

# Msander Reference Manual

Msander (“modern sander”) is an updated version of *sander*, the basic energy minimizer and molecular dynamics program in the *AmberTools suite*. 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.

## Principal contributors to the current codes:

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



# Contents

<b>Contents</b>	<b>3</b>
<b>1 msander</b>	<b>5</b>
1.1 Introduction	5
1.2 Main changes from the <i>sander</i> code in AmberTools	6
1.3 Installation	7
1.4 File usage	7
1.5 Example input files	7
1.6 Namelist Input Syntax	8
1.7 Overview of the information in the input file	9
1.8 General minimization and dynamics parameters	9
1.9 Potential function parameters	17
1.10 Varying conditions	22
1.11 File redirection commands	26
1.12 Getting debugging information	27
1.13 multisander	29
<b>2 The Generalized Born/Surface Area Model</b>	<b>31</b>
2.1 GB/SA input parameters	33
<b>3 Reference Interaction Site Model</b>	<b>37</b>
3.1 Introduction	37
3.2 Practical Considerations	40
3.3 3D-RISM in sander	42
<b>4 sqm: Semi-empirical quantum chemistry</b>	<b>49</b>
4.1 Available Hamiltonians	49
4.2 Dispersion and hydrogen bond correction	51
4.3 Usage	52
<b>5 QM/MM calculations</b>	<b>59</b>
5.1 Built-in semiempirical NDDO methods and SCC-DFTB	59
5.2 Interface for <i>ab initio</i> and DFT methods	68
5.3 QM/MM simulations with QUICK	85
<b>6 Atom and Residue Selections</b>	<b>89</b>
6.1 Amber Masks	89
6.2 GROUP Specification	92
<b>7 Sampling configuration space</b>	<b>97</b>
7.1 Self-Guided Langevin dynamics	97
7.2 Targeted MD	100
7.3 Multiply-Targeted MD (MTMD)	101
7.4 Low-MODE (LMOD) methods	102

## CONTENTS

<b>8</b>	<b>Free energies</b>	<b>107</b>
8.1	Thermodynamic integration . . . . .	107
8.2	Linear Interaction Energies . . . . .	113
8.3	Adaptively Biased MD, Steered MD, Umbrella Sampling with REMD and String Method . . . . .	113
<b>9</b>	<b>NMR refinement</b>	<b>137</b>
9.1	Distance, angle and torsional restraints . . . . .	138
9.2	NOESY volume restraints . . . . .	143
9.3	Chemical shift restraints . . . . .	145
9.4	Pseudocontact shift restraints . . . . .	146
9.5	Direct dipolar coupling restraints . . . . .	147
9.6	Residual CSA or pseudo-CSA restraints . . . . .	149
9.7	Preparing restraint files for Sander . . . . .	150
9.8	Getting summaries of NMR violations . . . . .	156
9.9	Time-averaged restraints . . . . .	156
9.10	Multiple copies refinement using LES . . . . .	157
9.11	Some sample input files . . . . .	158
<b>10</b>	<b>Xray and cryoEM refinement</b>	<b>163</b>
10.1	EMAP restraints for rigid and flexible fitting into EM maps . . . . .	163
10.2	Setting up crystal simulations . . . . .	164
10.3	X-ray functionality and diffraction-based restraints . . . . .	167
10.4	Preparing inputs X-ray refinement . . . . .	170
	<b>Bibliography</b>	<b>171</b>
	<b>Index</b>	<b>227</b>

# 1 msander

## 1.1 Introduction

This is a guide to *msander*, 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 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:

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

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 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. Translational periodic (P1) symmetry is supported, but no other space-group symmetry is. The size of the unit cell 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:

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

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. Xray and cryoEM refinements can be carried out in reciprocal space, with GPU-accelerated structure factor calculations. Real-space electron density maps can also be used as restraints for refinement.
9. 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.
10. 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.
11. 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.
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.

## 1.2 Main changes from the *sander* code in AmberTools

1. Some pieces are missing from the *sander* program in AmberTools:
  - a) Things that should be easy to re-introduce later: *emil*, *sebond*, *pbsa*, *APBS*, the charge-relocation option.
  - b) Things that are probably gone for good, but which don't represent the best current practice: Path-integral methods, thermostats that don't follow the "middle" scheme, Berendsen barostat
  - c) Things that might be useful, but really complicate the code: *evb* potentials, some parts of adaptive QM/MM, nudged elastic band, constant pH and constant redox potential simulations.
  - d) Non-periodic 3D-RISM has been removed for now, in an attempt to get the simplest possible RISM code, perhaps as a basis for future GPU work.
2. Key pieces of code that are still there, and being emphasized:
  - a) \* Periodic and non-periodic simulations, with all of Amber's GB models
  - b) \* 3D-RISM in periodic boundary conditions
  - c) \* QM/MM, including hooks to external codes
  - d) \* NMR, cryoEM and Xray restraints (including quite a bit of new code; Xray restraints include NVIDIA GPU-enabled capabilities)
  - e) \* Thermodynamic integration and non-equilibrium sampling methods
  - f) \* Replica exchange capabilities, except for constant pH and redox potential simulations

## 1.3 Installation

```
./configure --help    # then choose options
make install
make test             # optional--more info is below
```

The test suite also serves as a source of example input files. Look especially in the *rism*, *xray* and *cryoem* folders to see sample inputs for these sorts of calculation.

## 1.4 File usage

```
msander [-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
-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

**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)

## 1.5 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)
```

## 1.6 Namelist Input Syntax

Namelist provides list-directed input, and convenient specification of default values, and is a part of the Fortran 90 standard, 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.

## 1.7 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 1.8 and 1.9.

**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 1.10.

**File redirection** TYPE=*filename* lines. Section ends with the first non-blank line which does not correspond to a recognized redirection. Described in Section 1.11.

**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 6.2.

## 1.8 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 commonly modified, 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.

### 1.8.1 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 *ntr*), 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 used to 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.

### 1.8.2 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.

### 1.8.3 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.

**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  $\text{abs}(ntwr)$  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.

**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 combined 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

**ntwpri** The number of atoms to include in trajectory files (mdcrd and mdvel). This flag can be used to decrease the size of 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  $ntwpri = 0$ , all atoms will be included.

= 0 (default) Include all atoms of the system when writing trajectories.

> 0 Include only atoms 1 to  $ntwpri$  when writing trajectories.

#### 1.8.4 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 ( $\text{kcal} \cdot \text{mol}^{-1} \cdot \text{\AA}^{-2}$ ) for the positional restraints. The restraint is of the form  $k(\Delta x)^2$ , where  $k$  is the value given by this variable, and  $\Delta x$  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 restrained 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 6.1.1. Note that these mask strings are limited to a maximum of 256 characters.

### 1.8.5 Energy minimization

- maxcyc** The maximum number of cycles of minimization. Default = 1.
- ntmin** Flag for the method of minimization.  
**= 3** The XMIN method is used, see Section 7.4.1. (default)  
**= 4** The LMOD method is used, see Section 7.4.2.
- 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  $10^{-4} \text{kcal} \cdot \text{mol}^{-1} \cdot \text{\AA}^{-1}$ .

