics of innate immune glycoprotein MD-2 upon recognition of variably acylated bacterial endotoxins. *Mol. Immunol.*, **2011**, *49*, 124–133.

[740] N. Spackova; T. E. Cheatham; F. Ryjacek; F. Lankas; L. vanMeervelt; P. Hobza; J. Sponer. Molecular Dynamics Simulations and Thermodynamics Analysis of DNA-Drug Complexes. Minor Groove Binding between 4',6-Diamidino-2-phenylindole and DNA Duplexes in Solution. *J. Am. Chem. Soc.*, **2003**, *125*, 1759–1769.

[741] P. Varnai; D. Djuranovic; R. Lavery; B. Hartmann. alpha/gamma Transitions in the B-DNA backbone. *Nucl. Acids Res.*, **2002**, *30*, 5398–5406.

[742] D. Svozil; J. E. Sponer; I. Marchan; A. Perez; T. E. Cheatham; F. Forti; F. J. Luque; M. Orozco; J. Sponer. Geometrical and electronic structure variability of the sugar-phosphate backbone in nucleic acids. *J. Phys. Chem. B*, **2008**, *112*, 8188–8197.

[743] D. S. Cerutti; J. E. Rice; W. C. Swope; D. A. Case. Derivation of fixed partial charges for amino acids accommodating a specific water model and implicit polarization. *J. Phys. Chem. B*, **2103**, *117*, 2328–2338.

[744] T. Steinbrecher; J. Latzer; D. A. Case. Revised AMBER Parameters for Bioorganic Phosphates. *J. Chem. Theory Comput.*, **2012**, *8*, 4405–4412.

[745] F.-Y. Dupradeau; A. Pigache; T. Zaffran; C. Savineau; R. Lelong; N. Grivel; D. Lelong; W. Rosanskia; P. Cieplak. The R. E. D. tools: advances in RESP and ESP charge derivation and force field library building. *PhysChemChemPhys*, **2010**, *12*, 7821–7839.

[746] S. E. Feller. Molecular dynamics simulations of lipid bilayers. *Curr. Opin. Colloid Interface Sci.*, **2000**, *5*, 217–223.

[747] A. Lomize; I. Pogozheva; M. Lomize; H. Mosberg. The role of hydrophobic interactions in positioning of peripheral proteins in membranes. *BMC Struct. Biol.*, **2007**, *7*, 44.

[748] C. J. Dickson; L. Rosso; R. M. Betz; R. C. Walker; I. R. Gould. GAFFlipid: a General Amber Force Field for the accurate molecular dynamics simulation of phospholipid. *Soft Matter*, **2012**, *8*, 9617.

[749] R. M. Betz; N. A. DeBardeleben; R. C. Walker. An Investigation of the effects of hard and soft errors on graphics processing unit-accelerated molecular dynamics simulations. *Concurrency and Computation: Practice and Experience*, **2014**, *26*, 2134.

[750] D. E. Warschawski; P. F. Devaux. Order parameters of unsaturated phospholipids in membranes and the effect of cholesterol: a 1H-13C solid-state NMR study at natural abundance. *Eur. Biophys. J.*, **2005**, *34*, 987–996.

[751] L. Saiz; M. L. Klein. Computer Simulation Studies of Model Biological Membranes. *Acc. Chem. Res.*, **2002**, *35*, 482–489.

[752] J. T. Berryman; T. Schilling. Free Energies by Thermodynamic Integration Relative to an Exact Solution, Used to Find the Handedness-Switching Salt Concentration for DNA. *J. Chem. Theory Comput.*, **2013**, *9*, 679–686.

[753] J. T. Berryman; T. Schilling. Absolute Free Energies for Biomolecules in Implicit or Explicit Solvent. *Physics Procedia*, **2014**, *57*, 7–15.

[754] T. Schilling; F. Schmid. Computing absolute free energies of disordered structures by molecular simulation. *J. Chem. Phys.*, **2009**, *131*, 231102.

[755] F. Schmid; T. Schilling. A method to compute absolute free energies or enthalpies of fluids. *Physics Procedia*, **2010**, *4*, 131–143.

[756] D. Frenkel; A. J. C. Ladd. New Monte Carlo method to compute the free energy of arbitrary solids. Application to the fcc and hcp phases of hard spheres. *J. Chem. Phys.*, **1984**, *81*, 3188–3193.

[757] C. Vega; E. G. Noya. Revisiting the Frenkel-Ladd method to compute the free energy of solids: the Einstein molecule approach. *J. Chem. Phys.*, **2007**, *127*, 154113.

[758] J. M. Swails; A. E. Roitberg. Enhancing Conformation and Protonation State Sampling of Hen Egg White Lysozyme Using pH Replica Exchange Molecular Dynamics. *J. Chem. Theory Comput.*, **2012**, *8*, 4393–4404.

[759] S. G. Itoh; A. Damjanovic; B. R. Brooks. pH replica-exchange method based on discrete protonation states. *Proteins*, **2011**, *79*, 3420–3436.

[760] S. A. Showalter; R. Brüschweiler. Validation of molecular dynamics simulations of biomolecules using NMR spin relaxation as benchmarks: Application to the Amber99SB force field. *J. Chem. Theory Comput.*, **2007**, *3*, 961–975.

[761] R. B. Best; N.-V. Buchete; G. Hummer. Are Current Molecular Dynamics Force Fields too Helical? *Biophys. J.*, **2008**, *95*, L07–L09; 4494.

[762] R. B. Best; G. Hummer. Optimized Molecular Dynamics Force Fields Applied to the Helix-Coil Transition of Polypeptides. *J. Phys. Chem. B*, **2009**, *113*, 9904–9015.

[763] R. B. Best; J. Mittal. Free-energy landscape of the GB1 hairpin in all-atom explicit solvent simulations with different force fields: Similarities and differences. *Proteins*, **2011**, *79*, 1318–1328.

[764] K. K. Patapati; N. M. Glykos. Three force fields views of the 3-10 helix. *Biophys. J.*, **2011**, *101*, 1766–1771.

[765] S. Le Grand; A. W. Goetz; R. C. Walker. SPFP: Speed without compromise–A mixed precision model for GPU accelerated molecular dynamics simulations. *Comput. Phys. Commun.*, **2013**, *184*, 374–380.

[766] R. Salomon-Ferrer; D. A. Case; R. C. Walker. An overview of the Amber biomolecular simulation package. *WIREs Comput. Mol. Sci.*, **2013**, *3*, 198–210.

[767] PDB Current Holdings Breakdown. **2013**.

[768] M. L. Lundstrom, K. H.; Chiu. *G Protein-Coupled Receptors in Drug Discovery*. Taylor & Francis, London, 2005.

[769] C. Chipot; A. Pohorille, eds. *Free energy calculations. Theory and Applications in Chemistry and Biology*. Springer, Berlin, 2007.

[770] M. Griebel; S. Knapek; G. Zumbusch. *Numerical Simulation in Molecular Dynamics. Numerical Algorithms, Parallelization, Applications*. Springer-Verlag, Berlin, 2010.

[771] M. E. Tuckerman. *Statistical Mechanics: Theory and Molecular Simulation*. Oxford University Press, Oxford, 2010.

[772] M. Ester; H. Kriegel; J. Sander; X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *Proc. Second Int. Conf. Knowledge Disc. Data Mining (KDD-96)*, **1996**, pp 226–231.

[773] B. R. Miller; T. D. McGee; J. M. Swails; N. Homeyer; H. Gohlke; A. E. Roitberg. MMPBSA.py: An Efficient Program for End-State Free Energy Calculations. *J. Chem. Theory Comput.*, **2012**, *8*, 3314–3321.

[774] D. R. Roe; T. E. Cheatham, III. PTRAJ and CPPTRAJ: Software for Processing and Analysis of Molecular Dynamics Trajectory Data. *J. Chem. Theory Comput.*, **2013**, *9*, 3084–3095.

[775] J. M. Swails; D. M. York; A. E. Roitberg. Constant pH replica exchange molecular dynamics in explicit solvent using discrete protonation states: implementation, testing, and validation. *J. Chem. Theory Comput.*, **2014**, *10*, 1341–1352.

[776] M. Arrar; C. A. F. de Oliveira; M. Fajer; W. Sinko; J. A. McCammon. w-REXAMD: A Hamiltonian Replica Exchange Approach to Improve Free Energy Calculations for Systems with Kinetically Trapped Conformations. *J. Chem. Theory Comput.*, **2013**, *9*, 18–23.

[777] M. Fajer; D. Hamelberg; J. A. McCammon. Replica-Exchange Accelerated Molecular Dynamics (REXAMD) Applied to Thermodynamic Integration. *J. Chem. Theory Comput.*, **2008**, *4*, 1565–1569.

[778] P. Li; B. P. Roberts; D. K. Chakravorty; K. M. Merz, Jr. Rational Design of Particle Mesh Ewald Compatible Lennard-Jones Parameters for +2 Metal Cations in Explicit Solvent. *J. Chem. Theory Comput.*, **2013**, *9*, 2733–2748.

[779] P. Li; L. F. Song; K. M. Merz, Jr. Parameterization of Highly Charged Metal Ions Using the 12-6-4 LJ-Type Nonbonded Model in Explicit Water. *J. Phys. Chem. B*, **2015**, *119*, 883–895.

[780] P. Li; L. F. Song; K. M. Merz, Jr. Systematic Parameterization of Monovalent Ions Employing the Nonbonded Model. *J. Chem. Theory Comput.*, **2015**, *11*, 1645–1657.

[781] M. T. Panteva; G. M. Giambasu; D. M. York. Comparison of Structural, Thermodynamic, Kinetic and Mass Transport Properties of Mg2+ Models Commonly Used in Biomolecular Simulations. *J. Comput. Chem.*, **2015**, *36*, 970–982.

[782] M. T. Panteva; G. M. Giambasu; D. M. York. Force Field for Mg2+, Mn2+, Zn2+ and Cd2+ Ions That Have Balanced Interactions with Nucleic Acids. *J. Phys. Chem. B*, **2015**, *119*, 15460–15470.

[783] A. Ganguly; B. P. Weissman; T. J. Giese; N.-A. Li; S. Hoshika; S. Rao; A. A. Benner; J. A. Piccirilli; D. M. York. Confluence of theory and experiment reveals the catalytic mechanism of the Varkud satellite ribozyme. *Nat. Chem.*, **2020**, *12*, 192–201.

[784] P. Li; K. M. Merz, Jr. MCPB.py: A Python Based Metal Center Parameter Builder. *J. Chem. Inf. Model.*, **2016**, *56*, 599–604.

[785] P. Li; K. M. Merz, Jr. Metal Ion Modeling Using Classical Mechanics. *Chem. Rev.*, **2017**, *117*, 1564–1686.

[786] Z. Yu; P. Li; K. M. Merz, Jr. Extended Zinc AMBER Force Field (EZAFF). *J. Chem. Theory Comput.*, **2018**, *14*, 242–254.

[787] A. Sengupta; A. Seitz; K. M. Merz, Jr. Simulating the Chelate Effect. *J. Am. Chem. Soc.*, **2018**, *140*, 15166–15169.

[788] Z. Li; S. F. Lin; P. Li; K. M. Merz, Jr. Systematic Parametrization of Divalent Metal Ions for the OPC3, OPC, TIP3P-FB, and TIP4P-FB Water Models. *J. Chem. Theory Comput.*, **2020**, *16*, 4429–4442.

[789] S. F. Lin; A. Sengupta; K. M. Merz, Jr. Thermodynamics of Transition Metal Ion Binding to Proteins. *J. Am. Chem. Soc.*, **2020**, *142*, 6365–6374.

[790] P. Li; K. M. Merz, Jr. in *Methods Mol. Biol.*, (Humana Press, New York, NY). volume 2199. pp 257–275. 2021.

[791] A. Sengupta; Z. Li; S. F. Lin; P. Li; K. M. Merz, Jr. Parameterization of Monovalent Ions for the OPC3, OPC, TIP3P-FB, and TIP4P-FB Water Models. *J. Chem. Inf. Model.*, **2021**, *61*, 869–880.

[792] Z. Li; S. F. Lin; P. Li; K. M. Merz, Jr. Parametrization of Trivalent and Tetravalent Metal Ions for the OPC3, OPC, TIP3P-FB, and TIP4P-FB Water Models. *J. Chem. Theory Comput.*, **2021**, *17*, 2342–2354.

[793] J. M. Seminario. Calculation of Intramolecular Force Fields from Second-Derivative Tensors. *Int. J. Quantum Chem.*, **1996**, *30*, 1271–1277.

[794] P. Eastman; V. S. Pande. OpenMM: A Hardware-Independent Framework for Molecular Simulations. *Computing in Science and Engineering*, **2010**, *12*, 34–39.

[795] P. Eastman; M. S. Friedrichs; J. D. Chodera; R. J. Radmer; C. M. Bruns; J. P. Ku; K. A. Beauchamp; T. J. Lane; L. Wang; D. Shukla; T. Tye; M. Houston; T. Stich; C. Klein; M. R. Shirts; V. S. Pande. OpenMM 4: A Reusable, Extensible, Hardware Independent Library for High Performance Molecular Simulation. *J. Chem. Theory Comput.*, **2013**, *9*, 461–469.

[796] J. M. Swails. *Free Energy Simulations of Complex Biological Systems at Constant pH*. PhD thesis, University of Florida, 2013.

[797] C. N. Nguyen; T. Kurtzman Young; M. K. Gilson. Grid Inhomogeneous solvation theory: Hydration structure and thermodynamics of the miniature receptor cucurbit[7]uril. *J. Chem. Phys.*, **2012**, *137*, 044101–044118.

[798] C. Bergonzo; N. M. Henriksen; D. R. Roe; J. M. Swails; A. E. Roitberg; T. E. Cheatham III. Multidimensional Replica Exchange Molecular Dynamics Yields a Converged Ensemble of an RNA Tetranucleotide. *J. Chem. Theory Comput.*, **2013**, *10*, 492–499.

[799] A. Bakan; L. M. Meireles; I. Bahar. ProDy: Protein Dynamics Inferred from Theory and Experiments. *Bioinformatics*, **2011**, *27*, 1575–1577.

[800] N. Bernstein; C. Várnai; I. Solt; S. A. Winfield; M. C. Payne; I. Simon; M. Fuxreiter; G. Csányi. *Phys. Chem. Chem. Phys.*, **2011**, *14*, 646–656.

[801] C. Várnai; N. Bernstein; L. Mones; G. Csányi. Tests of an adaptive qm/mm calculation on free energy profiles of chemical reactions in solution. *J. Phys. Chem. B*, **2013**, *117*, 12202–12211.

[802] G. Csányi; T. Albaret; G. Moras; M. C. Payne; A. D. Vita. Multiscale hybrid simulation methods for material systems. *J. Phys. Condens. Matt.*, **2005**, *17*, R691.

[803] N. Bernstein; J. R. Kermode; G. Csányi. Hybrid atomistic simulation methods for materials systems. *Rep. Prog. Phys.*, **2009**, *72*, 026501.

[804] A. Jones; B. Leimkuhler. Adaptive stochastic methods for sampling driven molecular systems. *J. Chem. Phys.*, **2011**, *135*, 084125.

[805] T. Kerdcharoen; B. M. Rode. A QM/MM simulation method applied to the solution of Li+ in liquid ammonia. *Chem. Phys.*, **1996**, *211*, 313–323.

[806] N. Homeyer; H. Gohlke. FEW - A workflow tool for free energy calculations of ligand binding. *J. Comput. Chem.*, **2013**, *34*, 965–973.

[807] A. Metz. *Goethe University (Frankfurt am Main)*, **2006**.

[808] J. Åqvist; C. Medina; J. E. Samuelsson. A new method for predicting binding affinity in computer-aided drug design. *Protein Eng.*, **1994**, *7*, 385–391.

[809] J. Åqvist; V. B. Luzhkov; B. O. Brandsdal. Ligand binding affinities from MD simulations. *Acc. Chem. Res.*, **2002**, *35*, 358–365.

[810] H. G. Wallnoefer; K. R. Liedl; T. Fox. A challenging system: free energy prediction for factor Xa. *J. Comput. Chem.*, **2011**, *32*, 1743–1752.

[811] M. M. van Lipzig; A. M. ter Laak; A. Jongejan; N. P. Vermeulen; M. Wamelink; D. Geerke; J. H. Meerman. Prediction of ligand binding affinity and orientation of xenoestrogens to the estrogen receptor by molecular dynamics simulations and the linear interaction energy method. *J. Med. Chem.*, **2004**, *47*, 1018–1030.

[812] B. O. Brandsdal; F. Österberg; M. Almlöf; I. Feierberg; V. B. Luzhkov; Åqvist, J. Free energy calculations and ligand binding. *Adv. Protein Chem.*, **2003**, *66*, 123–158.