### 1.8.6 Molecular dynamics

The “middle” scheme offers a unified framework to develop efficient thermostatting algorithms for configurational sampling for the canonical ensemble, as described in Refs. [1–5]. It is the only MD scheme implemented in *msander*. 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)  $\Delta t$  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 [1, 5].

- 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 [6] 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.

### 1.8.7 Temperature regulation

Note: Flag "ntt" is used for the temperature regulation in the default thermostat scheme as shown below. The "middle" thermostat scheme [Section ??] 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 ?? 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.[7, 8] 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).
- = 3** Use Langevin dynamics with the collision frequency  $\gamma$  given by *gamma\_In*, discussed below. Note that when  $\gamma$  has its default value of zero, this is the same as setting *ntt* = 0. Since Langevin simulations are highly susceptible to "synchronization" artifacts,[9, 10] you should explicitly set the *ig* variable (described below) to a different value at each restart of a given simulation. (Setting *ig* = -1 will randomize this starting seed, based on the system clock.)
- 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.

<code>tempi</code>	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.
<code>ig</code>	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.
<code>gamma_ln</code>	<p>The collision frequency <math>\gamma</math>, in <math>\text{ps}^{-1}</math>, when <math>ntt = 3</math>. Default is 0. A. See Section ?? for details. Note that it is not necessary that <math>\gamma</math> approximate the physical collision frequency, which is about <math>50 \text{ ps}^{-1}</math> for liquid water. In fact, it is often advantageous, in terms of sampling[11, 12] or stability of integration[13], to use much smaller values, around 2 to <math>5 \text{ ps}^{-1}</math>. [11, 13] For implicit solvent (GB), even much lower values may be useful: for example, setting <math>\text{gamma\_ln}</math> to <math>0.01 \text{ ps}^{-1}</math> can lead to significant, up to 100-fold in some cases, speedup of conformational sampling.[14]</p> <p>The recommended value for <math>\text{gamma\_ln}</math> is related to the characteristic frequency (<math>\tilde{\omega}</math>) of the specific system. The characteristic time of the potential energy autocorrelation function is</p> $\tau_{UU} = \int_0^\infty \frac{\langle U(0)U(t) \rangle - \langle U \rangle^2}{\langle U^2 \rangle - \langle U \rangle^2} dt \quad (1.1)$ <p>The optimal value of the thermostat parameter that produces the minimum correlation time of the potential is <math>\xi^{opt} \approx \tilde{\omega}</math> for Langevin dynamics and <math>\xi^{opt} \approx \sqrt{2}\tilde{\omega}</math> for the Andersen thermostat, as the time interval <math>\Delta t</math> approaches zero. E.g. for a HO molecule, the frequency of the O-H stretch is around <math>3600 \text{ cm}^{-1}</math> (<math>680 \text{ ps}^{-1}</math>), so one can choose <math>680 \text{ ps}^{-1}</math> as the value of <math>\text{therm\_par}</math> when Langevin dynamics is used, or <math>960 \text{ ps}^{-1}</math> when the Andersen thermostat is employed. When the time interval <math>\Delta t</math> 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. [3–5] for more discussion.) 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 [15]. 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 <math>2 - 50 \text{ ps}^{-1}</math>.</p>
<code>vlimit</code>	If not equal to 0.0, then any component of the velocity that is greater than $\text{abs}(\text{VLIMIT})$ will be reduced to $\text{VLIMIT}$ (preserving the sign). This can be used to avoid occasional instabilities in molecular dynamics runs. $\text{VLIMIT}$ 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.

## 1.8.8 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,  $\text{pres0}$ . Equilibration with  $\text{ntp} > 0$  is generally necessary to adjust the density of the system to appropriate values. Pressure regulation only applies when Constant Pressure

periodic boundary conditions are used ( $ntp > 0$ ). The Monte Carlo barostat 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.

<b>ntp</b>	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
<b>barostat</b>	Flag used to control which barostat to use in order to control the pressure. = 2 Monte Carlo barostat
<b>mcbarint</b>	Number of steps between volume change attempts performed as part of the Monte Carlo barostat. Default is 100.
<b>pres0</b>	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.

### 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.[16] 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.

<b>csurften</b>	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
<b>gamma_ten</b>	Surface tension value in units of dyne/cm. Default value is 0.0 dyne/cm.
<b>ninterface</b>	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.

### 1.8.9 SHAKE bond length constraints

<b>ntc</b>	Flag for SHAKE to perform bond length constraints.[17] (See also NTF in the <b>Potential function</b> 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.[18] 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.[18]

= 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.)

### 1.8.10 NMR refinement options

(Users 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  $\text{mod}(\text{nstep}, \text{noeskp}) = 0$ ; 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 functional form of the penalty function for NOESY volume and chemical shift restraints.



	<p><b>= 1</b> the program will minimize the sum of the absolute values of the errors; this is akin to minimizing the crystallographic R-factor (default).</p> <p><b>= 2</b> the program will optimize the sum of the squares of the errors.</p> <p><b>= 3</b> For NOESY intensities, the penalty will be of the form <math>awt[I_c^{1/6} - I_o^{1/6}]^2</math>. Chemical shift penalties will be as for <i>ipnlty=1</i>.</p>
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.
scaln	"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 ( <i>NMROPT</i> = 2), intensities with mixing times less than 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.

### 1.8.11 EMAP restraints

EMAP restraints are used to perform targeted conformational search (TCS)[19]. 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 10.1 to see how to define EMAP restraints.

iemap	Turn on EMAP restrained simulation when <i>iemap</i> >0. (Default = 0). EMAP restraint information must be input from <i>&amp;emap</i> namelists in the input file.
gammamap	Friction constant for the EMAP restraint maps when allowed to move. (Default=1/ps). (See Section 10.1)

## 1.9 Potential function parameters

The parameters in this section generally control what sort of force field (or potential function) is used for the simulation.

### 1.9.1 Generic parameters

ntf	Force evaluation. Note: If SHAKE is used (see NTC), it is not necessary to calculate forces for the constrained bonds.
	<p><b>= 1</b> complete interaction is calculated (default)</p> <p><b>= 2</b> bond interactions involving H-atoms omitted (use with NTC=2)</p> <p><b>= 3</b> all the bond interactions are omitted (use with NTC=3)</p> <p><b>= 4</b> angle involving H-atoms and all bonds are omitted</p>

- = 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.

**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.

- irism** Flag for 3D-reference interaction site model (RISM) molecular solvation method. See Section 3.3 for 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 [20] 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!.

## 1.9.2 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.[21–24] Generalizations of this method to systems with polarizable dipoles and electrostatic multipoles are described in Refs. [25, 26].

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 choose 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.
- 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  $5 \times 10^{-5}$ .