[813] W. Wang; J. Wang; P. A. Kollman. What determines the van der Waals coefficient beta in the LIE (linear interaction energy) method to estimate binding free energies using molecular dynamics simulations? *Proteins: Struct., Funct., Genet.*, **1999**, *34*, 395–402.

[814] D. K. Jones-Hertzog; W. L. Jorgensen. Binding affinities for sulfonamide inhibitors with human thrombin using Monte Carlo simulations with a linear response method. *J. Med. Chem.*, **1997**, *40*, 1539–1549.

[815] M. L. Lamb; J. Tirado-Rives; W. L. Jorgensen. Estimation of the binding affinities of FKBP12 inhibitors using a linear response method. *Bioorg. Med. Chem.*, **1999**, *7*, 851–860.

[816] W. Yang; R. Bitetti-Putzer; K. M. Free energy simulations: Use of reverse cumulative averaging to determine the equilibrated region and the time required for convergence. *J. Chem. Phys.*, **2004**, *120*, 2618–2628.

[817] J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proc. Amer. Math. Soc.*, **1956**, *7*, 48–50.

[818] *ROCS, OpenEye Scientific Software, Santa Fe*, *http://www.eyesopen.com*.

[819] P. C. D. Hawkins; A. G. Skillman; A. Nicholls. Comparison of shape-matching and docking as virtual screening tools. *J. Med. Chem.*, **2007**, *50*, 74–82.

[820] T. E. Cheatham; D. A. Case. Twenty-five years of nucleic acid simulations. *Biopolymers*, **2013**, *99*, 969–977.

[821] C. J. Cramer. *Essentials of Computational Chemistry: Theories and Models*. John Wiley & Sons, New York, 2002.

[822] T. Lazaridis. Inhomogeneous Fluid Approach to Solvation Thermodynamics. 1 Theory. *J. Phys. Chem. B*, **1998**, *102*, 3531–3541.

[823] C. N. Nguyen; T. Kurtzman Young; M. K. Gilson. Grid Inhomogeneous Solvation Theory: Hydration Structure and Thermodynamics of the Miniature Receptor cucurbit[7]uril. *J. Chem. Phys.*, **2012**, *137*, 044101.

[824] S. Chatterjee; P. G. Debenedetti; F. H. Stillinger; R. M. Lynden-Bell. A Computational Investigation of Thermodynamics, Structure, Dynamics and Solvation Behavior in Modified Water Models. *J. Chem. Phys.*, **2008**, *128*, 124511.

[825] W. Humphrey; A. Dalke; K. Schulten. VMD Visual Molecular Dynamics. *J. Molec. Graph.*, **1996**, *14*, 33–38.

[826] D. J. Sindhikara; N. Yoshida; F. Hirata. Placevent: An Algorithm for Prediction of Explicit Solvent Atom Distribution-Application to HIV-1 Protease and F-ATP Synthase. *J. Comput. Chem.*, **2012**, *33*, 1536–1543.

[827] C. J. Dickson; B. D. Madej; A. A. Skjevik; R. M. Betz; K. Teigen; I. R. Gould; R. C. Walker. Lipid14: The Amber Lipid Force Field. *J. Chem. Theory Comput.*, **2014**, *10*, 865–879.

[828] R. Konecny; N. A. Baker; J. A. McCammon. iAPBS: a programming interface to the adaptive Poisson–Boltzmann solver. *Comput. Sci. Disc.*, **2012**, *5*, 15005–15013.

[829] R. Anandakrishnan; C. Baker; S. Izadi; A. V. Onufriev. Point charges optimally placed to represent the multipole expansion of charge distributions. *PloS one*, **2013**, *8*, e67715.

[830] D. P. Fernandez; A. R. H. Goodwin; E. W. Lemmon; J. M. H. Levelt Sengers; R. C. Williams. A formulation for the static permittivity of water and steam at temperatures from 238 k to 873 k at pressures up to 1200 mpa, including derivatives and debye huckel coefficients. *J. Phys. Chem. Ref. Data*, **1997**, *26*, 1125–1166.

[831] R. Mills. Self-diffusion in normal and heavy water in the range 1-45. deg. *J. Phys. Chem.*, **1973**, *77*, 685–688.

[832] W. Wagner; A. Pruss. The iapws formulation 1995 for the thermodynamic properties of ordinary water substance for general and scientific use. *J. Phys. Chem. Ref. Data*, **2002**, *31*, 387–535.

[833] L. B. Skinner; C. Huang; D. Schlesinger; L. G. M. Pettersson; A. Nilsson; C. J. Benmore. Benchmark oxygen-oxygen pair-distribution function of ambient water from x-ray diffraction measurements with a wide q-range. *J. Chem. Phys.*, **2013**, *138*, 074506+.

[834] G. S. Kell. Precise representation of volume properties of water at one atmosphere. *J. Chem. Eng. Data*, **1967**, *12*, 66–69.

[835] S. Niu; M. L. Tan; T. Ichiye. The large quadrupole of water molecules. *J. Chem. Phys.*, **2011**, *134*, 134501+.

[836] S. Izadi; R. Anandakrishnan; A. V. Onufriev. Building Water Models: A Different Approach. *J. Phys. Chem. Lett.*, **2014**, *5*, 3863–3871.

[837] A. Mukhopadhyay; A. T. Fenley; I. S. Tolokh; A. V. Onufriev. Charge hydration asymmetry: the basic principle and how to use it to test and improve water models. *J. Phys. Chem. B*, **2012**, *116*, 9776–9783.

[838] B. Aguilar; R. Shadrach; A. V. Onufriev. Reducing the secondary structure bias in the generalized born model via r6 effective radii. *J. Chem. Theory Comput.*, **2010**, *6*, 3613–3630.

[839] B. Aguilar; A. V. Onufriev. Efficient computation of the total solvation energy of small molecules via the r6 generalized born model. *J. Chem. Theory Comput.*, **2012**, *8*, 2404–2411.

[840] A. Mukhopadhyay; B. H. Aguilar; I. S. Tolokh; A. V. Onufriev. Introducing charge hydration asymmetry into the generalized born model. *J Chem. Theory Comput.*, **2014**, *10*, 1788–1794.

[841] B. Schneider; S. Neidle; H. M. Berman. Conformations of the sugar-phosphate backbone in helical dna crystal structures. *Biopolymers*, **1997**, *42*, 113–124.

[842] B. Schneider; Z. Moravek; H. M. Berman. Rna conformational classes. *Nucleic Acids Res.*, **2004**, *32*, 1666–1677.

[843] G. Monard; M. I. Bernal-Uruchurtu; A. Van Der Vaart; K. M. Merz, Jr.; M. F. Ruiz-López. Simulation of liquid water using semiempirical Hamiltonians and the divide and conquer approach. *J. Phys. Chem. A*, **2005**, *109*, 3425–3432.

[844] A. Marion; G. Monard; M. F. Ruiz-López; F. Ingrosso. Water interactions with hydrophobic groups: assessment and recalibration of semiempirical molecular orbital methods. *J. Chem. Phys.*, **2014**, *141*, 034106.

[845] A. Marion; H. Gockan; G. Monard. SemiEmpirical Born-Oppenheimer Molecular Dynamics (SEBOMD) Within the Amber Biomolecular Package. *J. Chem. Inf. Model.*, **2019**, *59*, 206–214.

[846] E. Thiriot; G. Monard. Combining a genetic algorithm with a linear scaling semiempirical method for protein-ligand docking. *J. Mol. Struct. Theochem*, **2009**, *898*, 31–41.

[847] W. Harb; M. I. Bernal-Uruchurtu; M. F. Ruiz-López. An improved semiempirical method for hydrated systems. *Theor. Chem. Acc.*, **2004**, *112*, 204–216.

[848] M. I. Bernal-Uruchurtu; M. T. C. Martins-costa; C. Millot; M. F. Ruiz-López. Improving Description of Hydrogen Bonds at the Semiempirical Level : Water - Water Interactions as Test Case. *J. Comput. Chem.*, **2000**, *21*, 572–581.

[849] O. Ludwig; H. Schinke; W. Brandt. Reparametrisation of Force Constants in MOPAC 6.0/7.0 for Better Description of the Activation Barrier of Peptide Bond Rotations. *J. Molec. Model.*, **1996**, *2*, 341–350.

[850] M. Zgarbová; F. J. Luque; J. Šponer; T. E. C. III; M. Otyepka; P. Jurečka. Toward improved description of dna backbone: Revisiting epsilon and zeta torsion force field parameters. *J. Chem. Theory Comput.*, **2013**, *9*, 2339–2354.