<code>mlimit(1,2,3)</code>	This allows the user to explicitly set the number of reciprocal vectors used in a regular Ewald run. Note that the sum goes from $-\text{MLIMIT}(2)$ to $\text{MLIMIT}(2)$ and $-\text{MLIMIT}(3)$ to $\text{MLIMIT}(3)$ with symmetry being used in first dimension. Note also the sum is truncated outside an automatically chosen sphere.
<code>ew_coeff</code>	Ewald coefficient, in $\text{\AA}^{-1}$ . Default is determined by <code>dsum_tol</code> and <code>cutoff</code> . If it is explicitly inputted then that value is used, and <code>dsum_tol</code> is computed from <code>ew_coeff</code> and <code>cutoff</code> .
<code>nbflag</code>	If <code>nbflag = 0</code> , construct the direct sum nonbonded list in the "old" way, <i>i.e.</i> update the list every <code>nsnb</code> steps. If <code>nbflag = 1</code> (the default when <code>imin = 0</code> or <code>ntb &gt; 0</code> ), <code>nsnb</code> is ignored, and the list is updated whenever any atom has moved more than $1/2 \text{ skinnb}$ since the last list update.
<code>skinnb</code>	Width of the nonbonded "skin". The direct sum nonbonded list is extended to <code>cut + skinnb</code> , and the van der Waals and direct electrostatic interactions are truncated at <code>cut</code> . Default is $2.0 \text{ \AA}$ . Use of this parameter is required for energy conservation, and recommended for all PME runs.
<code>netfrc</code>	The basic "smooth" PME implementation used here does not necessarily conserve momentum. If <code>netfrc = 1</code> , (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.
<code>vdwmeth</code>	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.
<code>eedmeth</code>	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.
<code>eedtb dns</code>	Density of spline or linear lookup table, if <code>eedmeth</code> is 1 or 2. Default is 500 points per unit.
<code>column_fft</code>	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).

### 1.9.3 Using IPS for the calculation of nonbonded interactions

Isotropic Periodic Sum (IPS) is a method for long-range interaction calculation.[27–32] 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. [31] In addition, the 3D IPS/DFFT algorithm [30] 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  $10 \text{ \AA}$ . 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.

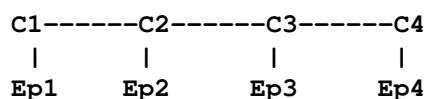
<i>ips</i>	<p>Flag to control nonbonded interaction calculation method. The <i>cut</i> value will be used to define the local region radius for <math>ips \leq 3</math>. When IPS is used for electrostatic interaction, PME will be turned off. When using the amoeba polarizable potentials, <i>iamoeba</i>=1, <math>ips &gt; 0</math> (same as <math>ips=2</math>) will turn on 3D IPS for all charge, dipole, and quadrupole interactions and the <i>ee_dsum_cut</i> value will be used to define the local region radius.</p> <p>= 0 IPS will not be used (default).</p> <p>= 1 3D IPS will be used for both electrostatic and L-J interactions.</p> <p>= 2 3D IPS will be used only for electrostatic, including all multipole, interactions.</p> <p>= 3 3D IPS will be used only for L-J interactions.</p> <p>= 4 3D IPS/DFFT will be used for both electrostatic and L-J interactions.</p> <p>= 5 3D IPS/DFFT will be used only for electrostatic interactions.</p> <p>= 6 3D IPS/DFFT will be used only for L-J interactions.</p>
<i>raips</i>	<p>Local region radius. <i>raips</i> is automatically set to <i>cut</i> for 3D IPS calculations (<math>ips \leq 3</math>) and should be set larger than <i>cut</i> for 3D IPS/DFFT calculations (<math>ips \geq 4</math>). 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, <i>raips</i>=<math>\infty</math>, which corresponding no image interaction. The default value is -1 Å.</p>
<i>mipsex, mipsey, mipsz</i>	<p>Number of grids along three periodic boundary sides when using 3D IPS/DFFT method (<math>ips \geq 4</math>). Negative values indicate they are calculated based on the grid size, <i>gridips</i>. Typical numbers are the lengths of box sides (in Å) divided by 2 Å. Default values are -1. When <math>ips=6</math> and PME is used for electrostatic interaction, they are set to <i>nfft1</i>, <i>nfft2</i>, and <i>nfft3</i> defined for PME, respectively.</p>
<i>mipso</i>	<p>The order of the B-spline interpolation (<math>ips \geq 4</math>). 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 <math>ips=6</math> and PME is used to electrostatic interaction, it is set to <i>order</i> defined for PME.</p>
<i>gridips</i>	<p>Grid size for 3D IPS/DFFT calculation (<math>ips \geq 4</math>). The default value is 2 Å.</p>
<i>dvbips</i>	<p>Volume tolerance for updating IPS function grids (<math>ips \geq 4</math>). When volume changes like in <i>NPT</i> 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 <i>dvbips</i>. The default value is <math>1 \times 10^{-8}</math>.</p>

### 1.9.4 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.

<i>frameon</i>	<p>If <i>frameon</i> 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 <i>frameon</i>=0.</p>
----------------	--

**chnngmask** If *chnngmask*=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



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.

## 1.9.5 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*.

## 1.10 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.

$$\text{VALUE2} = (\text{R}^{**}\text{INCREMENTS}) * \text{VALUE1}.$$

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).
XRAY	Varies the xray_weight restraint weight.
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:

$$\text{ISTEP1} \leq \text{ABS}(\text{delta\_residue}) \leq \text{ISTEP2},$$

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 =  $\tau$  (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  $\tau$  at any time after the start of the run will only affect the relative weighting of steps occurring after the change in  $\tau$ . Separate values for  $\tau$  and POWER are used for bond, angle, and torsion averaging.

The default value of  $\tau$  (if it is 0.0 here) is 1.0D+6, which results in no exponential decay weighting. Any value of  $\tau \geq 1.0D+6$  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  $\tau$ .

**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 + VALUE1
VALUE1 <= -1000.: Initial value = r_target + 1000.
1000. <= VALUE1 : Initial value = r_target - 1000.
```

$r_{\text{target}}$  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  $\tau$  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  $\tau$  was used during the simulation. If VALUE2>0, then  $\tau = \text{VALUE2}$  will be used in calculating these final reported averages. Note that the value of VALUE2 =  $\tau$  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  $(dE/dr_{\text{ave}})$  ( $dr_{\text{ave}}/dx$ ). If IINC = 1, then then forces for the class of time-averaged restraints will be calculated as  $(dE/dr_{\text{ave}})$  ( $dr(t)/dx$ ). 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 TEMPO, 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.

## 1.11 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 <i>before</i> the simulation has been run. By default, this listing is not printed. If 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 <i>after</i> 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 9.1) will be read.
NOESY	File from which NOESY volume information (Section 9.2) will be read.
SHIFTS	File from which chemical shift information (Section 9.3) will be read.
PCSHIFT	File from which paramagnetic shift information (Section 9.3) will be read.
DIPOLE	File from which residual dipolar couplings (Section 9.5) will be read.
CSA	File from which CSA or pseduo-CSA restraints (Section 9.6) 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 $\neq$ 0, then averaged values will be output. If the DUMPFREQ command has been used, the instantaneous values will be output.

## 1.12 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 &debug 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 neglgdel 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 reasonably, (forces to within about  $5 \times 10^{-4}$  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.

## 1.13 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 replical.mdin -suffix replical
-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.

## 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.[14, 33–36] 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.[37] To estimate the total solvation free energy of a molecule,  $\Delta G_{solv}$ , one typically assumes that it can be decomposed into the "electrostatic" and "non-electrostatic" parts:

$$\Delta G_{solv} = \Delta G_{el} + \Delta G_{nonel} \quad (2.1)$$

where  $\Delta G_{nonel}$  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  $\Delta G_{el}$  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,  $\Delta G_{nonel}$  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 [38] 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  $\Delta G_{el}$ . 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,[39–46] especially in molecular dynamics applications,[47–50] 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  $\epsilon$  (80 for water at 300 K). The GB model approximates  $\Delta G_{el}$  by an analytical formula,[39, 51]

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

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 [51] via the Debye-Huckel screening parameter  $\kappa$ .

A common choice [39] of  $f_{GB}$  is

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

although other expressions have been tried.[42, 52] 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  $\rho_i$ . Then one obtains the particularly simple form:

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

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  $r_{ij} \rightarrow 0$ , when atomic spheres merge into one, and the opposite extreme  $r_{ij} \rightarrow \infty$ , when the ions can be treated as point charges obeying the Coulomb's law.[45] 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  $\rho_i$ .

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:[45]

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

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,[53] 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,[54] commonly used in the PB calculations. For macromolecules, this approach tends to underestimate the effective radii for buried atoms,[45] 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.[52] 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 [40, 44, 55] 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,[45] 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. [45] was not optimal in keeping the model's established performance on small molecules. It was therefore proposed [50] 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:

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

where  $\Psi = I_i \tilde{\rho}_i$ , and  $\alpha, \beta, \gamma$  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  $\alpha, \beta, \gamma$ , 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. [50]; 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. [56].

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
<i>mbondi</i>	<i>mbondi2</i>	<i>mbondi2</i>	<i>bondi</i>	<i>mbondi3</i>

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 [57, 58] 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  $\alpha$ -helix present.[59, 60] The combination of the *ff14SBonlysc* force field with *igb*=8 gives the best results for proteins [61][62], nucleic acids and protein-nucleic acid complexes. [63]

Despite these limitations, implicit treatment of solvent is widely used in molecular simulations for two main reasons: algorithmic/computational speed and conformational sampling. [14, 64] 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.[14, 33–36] 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. [14] 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 *igb*5 can be found in Ref. [14].

The generalized Born models used here are based on the "pairwise" model introduced by Hawkins, Cramer and Truhlar,[53, 65] which in turn is based on earlier ideas by Still and others.[39, 44, 55, 66] 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.[51] 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. [67, 68].

## 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.[60] 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).[61] 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[53, 65] pairwise generalized Born model is used, with parameters described by Tsui and Case.[67] 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.[48] 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,[45] but the actual implementation here[50] 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:
 
$$R_i^{-1} = \bar{\rho}_i^{-1} - \tanh(\alpha\Psi - \beta\Psi^2 + \gamma\Psi^3) / \rho_i$$
 where  $\bar{\rho}_i = \rho_i - offset$ , and  $\Psi = I\rho_i$ , with *I* given in our earlier paper. The parameters  $\alpha$ ,  $\beta$ , and  $\gamma$  were determined by empirical fits, and have the values 0.8, 0.0, and 2.909125. This corresponds to model I in Ref [50]. 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  $\alpha, \beta, \gamma$  are 1.0, 0.8, and 4.85. This corresponds to model II in Ref [50]. 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,[50] both of the above parameterizations of the "OBC" model showed equal performance, although further tests [56] 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[69] 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.[69] This method is not recommended for 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  $\alpha$ ,  $\beta$ , and  $\gamma$  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.[61] 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.[63]

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  $\alpha$ ,  $\beta$ ,  $\gamma$  set for element X. gbalphaos, gbbetaos, gbgammaos is the  $\alpha$ ,  $\beta$ ,  $\gamma$  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  $\alpha$ ,  $\beta$ ,  $\gamma$  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.

<b>intdiel</b>	Sets the interior dielectric constant of the molecule of interest. Default is 1.0. Other values have not been extensively tested.
<b>extdiel</b>	Sets the exterior or solvent dielectric constant. Default is 78.5.
<b>saltcon</b>	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.[51] Default is 0.0 M ( <i>i.e.</i> no Debye-Hückel screening.) Setting <i>saltcon</i> to a nonzero value does result in some increase in computation time.
<b>rgbmax</b>	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 away than <i>rgbmax</i> 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 <i>rgbmax</i> 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*.

<b>rbornstat</b>	If <i>rbornstat</i> = 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.
<b>offset</b>	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.