[851] D. S. Cerutti; W. C. Swope; J. E. Rice; D. A. Case. ff14ipq: A Self-Consistent Force Field for Condensed-Phase Simulations of Proteins. *J. Chem. Theory Comput.*, **2014**, *10*, 4515–4534.

[852] B. D. Madej; I. R. Gould; R. C. Walker. A Parameterization of Cholesterol for Mixed Lipid Bilayer Simulation within the Amber Lipid14 Force Field. *J Phys Chem B*, **2015**, *119*, 12424–12435.

[853] C. Vega; J. L. F. Abascal. Simulating water with rigid non-polarizable models: a general perspective. *Phys Chem Chem Phys*, **2011**, *13*, 19663–19688.

[854] P. H. Hunenberger; A. E. Mark; W. F. van Gunsteren. Fluctuation and Cross-correlation Analysis of Protein Motions Observed in Nanosecond Molecular Dynamics Simulations. *J. Mol. Biol.*, **1995**, *252*, 492–503.

[855] R. Galindo-Murillo; D. R. Roe; T. E. Cheatham, III. Convergence and reproducibility in molecular dynamics simulations of the DNA duplex d(GCACGAACGAACGAACGC). *Biochim. Biophys. Acta*, **2015**, *1850*, 1041–1058.

[856] J. Domanski; P. Stansfeld; M. S. P. Sansom; O. Beckstein. Lipidbook: A Public Repository for Force Field Parameters Used in Membrane Simulations. *J. Membrane Biol.*, **2010**, *236*, 255–258.

[857] S. Jo; T. Kim; W. Im. Automated builder and database of protein/membrane complexes for molecular dynamics simulations. *PLoS One*, **2007**, *2*, e880.

[858] S. Jo; T. Kim; V. G. Iyer; I. W. CHARMM-GUI: a web-based graphical user interface for CHARMM. *J. Comput. Chem.*, **2008**, *29*, 1859–1865.

[859] S. Jo; J. B. Lim; J. B. Klauda; W. Im. CHARMM-GUI Membrane Builder for mixed bilayers and its application to yeast membranes. *Biophys. J.*, **2009**, *97*, 50–58.

[860] E. L. Wu; X. Cheng; S. Jo; H. Rui; K. C. Song; E. M. Davila-Contreras; Y. Qi; J. Lee; V. Monje-Galvan; R. M. Venable; J. B. Klauda; I. W. CHARMM-GUI Membrane Builder toward realistic biological membrane simulations. *J. Comput. Chem.*, **2014**, *35*, 1997–2004.

[861] N. A. Baker; D. Sept; J. Simpson; M. J. Holst; M. J. A. Electrostatics of nanosystems: application to microtubules and the ribosome. *Proc. Natl. Acad. Sci. U. S. A.*, **2001**, *98*, 10037–10041.

[862] M. Holst; F. Saied. Multigrid solution of the Poisson-Boltzmann equation. *J. Comput. Chem.*, **1993**, *14*, 105–113.

[863] M. Holst; F. Saied. Numerical solution of the nonlinear Poisson-Boltzmann equation: Developing more robust and efficient methods. *J. Comput. Chem.*, **1995**, *16*, 337–364.

[864] M. Holst. Adaptive numerical treatment of elliptic systems on manifolds. *Adv. Comput. Math.*, **2001**, *15*, 139–191.

[865] R. Bank; M. Holst. A New Paradigm for Parallel Adaptive Meshing Algorithms. *SIAM Review*, **2003**, *45*, 291–323.

[866] K. M. Callenberg; O. P. Choudhary; G. L. de Forest; D. W. Gohara; N. A. Baker; M. Grabe. APBSmem: A graphical interface for electrostatic calculations at the membrane. *PLoS One*, **2010**, *5*, e12722.

[867] H. Nymeyer; H. X. Zhou. A method to determine dielectric constants in nonhomogeneous systems: application to biological membranes. *Biophys. J.*, **2008**, *94*, 1185–1193.

[868] H. A. Stern; S. E. Feller. Calculation of the dielectric permittivity profile for a nonuniform system: application to a lipid bilayer simulation. *J. Chem. Phys.*, **2003**, *118*, 3401–3412.

[869] N. Homeyer; H. Gohlke. Extension of the free energy workflow FEW towards implicit solvent/implicit membrane MM-PBSA calculations. *BBA - Gen. Subjects*, **2015**, *1850*, 972–982.

[870] L. Waeschenbach; C. G. W. Gertzen; V. Keitel; H. Gohlke. Dimerization energetics of the G-protein coupled bile acid receptor TGR5 from all-atom simulations. *J. Comput. Chem.*, **2019**, *41*.

[871] J. Z. Ruscio; D. Kumar; M. Shukla; M. G. Prisant; T. M. Murali; A. V. Onufriev. Atomic level computational identification of ligand migration pathways between solvent and binding site in myoglobin. *Proc. Nat. Acad. Sci. USA*, **2008**, *105*, 9204–9209.

[872] N. Homeyer; A. H. C. Horn; H. Lanig; H. Sticht. AMBER force-field parameters for phosphorylated amino acids in different protonation states: phosphoserine, phosphothreonine, phosphotyrosine, and phosphohistidine. *J. Mol. Model.*, **2006**, *12*, 281–289.

[873] M. Grabe; H. Lecar; Y. N. Jan; J. L. Y. A quantitative assessment of models for voltage-dependent gating of ion channels. *Proc. Natl. Acad. Sci. U. S. A.*, **2004**, *101*, 17640–17645.

[874] J. A. Maier; C. Martinez; K. Kasavajhala; L. Wickstrom; K. E. Hauser; C. Simmerling. ff14SB: Improving the Accuracy of Protein Side Chain and Backbone Parameters from ff99SB. *J. Chem. Theory Comput.*, **2015**, *11*, 3696–3713.

[875] K. Takemura; A. Kitao. Water Model Tuning for Improved Reproduction of Rotational Diffusion and NMR Spectral Density. *J. Phys. Chem. B*, **2012**, *116*, 6279–6287.

[876] L.-P. Wang; T. J. Martinez; V. S. Pande. Building force fields: An automatic, systematic and reproducible approach. *J. Phys. Chem. Lett.*, **2014**, *5*, 1885–1891.

[877] I. Omelyan; A. Kovalenko. MTS-MD of Biomolecules Steered with 3D-RISM-KH Mean Solvation Forces Accelerated with Generalized Solvation Force Extrapolation. *J. Chem. Theor. and Comp.*, **2014**, *11*, 1875–1895.

[878] H. Nguyen; D. R. Roe; J. M. Swails; D. A. Case. PYTRAJ: Interactive data analysis for molecular dynamics simulations. *Manuscript in preparation*, **2016**.

[879] D. L. Blood; A. M. Rosnik; B. P. Krueger. Molecular dynamics parameters for the gfp chromophore and some of its analogues. *Manuscript in preparation*, **2016**.

[880] I. Ivani; P. D. Dans; A. Noy; A. Pérez; I. Faustino; A. Hopsital; J. Walther; P. Andrió; R. Goni; A. Balaceanu; G. Portella; F. Battistini; J. L. GelpÃ; C. González; M. Vendruscolo; C. A. Laughton; S. Harris; D. A. Case; M. Orozco. Parmbsc1: A refined force field for DNA simulations. *Nature Meth.*, **2016**, *13*, 55–58.

[881] M. Zgarbová; J. Sponer; M. Otyepka; T. E. Cheatham, III; R. Galindo-Murillo; P. Jurečka. Refinement of the Sugar-Phosphate Backbone Torsion Beta for AMBER Force Fields Improves the Description of Z- and B-DNA. *J. Chem. Theor. and Comp.*, **2015**, *12*, 5723–5736.

[882] C. Bergonzo; T. E. C. III. Improved force field parameters lead to a better description of rna structure. *J. of Chem. Theory Comput.*, **2015**, *11*, 3969–3972.

[883] K. Gao; J. Yin; N. M. Henriksen; A. T. Fenley; M. K. Gilson. Binding enthalpy calculations for a neutral host-guest pair yield widely divergent salt effects across water models. *J. of Chem. Theory Comput.*, **2015**, *11*, 4555–4564.

[884] S. Izadi; B. Aguilar; A. V. Onufriev. Protein-ligand electrostatic binding free energies from explicit and implicit solvation. *J. Chem. Theory Comput.*, **2015**, *11*, 4450–4459.