<b>gbsa</b>	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 <i>gbsa</i> = 1, surface area will be computed using the LCPO model.[38] If <i>gbsa</i> = 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, <i>gbsa</i> = 2 only works for a single point energy calculation and is mainly intended for energy decomposition in the realm of MM-GBSA. If <i>gbsa</i> = 3, surface area will be computed using a fast pairwise approximation [70] suitable for GPU computing in pmemd.cuda program; the acceleration in pmemd.cuda compared with <i>gbsa</i> = 2 is ~30 times faster [70]. Note that <i>gbsa</i> = 3 is currently not supported in sander, MM-GBSA, QM/MM or libsff. Although <i>gbsa</i> = 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 <i>gbsa</i> =3 with pmemd.cuda.
<b>surften</b>	Surface tension used to calculate the nonpolar contribution to the free energy of solvation (when <i>gbsa</i> = 1), as $Enp = surften * SA$ . The default is 0.005 kcal/mol/Å <sup>2</sup> . [71] For <i>gbsa</i> = 3, <i>surften</i> works comparably with <i>gbsa</i> = 1 given the same value. [70]
<b>rdt</b>	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 ?? 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[72–85]. 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 1.

### 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[73, 82, 83, 86, 87]

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

where  $r_{12}$  is the separation between particles 1 and 2 while  $\Omega_1$  and  $\Omega_2$  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  $\alpha$  of species  $a$  and site  $\gamma$  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.,  $\rho^U \rightarrow 0$ . We can rewrite Equation (3.1), in the limit of infinite dilution, as a set of three equations:

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

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

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

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[86]

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

$u$  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  $h$  [86]). 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[86]

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

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[83].

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[82, 83]

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

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,

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

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

### 3.1.1 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[81, 82]

$$h_{\gamma}^{\text{UV}}(\mathbf{r}) = \sum_{\alpha} \int d\mathbf{r}' c_{\alpha}^{\text{UV}}(\mathbf{r} - \mathbf{r}') \chi_{\alpha\gamma}^{\text{VV}}(r'), \quad (3.9)$$

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

$$\chi_{\alpha\gamma}^{\text{VV}}(r) = \omega_{\alpha\gamma}^{\text{VV}}(r) + \rho_{\alpha} h_{\alpha\gamma}^{\text{VV}}(r).$$

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

$$g_{\gamma}^{\text{HNC,UV}}(\mathbf{r}) = \exp(-\beta u_{\gamma}^{\text{UV}}(\mathbf{r}) + h_{\gamma}^{\text{UV}}(\mathbf{r}) - c_{\gamma}^{\text{UV}}(\mathbf{r})). \quad (3.10)$$



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  $g_H^{UV}(\mathbf{r})$  and  $g_O^{UV}(\mathbf{r})$  grids. Each point on the  $g_H^{UV}(\mathbf{r})$  will give the fractional density of water hydrogen at 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 [83, 84, 88]. 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 [88]. The convolution of the short-range terms in the integral equation (3.9) is calculated using 3D fast Fourier transform [89, 90]. Accordingly, the electrostatic asymptotics terms in the thermodynamics integral (3.12) below are handled analytically and reduced to one-dimensional integrals easy to compute [88].

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),  $\mu^{\text{ex}}$  and the mean solvation force,  $\mathbf{f}_i^{\text{UV}}(\mathbf{R}_i)$ , on each solute atom,  $i$ .  $\mu^{\text{ex}}$  can be obtained through analytical thermodynamic integration for HNC,

$$\mu^{\text{ex,HNC}} = k_B T \sum_{\alpha} \rho_{\alpha}^V \int d\mathbf{r} \left[ \frac{1}{2} (h_{\alpha}^{\text{UV}}(\mathbf{r}))^2 - c_{\alpha}^{\text{UV}}(\mathbf{r}) - \frac{1}{2} h_{\alpha}^{\text{UV}}(\mathbf{r}) c_{\alpha}^{\text{UV}}(\mathbf{r}) \right], \quad (3.11)$$

KH,

$$\mu^{\text{ex,KH}} = k_B T \sum_{\alpha} \rho_{\alpha}^V \int d\mathbf{r} \left[ \frac{1}{2} (h_{\alpha}^{\text{UV}}(\mathbf{r}))^2 \Theta(-h_{\alpha}^{\text{UV}}(\mathbf{r})) - c_{\alpha}^{\text{UV}}(\mathbf{r}) - \frac{1}{2} h_{\alpha}^{\text{UV}}(\mathbf{r}) c_{\alpha}^{\text{UV}}(\mathbf{r}) \right], \quad (3.12)$$

and PSE- $n$ ,

$$\mu^{\text{ex,PSE-}n} = k_B T \sum_{\alpha} \rho_{\alpha}^V \int d\mathbf{r} \left[ \frac{1}{2} (h_{\alpha}^{\text{UV}}(\mathbf{r}))^2 - c_{\alpha}^{\text{UV}}(\mathbf{r}) - \frac{1}{2} h_{\alpha}^{\text{UV}}(\mathbf{r}) c_{\alpha}^{\text{UV}}(\mathbf{r}) - \frac{(t^*(\mathbf{r}))^{n+1}}{(n+1)!} \Theta(h_{\alpha}^{\text{UV}}(\mathbf{r})) \right], \quad (3.13)$$

where  $\Theta$  is the Heaviside function.

Analogous versions of Eqs. 3.6, 3.12 and 3.13 are used in 1D-RISM. While these are used for DRISM they 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 [82, 91–94]. These different functional forms are equivalent in XRISM but not in DRISM. The form introduced by Pettitt and Rossky [92] is the most popular in the literature and the default selection in `rismld`. It is possible to have `rismld` evaluate and output all three functional forms (see ??) but, for DRISM, none of these expressions are strictly correct.

The force equation

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

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 [72, 95–97].

In addition to closure specific expressions for the solvation free energy, other approximations also exist. The Gaussian fluctuation (GF) approximation[98, 99] is given as

$$\mu^{\text{ex,GF}} = k_B T \sum_{\alpha} \rho_{\alpha}^V \int d\mathbf{r} \left[ -c_{\alpha}^{\text{UV}}(\mathbf{r}) - \frac{1}{2} h_{\alpha}^{\text{UV}}(\mathbf{r}) c_{\alpha}^{\text{UV}}(\mathbf{r}) \right] \quad (3.14)$$

and has been shown to yield improved absolute solvation free energies for both polar and non-polar solutes[99, 100] but not necessarily for relative free energies[101]. It is not associated with a particular closure but is typically used in place of the expression for a given closure.

Eqs. (3.11)-(3.13) give the total solvation free energy,  $\Delta G_{\text{sol}}$ , but it is often useful to decompose this into electrostatic (solvent polarization),  $\Delta G_{\text{pol}}$ , and non-electrostatic (dispersion and cavity formation),  $(\Delta G_{\text{dis}} + \Delta G_{\text{cav}})$ , 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,

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

$\Delta G_{\text{sol}}$  is produced by a 3D-RISM calculation on the charged solute.  $\Delta G_{\text{pol}}$  is then the difference of the two calculations. As a point of reference, generalized-Born and Poisson-Boltzmann methods calculate only  $\Delta G_{\text{pol}}$  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

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

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  $N_{\text{MDIIS}}$  is the number of MDIIS vectors used to accelerate convergence.  $u^{\text{UV}}$ ,  $c^{\text{UV}}$  and the residual of  $c^{\text{UV}}$  are stored in real-space only and require a full grid for each solvent.  $c^{\text{UV}}$  and its residual also require  $N_{\text{MDIIS}}$  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  $2N_y N_x$  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  $N_{\text{box}}$  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 1) 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 ( $\geq 7$ ). 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

$g^{\text{UV}}$ ,  $h^{\text{UV}}$  and  $c^{\text{UV}}$  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

$$\epsilon_{\Delta G_{\text{solv}}} \approx 10 \times \text{tolerance}. \quad (3.15)$$

Molecular dynamics [72], minimization and trajectory post-processing [101] 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  $\Delta G_{\text{solv}}$  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  $\Delta G_{\text{solv}}$ .

Almost all applications should use a grid spacing of  $0.3$  to  $0.5 \text{ \AA}$  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).

### 3.2.4 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.

#### 3.2.4.1 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.

**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  $1\text{e-}2$ . See details in Subsection ??, Subsection 3.3.1.1 and Section ??.

#### 3.2.4.2 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 `rismld`, 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.

#### 3.2.4.3 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.

#### 3.2.4.4 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 than 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.

## 3.3 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.

### 3.3.1 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] -xv xvvfile -guv guvroot -huv huvroot
```

```

-cuv cuvroot -uuv uuvroot -asyp asympfile
-quv quvroot -chgdist chgdistroot
-exchem exchemroot -solvene solveneroot -entropy entropyroot -potUV potUVroot

```

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

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

**asypfile** *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 [ $e/\text{\AA}$ ]. 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`.

### 3.3.1.1 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 §3.1.1).

= **0** Off.

= **1** On.

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

= **0** Off.

= **1** On.

**uccoeff** [0,0,0,0] Compute the UC excess chemical potential functional with the provided coefficients (see §??).  $a$  and  $b$  are the coefficients for the original UC functional, though using the closure excess chemical potential functional.  $aI$  and  $bI$  are optional and provide temperature dependence to the correction (UCT in [102]).

#### 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<sup>3</sup> 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

**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.

**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`.

#### 3.3.1.2 Example

##### Molecular Dynamics (`imin=0`)

```
molecular dynamics with 3D-RISM and impulse MTS
&cntrl
  ntx=1, ntp=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
  ntp=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, ntp=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,[103] 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,[104] and is based on earlier work of Marcus Elstner.[105, 106]
- 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[107] 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,[108] AM1,[109] RM1,[110] MNDO,[111] PDDG/PM3,[112] PDDG/MNDO,[112] PM3CARB1,[113], PM3-MAIS[114, 115], MNDO/d[116–118], AM1/d (Mg from AM1/d[119] and H, O, and P from AM1/d-PhoT[120]) and PM6[121].

Also available is the density functional theory-based tight-binding (DFTB) Hamiltonian[104, 122, 123] and its self-consistent-charge version with Taylor expansion up to second order (SCC-DFTB or DFTB2)[105] and third-order (DFTB3)[124]. If you use the mio-1-1 parameters for DFTB2, you can add an empirical correction for dispersion effects[125] and calculate CM3 charges[126] (both only for elements H, C, N, O, S, P). Diagonal third-order corrections are available for DFTB2[127] 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](http://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 5) 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  $\alpha$  of the QM atoms are taken from PM3 (a default value of five is used for the exponents  $\alpha$  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/AmberTools/src/sqm/qm2\_parameters.F90, variable `alp_pm6`). The magnitude of the coefficients  $\alpha$  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](http://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. Frauenheim, 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. Elstner, *J. Chem. Theory Comput.* **11** (2015) 332-342.

Additional parameter files can be obtained from the website [www.dftb.org](http://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[128]. The empirical dispersion correction follows the formalism for DFT-D[129] 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

$$E_{dis} = -s_6 \sum_{ij} f_{damp}(r_{ij}, R_{ij}^0) C_{6,ij} r_{ij}^{-6}, \quad (4.1)$$

where  $r_{ij}$  is the distance between two atoms  $i$  and  $j$ ,  $R_{ij}^0$  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

$$f_{damp}(r_{ij}, R_{ij}^0) = \left[ 1 + \exp \left( -\alpha \frac{r_{ij}}{s_R R_{ij}^0} - 1 \right) \right]^{-1}. \quad (4.2)$$

Bondi vdW radii[130] are used and for a pair of unlike atoms we have

$$R_{ij}^0 = \frac{R_{ii}^{0^3} + R_{jj}^{0^3}}{R_{ii}^{0^2} + R_{jj}^{0^2}}. \quad (4.3)$$

For the  $C_6$  coefficients the following equation is used,

$$C_{6,ij} = 2 \frac{(C_{6,ii}^2 C_{6,jj}^2 N_{eff,i} N_{eff,j})^{1/3}}{(C_{6,ii} N_{eff,j}^2)^{1/3} + (C_{6,jj} N_{eff,i}^2)^{1/3}}, \quad (4.4)$$

where the Slater-Kirkwood effective number of electrons  $N_{eff,i}$  and the  $C_6$  coefficients can easily be found in the literature[129].

An empirical hydrogen bonding correction[128] 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,

$$E_{\text{H-bond}} = \frac{C_{AB}}{r_{AB}^2} f_{\text{geom}} f_{\text{damp}}, \quad (4.5)$$

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

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

$$f_{\text{damp}} = \left( \frac{1}{1 + \exp[-100(r_{AB}/2.4 - 1)]} \right) \left( 1 - \frac{1}{1 + \exp[-10(r_{AB}/7.0 - 1)]} \right), \quad (4.8)$$

$$C_{AB} = \frac{C_A + C_B}{2}. \quad (4.9)$$

Here,  $C_A$  and  $C_B$  are the atomic hydrogen bonding correction parameters and the (torsion) angles in the function  $f_{\text{geom}}$  are defined similarly to an earlier hydrogen bond correction[131].

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  $f_{\text{geom}}$  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](http://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 ntp 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  $0.05 \times \sqrt{\text{scfconv}}$ .
  - = 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[107].
- `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 FN*n*3 (*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  $\phi$  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 5), 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 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 5.1 below 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 5.2.
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 ?? and QM/MM simulations via *sander* and QUICK are described in section 5.3 below.

The built-in semi-empirical QM/MM support was written by Ross Walker and Mike Crowley, [103] 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 [132] while regular QM/MM Ewald support is based on the work of Nam *et al.* [133] with QM/MM PME support based on the work of Walker *et al.* [103]. SCC-DFTB support was written by Gustavo Seabra, Ross Walker and Adrian Roitberg, [104] and is based on earlier work of Marcus Elstner. [105, 106] The extension for DFTB3 was written by Andreas Goetz. The interface to external QM packages [134] was originally developed by Andreas Goetz but many others have contributed in the meantime. Vinicius Cruzeiro has coupled Amber to QUICK [135] via the API that was developed by Madu Manathunga.

### 5.1 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  $\Psi$ , which is dependent on the position of the MM and QM nuclei, to yield the system energy  $E_{eff}$ :

$$H_{eff}\Psi(x_e, x_{QM}, x_{MM}) = E_{eff}(x_{QM}, x_{MM})\Psi(x_e, x_{QM}, x_{MM}) \quad (5.1)$$

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

## 5 QM/MM calculations

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:

$$E_{eff} = \langle \Psi | H_{QM} + H_{QM/MM} | \Psi \rangle + E_{MM} \quad (5.2)$$

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

$$H_{QM/MM} = \sum_q \sum_m \left[ Q_m h_{electron}(x_e, x_{MM}) - Q_m Z_q h_{core}(x_{QM}, x_{MM}) + \left( \frac{A}{r_{qm}^{12}} - \frac{B}{r_{qm}^6} \right) \right] \quad (5.3)$$

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

$$H_{QM/MM} = \sum_q \sum_m \left[ \frac{Q_m Q_q}{r_{qm}} + \left( \frac{A}{r_{qm}^{12}} - \frac{B}{r_{qm}^6} \right) \right] \quad (5.4)$$

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. 5.2 over a single determinant built from molecular orbitals

$$\phi_i = \sum_j c_{ij} \chi_j \quad (5.5)$$

where the  $c_{ij}$  are molecular orbital coefficients and the  $\chi_j$  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.

### 5.1.1 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).

### 5.1.2 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:[136]

$$E_{QM/MM}^{core} = Z_a q_m (s_a s_a, s_m s_m) \left[ 1 + \frac{|q_m|}{q_m} \cdot \left( -e^{-f_1^a \cdot R_{am}} + e^{-f_2^a \cdot R_{am}} \right) \right] \quad (5.6)$$

## 5 QM/MM calculations

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.[136] 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,  $\rho_{mm}$ , is introduced to each type of QM atoms in order to "fine-tune" the QM electron-MM charge interaction (Eq. 5.7). Although  $\rho_{mm}$  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).

$$E_{QM/MM}^{electron} = -q_m(\mu_a V_a, s_m s_m) = \sum_{\ell_a} \sum_{\ell_m} [M_{\ell_a k}^a M_{\ell_m k}^m] \quad (5.7)$$

where

$$[M_{\ell_a k}^a M_{\ell_m k}^m] = \frac{e^2}{2^{l_a+l_m}} \sum_{i=1}^{2^{l_a}} \sum_{j=1}^l \left[ r_{ij}^2 + (\rho_{l_a}^a + \rho_{mm}^a)^2 \right]^{-1/2} \quad (5.8)$$

### 5.1.3 Generalized Born implicit solvent

The implementation of Generalized Born (GB) for QM/MM calculations is based on the method described by Pellegrini and Field.[132] Here, the total energy is taken to be  $E_{eff}$  from Eq. 5.2 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.

### 5.1.4 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.[133] 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 [133] 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[103] 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. 5.9 and 5.10), thus resulting in energy conservation problems in the simulation.

$$E_{QM/MM}^{r < cutoff} = -q_m(\mu_a v_a, s_m s_m) + Z_a q_m(s_a s_a, s_m s_m)(1 + scale) \quad (5.9)$$