[885] C. Torrence; G. P. Compo. A practical guide to wavelet analysis. *Bull. Am. Meteorol. Soc.*, **1998**, *79*, 61–78.

[886] N. C. Benson; V. Dagget. Wavelet analysis of protein motion. *Int. J. Wavelets Multi.*, **2012**, *10*.

[887] A. Rodriguez; A. Laio. Clustering by fast search and find of density peaks. *Science*, **2014**, *344*, 1492–1496.

[888] D. R. Roe; C. Bergonzo; T. E. C. III. Evaluation of enhanced sampling provided by accelerated molecular dynamics with hamiltonian replica exchange methods. *J. Phys. Chem. B*, **2014**, *118*, 3543–3552.

[889] M. A. E. Hassan; C. R. Calladine. Two distinct modes of protein-induced bending in dna. *J. Mol. Biol.*, **1998**, *282*, 331–343.

[890] K. T. Debiec; D. S. Cerutti; L. R. Baker; A. M. Gronenborn; D. A. Case; L. T. Chong. Further along the Road Less Traveled: AMBER ff15ipq, an Original Protein Force Field Built on a Self-Consistent Physical Model. *J. Chem. Theory Comput.*, **2016**, *12*, 3926–3947.

[891] R. Galindo-Murillo; J. C. Robertson; M. Zgarbovic; J. Sponer; M. Otyepka; P. Jureska; T. E. Cheatham. Assessing the Current State of Amber Force Field Modifications for DNA. *J. Chem. Theory Comput.*, **2016**, *12*, 4114–4127.

[892] S. Izadi; A. V. Onufriev. Accuracy limit of rigid 3-point water models. *J. Chem. Phys.*, **2016**, *145*, 074501.

[893] S. Izadi; R. Anandakrishnan; A. V. Onufriev. Implicit solvent model for Million-Atom atomistic simulations: Insights into the organization of 30-nm chromatin fiber. *J. Chem. Theory Comput.*, **2016**, *12*, 5946–5959.

[894] A. H. Aytenfisu; A. Spasic; A. Grossfield; H. A. Stern; D. H. Mathews. Revised RNA Dihedral Parameters for the Amber Force Field Improve RNA Molecular Dynamics. *J. Chem. Theory Comput.*, **2017**, *13*, 900–915.

[895] Z. Heidari; D. R. Roe; R. Galindo-Murillo; J. B. Ghasemi; T. E. Cheatham, III. Using Wavelet Analysis To Assist in Identification of Significant Events in Molecular Dynamics Simulations. *J. Chem. Inf. Model.*, **2016**, *56*, 1282–1291.

[896] G. Cui; J. M. Swails; E. S. Manas. SPAM: A Simple Approach for Profiling Bound Water Molecules. *J. Chem. Theory Comput.*, **2013**, *9*, 5539–5549.

[897] L. Wang; K. A. McKiernan; J. Gomes; K. A. Beauchamp; T. Head-Gordon; J. E. Rice; W. C. Swope; T. J. MartÃnez; V. S. Pande. Building a More Predictive Protein Force Field: A Systematic and Reproducible Route to AMBER-FB15. *J. Phys. Chem. B*, **2017**, *121*, 4023–4039.

[898] N. Forouzesh; S. Izadi; A. V. Onufriev. Grid-Based Surface Generalized Born Model for Calculation of Electrostatic Binding Free Energies. *J. Chem. Inf. Model.*, **2017**, *57*, 2505–2513.

[899] D. S. Cerutti; K. T. Debiec; D. A. Case; L. T. Chong. Links between the charge model and bonded parameter force constants in biomolecular force fields. *J. Chem. Phys.*, **2017**, *147*, 161730.

[900] V. W. D. Cruzeiro; M. S. Amaral; A. E. Roitberg. Redox Potential Replica Exchange Molecular Dynamics at constant pH in AMBER: Implementation, Validation and Application. *J. Chem. Phys.*, **2018**, *149*, 072338.

[901] V. W. D. Cruzeiro; A. E. Roitberg. Multidimensional Replica Exchange Simulations for Efficient Constant pH and Redox Potential Molecular Dynamics. *J. Chem. Theory Comput.*, **2019**.

[902] J. Khandogin; C. L. Brooks, III. Constant pH molecular dynamics with proton tautomerism. *Biophys. J.*, **2005**, *89*, 141–157.

[903] J. Khandogin; C. L. Brooks, III. Toward the accurate first-principles prediction of ionization equilibria in proteins. *Biochemistry*, **2006**, *45*, 9363–9373.

[904] J. A. Wallace; J. K. Shen. Continuous constant pH molecular dynamics in explicit solvent with pH-based replica exchange. *J. Chem. Theory Comput.*, **2011**, *7*, 2617–2629.

[905] J. A. Wallace; J. K. Shen. Charge-leveling and proper treatment of long-range electrostatics in all-atom molecular dynamics at constant pH. *J. Chem. Phys.*, **2012**, *137*, 184105.

[906] W. Chen; J. A. Wallace; Z. Yue; J. K. Shen. Introducing titratable water to all-atom molecular dynamics at constant pH. *Biophys. J.*, **2013**, *105*, L15–L17.

[907] Y. Huang; W. Chen; J. A. Wallace; J. Shen. All-Atom continuous constant pH molecular dynamics with particle mesh Ewald and titratable water. *J. Chem. Theory Comput.*, **2016**, *12*, 5411–5421.

[908] Y. Huang; R. C. Harris; J. Shen. Generalized Born based continuous constant pH molecular dynamics in Amber: implementation, benchmarking, and analysis. *J. Chem. Inform. Model.*, **2018**, *58*, 1372–1383.

[909] R. C. Harris; J. Shen. GPU-Accelerated Implementation of Continuous Constant pH Molecular Dynamics in Amber: pKa Predictions with Single-pH Simulations. *J. Chem. Inform. Model.*, **2019**, *59*, 4821–4832.

[910] O. N. Starovoytov; H. Torabifard; G. A. Cisneros. Development of amoeba force field for 1,3-dimethylimidazolium based ionic liquids. *J. Phys. Chem. B*, **2014**, *118*, 7156–7166.

[911] Y.-J. Tu; Z. Lin; M. J. Allen; G. A. Cisneros. Molecular dynamics investigation of water-exchange reactions on lanthanide ions in water/1-ethyl-3-methylimidazolium trifluoromethylsufate ([emim][otf]). *J. Chem. Phys.*, **2018**, *148*, 024503.

[912] Y.-J. Tu; M. J. Allen; G. A. Cisneros. Simulations of the water exchange dynamics of lanthanide ions in 1-ethyl-3-methylimidazolium ethyl sulfate ([emim][etso4]) and water. *Phys. Chem. Chem. Phys.*, **2016**, *18*, 30323–30333.

[913] H. Torabifard; O. N. Starovoytov; P. Ren; G. A. Cisneros. Development of an amoeba water model using gem distributed multipoles. *Theo. Chem. Acc.*, **2015**, *134*, 1–10.

[914] H. Torabifard; L. Reed; M. T. Berry; J. E. Hein; E. Menke; G. A. Cisneros. Computational and experimental characterization of a pyrrolidinium-based ionic liquid for electrolyte applications. *J. Chem. Phys.*, **2017**, *147*, 161731.

[915] G. A. Cisneros. Application of gaussian electrostatic model (gem) distributed multipoles in the amoeba force field. *J. Chem. Theo. Comput.*, **2012**, *12*, 5072–5080.

[916] G. A. Cisneros; J.-P. Piquemal; T. A. Darden. Generalization of the gaussian electrostatic model: extension to arbitrary angular momentum, distributed multipoles and computational speedup with reciprocal space methods. *J. Chem. Phys.*, **2006**, *125*, 184101.

[917] J.-P. Piquemal; G. A. Cisneros; P. Reinhardt; N. Gresh; T. A. Darden. Towards a force field based on density fitting. *J. Chem. Phys.*, **2006**, *124*, 104101.

[918] J.-P. Piquemal; G. Cisneros. in *Many-body effects and electrostatics in multi-scale computations of Biomolecules*, Q. Cui; P. Ren; M. Meuwly, Eds., pp 269–300. Pan Stanford Publishing, 2015.

[919] R. E. Duke; O. N. Starovoytov; J.-. Piquemal; G. A. Cisneros. Gem*: A molecular electronic density-based force field for molecular dynamics simulations. *J. Chem. Theo. Comput.*, **2014**, *10*, 1361–1365.