$$E_{QM/MM}^{r > cutoff} = \frac{q_m(Z_a - \sum c_{\mu\mu})}{r} \quad (5.10)$$

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 5.1.6 below.

### 5.1.5 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).

**5.1.6 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.

<i>iqmatoms</i>	comma-separated integer list containing the atom numbers (from the prmtop file) of the atoms to be treated quantum mechanically.
<i>qmmask</i>	Mask specifying the quantum atoms. E.g. :1-2, = residues 1 and 2. See mask documentation for more info.
<i>qmcut</i>	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 <i>qmcut</i> of <i>any</i> QM atom is included in the pair list. For PME calculations, this parameter just affects the division of forces between direct and reciprocal space. <i>Note</i> : 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 <i>cut</i> .
<i>qm_ewald</i>	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.

## 5 QM/MM calculations

<code>qm_theory</code>	Level of theory to use for the QM region of the simulation. (Hamiltonian). Default is to use the semi-empirical hamiltonian PM3. See the <a href="#">Section 4.3</a> for details.
<code>qmmm_int</code>	<p>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 <code>qmmm_int=0</code> DFTB calculations (<code>qm_theory=DFTB</code>) always use a simple mulliken charge - resp charge interaction and the value of <code>qmmm_int</code> has no influence.</p> <p><b>= 0</b> 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.</p> <p><b>= 1</b> (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.</p> <p><b>= 2</b> 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.</p> <p><b>= 3</b> Using this along with <code>qm_theory=PM3</code> invokes a reformulated QM core-MM charge potential at the QM-MM interface (Eq. 5.6). Current parametrization limits the QM region to H, C, N and O atoms only; MM region is not restricted.[136]</p> <p><b>= 4</b> Currently not in use.</p> <p><b>= 5</b> 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.</p>
<code>qmshake</code>	<p>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 <code>NTC=2</code>. 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 <code>prmtop</code> 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.</p> <p><b>= 0</b> Do not shake QM H atoms.</p> <p><b>= 1</b> Shake QM H atoms if SHAKE is turned on (<code>NTC&gt;1</code>) (default).</p>
<code>printdipole</code>	<p>Controls whether the dipole moment shall be printed every <code>ntpr</code> steps.</p> <p><b>= 0</b> Do not print the dipole moment (default).</p> <p><b>= 1</b> Print the dipole moment of the QM region.</p> <p><b>= 2</b> Print the total dipole moment of the QM and MM region.</p>
<code>writpdb</code>	<p><b>= 0</b> Do not write a PDB file of the selected QM region. (default).</p> <p><b>= 1</b> 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 <code>qmmm_region.pdb</code>.</p>

**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.

### 5.1.7 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.

## 5 QM/MM calculations

- `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*  $\geq nlink$  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*  $\geq nlink$ .

### 5.1.8 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[107].

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[107]) when the charge dependence parameters are zero. There are eight *qxd* parameters (symbols used in the reference[107] are indicated in the parentheses): *qxd\_s* (*s*), *qxd\_z0* ( $\zeta(0)$ ), *qxd\_zq* ( $\zeta_q$ ), *qxd\_d0* ( $\alpha_1$ ), *qxd\_dq* ( $3 \times B$ ), *qxd\_q0* ( $\alpha_2$ ), *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).

## 5.2 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[134]. 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) [137, 138]

- GAMESS-US [139, 140]
- NWChem [141]

Mechanical and electrostatic embedding is available for

- Gaussian [142]
- Orca [143]
- Q-Chem[144][144]
- TeraChem [145]
- QUICK [146–148]
- MRCC [149, 150]
- Fireball [151]

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 ??). 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 5.3. 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. 152, 153. In what follows we will describe the new interface that is native to *sander*.

### 5.2.1 Theory

As described in section 5.1, 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

$$E = \langle \Psi | \mathcal{H}_{QM} + \mathcal{H}_{QM/MM} | \Psi \rangle + E_{MM}. \quad (5.11)$$

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

$$\begin{aligned} E_{QM/MM}^{electronic} = & \sum_{A \in MM} \int \rho(\mathbf{r}) \frac{Q_A}{|\mathbf{r} - \mathbf{R}_A|} d\mathbf{r} + \sum_{A \in QM, B \in MM} \frac{Z_A Q_B}{R_{AB}} \\ & + \sum_{A \in QM, B \in MM} \epsilon_{AB} \left[ \left( \frac{\sigma_{AB}}{R_{AB}} \right)^{12} - \left( \frac{\sigma_{AB}}{R_{AB}} \right)^6 \right]. \end{aligned} \quad (5.12)$$

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 interaction 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 (5.11) with respect to the Cartesian coordinates of the atom,

$$\mathbf{F}_A = -\nabla_A E_{QM} - \nabla_A E_{QM/MM} - \nabla_A E_{MM}, \quad (5.13)$$

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

$$\begin{aligned} \nabla_A E_{QM/MM}^{electronic} = & Z_A \sum_{B \in MM} \frac{Q_B (\mathbf{R}_A - \mathbf{R}_B)}{R_{AB}^3} + \sum_{B \in MM} \int \frac{\partial \rho(\mathbf{r})}{\partial \mathbf{R}_A} \frac{Q_B}{|\mathbf{r} - \mathbf{R}_B|} d\mathbf{r} + \sum_{B \in MM} \nabla_A V_{AB}^{LJ} \\ = & -Z_A \mathbf{E}_{MM}(\mathbf{R}_A) - \int \rho(\mathbf{r}) \mathbf{E}_{MM}(\mathbf{r}) d\mathbf{r} + \sum_{B \in MM} \nabla_A V_{AB}^{LJ} \end{aligned} \quad (5.14)$$

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  $V_{AB}^{LJ}$  is the Lennard-Jones potential from (5.12) and

$$\begin{aligned} \nabla_B E_{QM/MM}^{electronic} = & Q_B \sum_{A \in QM} \frac{Z_A (\mathbf{R}_B - \mathbf{R}_A)}{R_{AB}^3} + \int \rho(\mathbf{r}) \frac{Q_B (\mathbf{R}_B - \mathbf{r})}{|\mathbf{r} - \mathbf{R}_B|^3} d\mathbf{r} + \nabla_B E_{QM/MM}^{mechanic} \\ = & -Q_B \mathbf{E}_{QM}(\mathbf{R}_B) + \sum_{A \in QM} \nabla_B V_{AB}^{LJ} \end{aligned} \quad (5.15)$$

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.

### 5.2.2 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 conservation 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).

### 5.2.3 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 (5.12). 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.



### 5.2.4 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.

### 5.2.5 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.

### 5.2.6 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 opposed 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.

### 5.2.6.1 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**

<i>basis</i>	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: <i>basis</i> = 'DZP')
<i>core</i>	Type of frozen core to use. Allowed values are: None, Small, Medium, Large. (Default: <i>core</i> = 'None')
<i>zlmfit</i>	Quality of density fit with the ZLM fit method. (Default: <i>zlmfit</i> = 'good')
<i>fit_type</i>	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: <i>fit_type</i> = '')
<i>xc</i>	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: <i>xc</i> = 'GGA BLYP')
<i>scf_iter</i>	Maximum number of SCF cycles allowed. (Default: <i>scf_iter</i> = 50)

## 5 QM/MM calculations

<code>scf_conv</code>	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 <code>scf_conv</code> . (Default: <code>scf_conv = 1.0d-06</code> )
<code>beckegrid</code>	Quality of Becke integration grid. Allowed values are: Normal, Good, VeryGood. (Default: <code>core = 'Good'</code> )
<code>integration</code>	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: <code>integration = -1.0</code> )
<code>num_threads</code>	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: <code>num_threads = 0</code> [this causes ADF to use all available cores on a machine])
<code>use_dftb</code>	Specifies whether DFTB shall be used with ADF's DFTB program <code>dftb</code> . If <code>use_dftb = 1</code> then DFTB will be used and only variables <code>charge</code> and <code>scf_conv</code> will be considered. (Default: <code>use_dftb = 0</code> [do not use DFTB, regular DFT calculation]) - works only with older DFTB versions (prior to 2011).
<code>exactdensity</code>	The exact (as opposed to fitted) electron density is used for the evaluation of the exchange-correlation potential if <code>exactdensity = 1</code> . (Default: <code>exactdensity = 0</code> )
<code>use_template</code>	Determine whether or not to use a user-provided template file for running external programs. (Default: <code>use_template = 0</code> )
<code>ntpr</code>	Controls frequency of printing for dipole moment to file <code>adf_job.dip</code> (Defaults to <code>&amp;cntrl</code> namelist variable <code>ntpr</code> )
<code>dipole</code>	Toggles writing of dipole moment to file <code>adf_job.dip</code> (Default: <code>dipole = 0</code> )

**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
```