[920] J.-P. Ryckaert; G. Ciccotti; H. J. Berendsen. Numerical integration of the cartesian equations of motion of a system with constraints: molecular dynamics of n-alkanes. *J. of Computat. Phys.*, **1977**, *23*, 327 – 341.

[921] S. Myamoto; P. A. Kollman. Settle: An analytical version of the SHAKE and RATTLE algorithm for rigid water models. *J. Computat. Chem.*, **1992**, *13*, 952–962.

[922] T. Nugent; D. T. Jones. Membrane protein orientation and refinement using a knowledge-based statistical potential. *BMC Bioinformatics*, **2013**, *14*, 276.

[923] S. Schott-Verdugo; H. Gohlke. PACKMOL-Memgen: A simple-to-use generalized workflow for membrane-protein/lipid-bilayer system building. *J. Chem. Inf. Model.*, **2019**, *59*, 2522–2528.

[924] L. Martínez; R. Andrade; E. G. Birgin; J. M. Martínez. PACKMOL: A package for building initial configurations for molecular dynamics simulations. *J. Comput. Chem.*, **2009**, *30*, 2157–2164.

[925] J. M. Martínez; L. Martínez. Packing optimization for automated generation of complex system's initial configurations for molecular dynamics and docking. *J. Computat. Chem.*, **2003**, *24*, 819–825.

[926] B. Ho. pdbremix. https://github.com/boscoh/pdbremix, 2018.

[927] H. C. Andersen. Molecular dynamics simulations at constant pressure and/or temperature. *J. Chem. Phys.*, **1980**, *72*, 2384–2393.

[928] C.-E. Chang; W. Chen; M. K. Gilson. Evaluating the Accuracy of the Quasiharmonic Approximation. *J. Chem. Theory Computat*, **2005**, *1*, 1017–1028. PMID: 26641917.

[929] D. A. McQuarrie. *Statistical Thermodynamics*. Harper and Row, New York, 1973.

[930] S. Izadi; R. C. Harris; M. O. Fenley; A. V. Onufriev. Accuracy comparison of generalized born models in the calculation of electrostatic binding free energies. *J. Chem. Theory Computat.*, **2018**, *14*, 1656–1670.

[931] H. Nguyen; D. A. Case; A. S. Rose. Nglview–interactive molecular graphics for jupyter notebooks. *Bioinformatics*, **2017**, *34*, 1241–1242.

[932] D. R. Roe; T. E. Cheatham III. Parallelization of CPPTRAJ enables large scale analysis of molecular dynamics trajectory data. *J. Computat. Chem.*, **2018**, *39*, 2110–2117.

[933] H. A. Boateng; R. Krasny. Comparison of treecodes for computing electrostatic potentials in charged particle systems with disjoint targets and sources. *J. Computat. Chem.*, **2013**, *34*, 2159–2167.

[934] Z.-H. Duan; R. Krasny. An adaptive treecode for computing nonbonded potential energy in classical molecular systems. *J. Computat. Chem.*, **2001**, *22*, 184–195.

[935] P. Li; H. Johnston; R. Krasny. A Cartesian treecode for screened Coulomb interactions. *J. Computat. Phys.*, **2009**, *228*, 3858–3868.

[936] T. Graen; M. Hoefling; H. Grubmueller. Amber-dyes: Characterization of charge fluctuations and force field parameterization of fluorescent dyes for molecular dynamics simulations. *Journal of Chemical Theory and Computation*, **2014**, *10*, 5505–5512.

[937] B. Schepers; H. Gohlke. Amber-dyes in amber: Implementation of fluorophore and linker parameters into ambertools. *Journal of Chemical Physics*, **2020**, *152*.

[938] S. Kalinin; T. Peulen; S. Sindbert; P. J. Rothwell; S. Berger; T. Restle; R. S. Goody; H. Gohlke; C. A. Seidel. A toolkit and benchmark study for fret-restrained high-precision structural modeling. *Nature Methods*, **2012**, *9*, 1218–1225.

[939] M. Dimura; T. O. Peulen; C. A. Hanke; A. Prakash; H. Gohlke; C. A. Seidel. Quantitative fret studies and integrative modeling unravel the structure and dynamics of biomolecular systems. *Curr. Opin. Struct. Biol.*, **2016**, *40*, 163–185.

[940] M. Dimura; T. O. Peulen; H. Sanabria; D. Rodnin; K. Hemmen; C. A. Seidel; H. Gohlke. Automated and optimally fret-assisted structural modeling. **2019**.

[941] R. T. McGibbon; K. A. Beauchamp; M. P. Harrigan; C. Klein; J. M. Swails; C. X. Hernandez; C. R. Schwantes; L.-P. Wang; T. J. Lane; V. S. Pande. Mdtraj: A modern open library for the analysis of molecular dynamics trajectories. *Biophys. J.*, **2015**, *109*, 1528–1532.

[942] A. Patriksson; D. van der Spoel. A temperature predictor for parallel tempering simulations. *Phys. Chem. Chem. Phys.*, **2008**, *10*, 2073–2077.

[943] P. S. Shabane; S. Izadi; A. V. Onufriev. General purpose water model can improve atomistic simulations of intrinsically disordered proteins. *Journal of Chemical Theory and Computation*, **2019**, *15*, 2620–2634.

[944] D. Pantoja-Uceda; J. L. Neira; L. M. Contreras; C. A. Manton; D. R. Welch; B. Rizzuti. The isolated C-terminal nuclear localization sequence of the breast cancer metastasis suppressor 1 is disordered. *Arch. Biochem. Biophys.*, **2019**, *664*, 95 – 101.

[945] D. Dans; D. Gallego; A. Balaceanu; L. Darre; H. Gomez; M. Orozco. Modeling, simulations, and bioinformatics at the service of rna structure. *Chem*, **2019**, *5*, 51 – 73.

[946] P. Kuhrova; V. Mlynsky; M. Zgarbova; M. Krepl; G. Bussi; R. B. Best; M. Otyepka; J. Sponer; P. Banas. Improving the performance of the Amber RNA force field by tuning the hydrogen-bonding interactions. *bioRxiv*, **2019**.

[947] A. Bochicchio; M. Krepl; F. Yang; G. Varani; J. Sponer; P. Carloni. Molecular basis for the increased affinity of an RNA recognition motif with re-engineered specificity: A molecular dynamics and enhanced sampling simulations study. *PLOS Computat. Biol.*, **2018**, *14*, 1–27.

[948] N. M. Kumbhar; J. S. Gopal. Structural significance of hypermodified nucleoside 5-carboxymethylaminomethyluridine (cmnm5U) from wobble (34th) position of mitochondrial tRNAs: Molecular modeling and Markov state model studies. *J. Molec. Graph. Model.*, **2019**, *86*, 66 – 83.

[949] F. Leonarski; M. Jasinski; J. Trylska. Thermodynamics of the fourU RNA thermal switch derived from molecular dynamics simulations and spectroscopic techniques. *Biochimie*, **2019**, *156*, 22 – 32.

[950] M. Havrila; M. Otyepka; P. Stadlbauer; J. Sponer; J. L. Mergny; P. Banas; P. Kuhrova. Structural dynamics of propeller loop: towards folding of RNA G-quadruplex. *Nucl. Acids Res.*, **2018**, *46*, 8754–8771.

[951] C. Yang; M. Kulkarni; M. Lim; Y. Pak. Insilico direct folding of thrombin-binding aptamer G-quadruplex at all-atom level. *Nucl. Acids Res.*, **2017**, *45*, 12648–12656.

[952] M. Javanainen; A. Lamberg; L. Cwiklik; I. Vattulainen; O. H. S. Ollila. Atomistic model for nearly quantitative simulations of langmuir monolayers. *Langmuir*, **2018**, *34*, 2565–2572. PMID: 28945973.

[953] O. H. S. Ollila; H. A. Heikkinen; H. IwaÃ¯. Rotational dynamics of proteins from spin relaxation times and molecular dynamics simulations. *The Journal of Physical Chemistry B*, **2018**, *122*, 6559–6569. PMID: 29812937.

[954] M. Kulkarni; C. Yang; Y. Pak. Refined alkali metal ion parameters for the opc water model. *Bulletin of the Korean Chemical Society*, **2018**, *39*, 931–935.

[955] G. M. Giambasu; M. K. Gebala; M. T. Panteva; T. Luchko; D. A. Case; D. M. York. Competitive Interaction of Monovalent Cations with DNA from 3D-RISM. *Nucleic Acids Res.*, **2015**, *43*, 8405–8415.

[956] G. M. Giambasu; D. A. Case; D. M. York. Predicting Site-Binding Modes of Ions and Water to Nucleic Acids Using Molecular Solvation Theory. *J. Am. Chem. Soc.*, **2019**, *141*, 2435–2445.

[957] C. Nguyen; T. Yamazaki; A. Kovalenko; D. A. Case; M. K. Gilson; T. Kurtzman; T. Luchko. A molecular reconstruction approach to site-based 3D-RISM and comparison to GIST hydration thermodynamic maps in an enzyme active site. *PLoS One*, **2019**, *14*, e0219743.

[958] M. R. Machado; E. E. Barrera; F. Klein; M. Sonora; S. Silva; S. Pantano. The SIRAH 2.0 Force Field: Altius, Fortius, Citius. *J. Chem. Theory Comput.*, **2019**, *15*, 2719–2733. PMID: 30810317.

[959] P. D. Dans; A. Zeida; M. R. Machado; S. Pantano. A coarse grained model for atomic-detailed dna simulations with explicit electrostatics. *J. Chem. Theory Comput.*, **2010**, *6*, 1711–1725. PMID: 26615701.

[960] E. E. Barrera; M. R. Machado; S. Pantano. Fat sirah: Coarse-grained phospholipids to explore membrane-protein dynamics. *J. Chem. Theory Comput.*, **2019**, *15*, 5674–5688. PMID: 31433946.

[961] P. G. Garay; E. E. Barrera; S. Pantano. Post-translational modifications at the coarse-grained level with the sirah force field. *J. Chem. Inform. Model.*, **2020**, *60*, 964–973. PMID: 31840995.

[962] F. Klein; D. Cáceres; M. A. Carrasco; J. C. Tapia; J. Caballero; J. Alzate-Morales; S. Pantano. Coarse-Grained Parameters for Divalent Cations within the SIRAH Force Field. *J. Chem. Inf. Model.*, **2020**, *60*, 3935–3943.

[963] L. Darré; M. R. Machado; P. D. Dans; F. E. Herrera; S. Pantano. Another coarse grain model for aqueous solvation: Wat four? *J. Chem. Theory Comput.*, **2010**, *6*, 3793–3807.

[964] M. R. Machado; S. Pantano. SIRAH tools: mapping, backmapping and visualization of coarse-grained models. *Bioinformatics*, **2016**, *32*, 1568–1570.

[965] A. Zeida; M. R. Machado; P. D. Dans; S. Pantano. Breathing, bubbling, and bending: DNA flexibility from multimicrosecond simulations. *Phys. Rev. E*, **2012**, *86*, 021903.

[966] M. R. Machado; H. C. González; S. Pantano. Md simulations of viruslike particles with supra cg solvation affordable to desktop computers. *J. Chem. Theory Comput.*, **2017**, *13*, 5106–5116. PMID: 28876928.

[967] H. C. Gonzalez; L. Darré; S. Pantano. Transferable mixing of atomistic and coarse-grained water models. *J. Phys. Chem. B*, **2013**, *117*, 14438–14448. PMID: 24219057.

[968] M. R. Machado; P. D. Dans; S. Pantano. A hybrid all-atom/coarse grain model for multiscale simulations of dna. *Phys. Chem. Chem. Phys.*, **2011**, *13*, 18134–18144.

[969] M. R. Machado; S. Pantano. Exploring LacI–DNA Dynamics by Multiscale Simulations Using the SIRAH Force Field. *J. Chem. Theory Comput.*, **2015**, *11*, 5012–5023. PMID: 26574286.

[970] M. R. Machado; A. Zeida; L. Darré; S. Pantano. From quantum to subcellular scales: multi-scale simulation approaches and the sirah force field. *Interface Focus*, **2019**, *9*.

[971] R. Read; A. McCoy. A log-likelihood-gain intensity target for crystallographic phasing that accounts for experimental error. *Acta Cryst. D*, **2016**, *72*, 375–387.

[972] S. Meisburger; D. Case; N. Ando. Correlated motions in a protein crystal. *Nature Commun.*, **2020**, *11*, 1271.

[973] S. Meisburger; N. Ando. Correlated Motions from Crystallography beyond Diffraction. *Acc. Chem. Res.*, **2017**, *50*, 580–583.

[974] S. Boresch; M. Karplus. The role of bonded terms in free energy simulations. 1. theoretical analysis. *J. Phys. Chem. A*, **1999**, *103*, 103–118.

[975] S. Boresch; M. Karplus. The role of bonded terms in free energy simulations. 2. calculation of their influence on free energy differences of solvation. *J. Phys. Chem. A*, **1999**, *103*, 119–136.

[976] S. Boresch. The role of bonded energy terms in free energy simulations - insights from analytical results. *Mol. Simul.*, **2002**, *28*, 13–37.

[977] T.-S. Lee; Y. Hu; B. Sherborne; Z. Guo; D. M. York. Toward fast and accurate binding affinity prediction with pmemdgti: An efficient implementation of GPU-accelerated thermodynamic integration. *J. Chem. Theory Comput.*, **2017**, *13*, 3077–3084.

[978] T.-S. Lee; D. S. Cerutti; D. Mermelstein; C. Lin; S. LeGrand; T. J. Giese; A. Roitberg; D. A. Case; R. C. Walker; D. M. York. Gpu-accelerated molecular dynamics and free energy methods in amber18: Performance enhancements and new features. *J. Chem. Inf. Model.*, **2018**, *58*, 2043–2050.

[979] L. F. Song; T.-S. Lee; C. Zhu; D. M. York; K. M. Merz Jr. Using AMBER18 for Relative Free Energy Calculations. *J. Chem. Inf. Model.*, **2019**, *59*, 3128–3135.

[980] T. J. Giese; D. M. York. A GPU-Accelerated Parameter Interpolation Thermodynamic Integration Free Energy Method. *J. Chem. Theory Comput.*, **2018**, *14*, 1564–1582.

[981] D. Tan; S. Piana; R. Dirks; D. Shaw. RNA force field with accuracy comparable to state-of- the-art protein force fields. *Proc. Natl. Acad. Sci. USA*, **2018**, *115*, E1346–E1355.

[982] C. Tian; K. Kasavajhala; K. Belfon; L. Raguette; H. Huang; A. Migues; J. Bickel; Y. Wang; J. Pincay; Q. Wu; C. Simmerling. ff19SB: Amino-Acid-Specific Protein Backbone Parameters Trained against Quantum Mechanics Energy Surfaces in Solution. *J. Chem. Theory Comput.*, **2020**, *16*, 528–552.

[983] L. Raguette; A. Cuomo; K. Belfon; C. Tian; Q. Wu; C. Simmerling. Updated Amber force field parameters for phosphorylated amino acids for ff14SB and ff19SB. *In Prep*, **2020**.

[984] K. Belfon; C. Tian; J. Maier; L. Raguette; Q. Wu; C. Simmerling. RAGTAG: Rapid Amber Gpu Torsion pArameter Generator. . *In Prep*, **2020**.

[985] K. Belfon; L. Raguette; Q. Wu; C. Simmerling. Application of RAGTAG: modified amino acids for comparing MD simulations with FRET/EPR experiments. *In Prep*, **2020**.

[986] A. V. Onufriev; S. Izadi. Water models for biomolecular simulations. *WIREs Computational Molecular Science*, **2018**, *8*, e1347.

[987] V. N. Uversky; C. J. Oldfield; A. K. Dunker. Intrinsically disordered proteins in human diseases: Introducing the d2 concept. *Annual Review of Biophysics*, **2008**, *37*, 215–246.

[988] S. Piana; A. G. Donchev; P. Robustelli; D. E. Shaw. Water dispersion interactions strongly influence simulated structural properties of disordered protein states. *The Journal of Physical Chemistry B*, **2015**, *119*, 5113–5123.

[989] V. T. Duong; Z. Chen; M. T. Thapa; R. Luo. Computational studies of intrinsically disordered proteins. *The Journal of Physical Chemistry B*, **2018**, *122*, 10455–10469.

[990] D. Song; W. Wang; W. Ye; D. Ji; R. Luo; H.-F. Chen. ff14idps force field improving the conformation sampling of intrinsically disordered proteins. *Chemical Biology & Drug Design*, **2017**, *89*, 5–15.

[991] W. Ye; D. Ji; W. Wang; R. Luo; H.-F. Chen. Test and evaluation of ff99idps force field for intrinsically disordered proteins. *Journal of Chemical Information and Modeling*, **2015**, *55*, 1021–1029.

[992] J. Huang; S. Rauscher; G. Nawrocki; T. Ran; M. Feig; B. L. de Groot; H. Grubmuller; A. D. MacKerell Jr. Charmm36m: an improved force field for folded and intrinsically disordered proteins. *Nat Meth*, **2017**, *14*, 71–73.

[993] F. Meng; M. M. Bellaiche; J.-Y. Kim; G. H. Zerze; R. B. Best; H. S. Chung. Highly disordered amyloid-β monomer probes by single-molecule fret and md simulation. *Biophysical Journal*, **2018**, *114*, 870–884.

[994] E. Wang; H. Sun; J. Wang; Z. Wang; H. Liu; J. Zhang; T. Hou. End-Point Binding Free Energy Calculation with MM/PBSA and MM/GBSA: Strategies and Applications in Drug Design. *Chem. Rev.*, **2019**, *119*, 9478–9508.

[995] E. W. Weisstein. Euler angles From MathWorld—A Wolfram Web Resource.

[996] D. C. Rapaport. *The Art of Molecular Dynamics Simulation*. Cambridge University Press, 2010.

[997] M. Machado; S. Pantano. Split the Charge Difference in Two! A Rule of Thumb for Adding Proper Amounts of Ions in MD Simulations. *J. Chem. Theory Comput.*, **2020**, *16*, 1367–1372.

[998] J. Schmit; N. Kariyawasam; V. Needham; P. Smith. SLTCAP: A Simple Method for Calculating the Number of Ions Needed for MD Simulation. *J. Chem. Theory Comput.*, **2018**, *14*, 1823–1827.

[999] C. Gebhardt; M. Lehmann; M. M. Reif; M. Zacharias; T. Cordes. Molecular and spectroscopic characterization of green and red cyanine fluorophores from the alexa fluor and af series. *bioRxiv*, **2020**.

[1000] M. E. O'Neill. PCG: A Family of Simple Fast Space-Efficient Statistically Good Algorithms for Random Number Generation. Technical Report HMC-CS-2014-0905, Harvey Mudd College, Claremont, CA, 2014.

[1001] S. Vigna. Further scramblings of Marsaglia's xorshift generators. *J. Comput. Appl. Math.*, **2017**, *315*, 175–181.

[1002] D. R. Roe; B. R. Brooks. A protocol for preparing explicitly solvated systems for stable molecular dynamics simulations. *J. Chem. Phys.*, **2020**, *153*, 054123.

[1003] D. R. Roe; B. R. Brooks. Improving the Speed of Volumetric Density Map Generation via Cubic Spline Interpolation. *J. Mol. Graphics Model.*, **2021**, *104*, 107832.

[1004] V. W. D. Cruzeiro; G. T. Feliciano; A. E. Roitberg. Exploring Coupled Redox and pH Processes with a Force-Field-Based Approach: Applications to Five Different Systems. *J. Am. Chem. Soc.*, **2020**, *142*, 3823–3835.

[1005] H. M. Aktulga; J. C. Fogarty; S. A. Pandit; A. Y. Grama. Parallel reactive molecular dynamics: Numerical methods and algorithmic techniques. *Parallel Computing*, **2012**, *38*, 245–259.

[1006] S. B. Kylasa; H. M. Aktulga; A. Y. Grama. Puremd-gpu: A reactive molecular dynamics simulation package for gpus. *Journal of Computational Physics*, **2014**, *272*, 343–359.

[1007] A. C. Van Duin; S. Dasgupta; F. Lorant; W. A. Goddard. Reaxff: a reactive force field for hydrocarbons. *The Journal of Physical Chemistry A*, **2001**, *105*, 9396–9409.

[1008] A. K. Rappe; W. A. Goddard III. Charge equilibration for molecular dynamics simulations. *The Journal of Physical Chemistry*, **1991**, *95*, 3358–3363.

[1009] W. J. Mortier; S. K. Ghosh; S. Shankar. Electronegativity-equalization method for the calculation of atomic charges in molecules. *Journal of the American Chemical Society*, **1986**, *108*, 4315–4320.

[1010] M. Samieegohar; F. Sha; A. Z. Clayborne; T. Wei. Reaxff md simulations of peptide-grafted gold nanoparticles. *Langmuir*, **2019**, *35*, 5029–5036.

[1011] S. Yang; T. Zhao; L. Zou; X. Wang; Y. Zhang. Reaxff-based molecular dynamics simulation of dna molecules destruction in cancer cells by plasma ros. *Physics of Plasmas*, **2019**, *26*, 083504.

[1012] P. O. Hubin; D. Jacquemin; L. Leherte; D. P. Vercauteren. Parameterization of the reaxff reactive force field for a proline-catalyzed aldol reaction. *Journal of computational chemistry*, **2016**, *37*, 2564–2572.

[1013] S. Monti; J. Jose; A. Sahajan; N. Kalarikkal; S. Thomas. Structure and dynamics of gold nanoparticles decorated with chitosan–gentamicin conjugates: Reaxff molecular dynamics simulations to disclose drug delivery. *Physical Chemistry Chemical Physics*, **2019**, *21*, 13099–13108.

[1014] T. Trnka; I. Tvaroska; J. Koca. Automated training of reaxff reactive force fields for energetics of enzymatic reactions. *Journal of chemical theory and computation*, **2018**, *14*, 291–302.

[1015] A. Rahnamoun; A. Van Duin. Reactive molecular dynamics simulation on the disintegration of kapton, poss polyimide, amorphous silica, and teflon during atomic oxygen impact using the reaxff reactive force-field method. *The Journal of Physical Chemistry A*, **2014**, *118*, 2780–2787.

[1016] D. Raymand; A. C. van Duin; M. Baudin; K. Hermansson. A reactive force field (reaxff) for zinc oxide. *Surface science*, **2008**, *602*, 1020–1031.

[1017] K. Chenoweth; S. Cheung; A. C. Van Duin; W. A. Goddard; E. M. Kober. Simulations on the thermal decomposition of a poly (dimethylsiloxane) polymer using the reaxff reactive force field. *Journal Of The American Chemical Society*, **2005**, *127*, 7192–7202.

[1018] M. F. Russo Jr; R. Li; M. Mench; A. C. Van Duin. Molecular dynamic simulation of aluminum–water reactions using the reaxff reactive force field. *International Journal of Hydrogen Energy*, **2011**, *36*, 5828–5835.

[1019] J. Ludwig; D. G. Vlachos; A. C. Van Duin; W. A. Goddard. Dynamics of the dissociation of hydrogen on stepped platinum surfaces using the reaxff reactive force field. *The Journal of Physical Chemistry B*, **2006**, *110*, 4274–4282.

[1020] K. Yoon; A. Rahnamoun; J. L. Swett; V. Iberi; D. A. Cullen; I. V. Vlassiouk; A. Belianinov; S. Jesse; X. Sang; O. S. Ovchinnikova et al. Atomistic-scale simulations of defect formation in graphene under noble gas ion irradiation. *ACS nano*, **2016**, *10*, 8376–8384.

[1021] A. Rahnamoun; A. Van Duin. Study of thermal conductivity of ice clusters after impact deposition on the silica surfaces using the reaxff reactive force field. *Physical Chemistry Chemical Physics*, **2016**, *18*, 1587–1594.

[1022] C. Zou; A. Van Duin. Investigation of complex iron surface catalytic chemistry using the reaxff reactive force field method. *Jom*, **2012**, *64*, 1426–1437.

[1023] Y. K. Shin; H. Kwak; A. V. Vasenkov; D. Sengupta; A. C. van Duin. Development of a reaxff reactive force field for fe/cr/o/s and application to oxidation of butane over a pyrite-covered cr2o3 catalyst. *Acs Catalysis*, **2015**, *5*, 7226–7236.

[1024] T. P. Senftle; S. Hong; M. M. Islam; S. B. Kylasa; Y. Zheng; Y. K. Shin; C. Junkermeier; R. Engel-Herbert; M. J. Janik; H. M. Aktulga et al. The reaxff reactive force-field: development, applications and future directions. *npj Computational Materials*, **2016**, *2*, 1–14.

[1025] L. Chen; A. Cruz; D. R. Roe; A. C. Simmonett; L. Wickstrom; N. Deng; T. Kurtzman. Thermodynamic Decomposition of Solvation Free Energies with Particle Mesh Ewald and Long-Range Lennard-Jones Interactions in Grid Inhomogeneous Solvation Theory. *J. Chem. Theory Comput.*, **2021**, *17*, 2714–2724.

Index