### 5.2.6.2 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 `run.gms` look for the executable `gamesss.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 `run.gms` is located (for example `/opt/gamess/`). We assume that the `run.gms` script copies the output `.dat` files to the directory from which GAMESS is invoked. If this is not the case, please modify the script `run.gms` 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

<code>basis</code>	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: <code>basis = "6-31G**"</code> )
<code>method</code>	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: <code>method = "BP86"</code> )
<code>nrad</code>	Number of radial points in the Euler-MacLaurin quadrature of the XC potential and energy density. (Default: <code>nrad = 96</code> )
<code>nleb</code>	Number of angular points in the Lebedev grids for the numerical quadrature of the XC potential and energy density. (Default: <code>nleb = 590</code> [The GAMESS default of 302 is not accurate enough to conserve energy])
<code>scf_conv</code>	SCF convergence threshold. Convergence is reached when the absolute density change between two consecutive SCF cycles is less than <code>scf_conv</code> . (Default: <code>scf_conv = 1.0D-06</code> )
<code>maxit</code>	Maximum number of SCF iterations. (Default: <code>maxit = 50</code> )
<code>gms_version</code>	This is the version number specified when building GAMESS. (Default: <code>gms_version = 00</code> )
<code>num_threads</code>	Number of threads (and thus CPU cores) for GAMESS to use. Note that GAMESS may require a special setup in the <code>run.gms</code> script to be able to run using multiple threads. Unless <code>num_threads</code> is explicitly specified, GAMESS will only use one thread (run on one core). (Default: <code>num_threads = 1</code> )
<code>mwords</code>	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: <code>mwords = 50</code> )
<code>use_template</code>	Determine whether or not to use a user-provided template file for running external programs. (Default: <code>use_template = 0</code> )
<code>ntpr</code>	Controls frequency of printing for dipole moment and atomic charges to files <code>gms_prop.ext</code> (Defaults to <code>&amp;cntrl</code> namelist variable <code>ntpr</code> )
<code>chelpg</code>	CHELPG charges are calculated if <code>chelpg = 1</code> . These charges are written to the file <code>gms_prop.chg</code> (Default: <code>chelpg = 0</code> )
<code>dipole</code>	Toggles writing of dipole moment to file <code>gms_prop.dip</code> (Default: <code>dipole = 0</code> )

## 5 QM/MM calculations

**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 `$CONTROL` 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*.

### 5.2.6.3 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 <code>scf_conv</code> . Set in the form of $10^{-N}$ . (Default: <code>scf_conv</code> = 8)
num_threads	Number of threads (and thus CPU cores) for Gaussian to use. Unless <code>num_threads</code> is explicitly specified, Gaussian will only use one thread (run on one core). (Default: <code>num_threads</code> = 1)
executable	Optional name of the Gaussian executable. (Default: If a string for this namelist variable is not specified then <code>g16</code> , <code>g09</code> , and <code>g03</code> 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.)

<code>use_template</code>	Determine whether or not to use a user-provided template file for running external programs. (Default: <code>use_template = 0</code> )
<code>ntpr</code>	Controls frequency of printing for dipole moment to file <code>gau_job.dip</code> (Defaults to <code>&amp;cntrl</code> namelist variable <code>ntpr</code> )
<code>dipole</code>	Toggles writing of dipole moment to file <code>gau_job.dip</code> (Default: <code>dipole = 0</code> )
<code>mem</code>	String that specifies how much memory Gaussian should be allowed to use. (Default: <code>'256MB'</code> )

**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.

#### 5.2.6.4 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

<code>basis</code>	Basis set type to be used in the calculation. Possible choices include <code>svp</code> , <code>6-31g</code> , etc. See Orca manual for a complete list. (Default: <code>basis = "SV(P)"</code> )
<code>cbasis</code>	Auxiliary basis set for correlation fitting. See Orca manual for a complete list. (Default: <code>basis = "NONE"</code> )
<code>jbasis</code>	Auxiliary basis set for Coulomb fitting. See Orca manual for a complete list. (Default: <code>basis = "NONE"</code> )

## 5 QM/MM calculations

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 <i>convkey</i> . 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 <i>grid</i> to conserve energy.)
maxiter	Maximum number of SCF iterations. (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 <code>orc_job.dip</code> (Defaults to &cntrl namelist variable ntptr)
dipole	Toggles writing of the dipole moment to file <code>orc_job.dip</code> (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 runtime and adding coordinates):

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

### 5.2.6.5 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_procs	Number of MPI processes for Q-Chem to use. The total number of CPUs to be used is num_mpi_procs times num_threads. (Default: num_mpi_procs = 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_procs 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)
ntrpr	Controls frequency of printing for the dipole moment to file qc_job.dip (Defaults to &cntrl namelist variable ntrpr)
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')

## 5 QM/MM calculations

**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 JOBTYP and writing the \$molecule input file sections.

### 5.2.6.6 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**

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



`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[154]; the region highlighted in bold is controlled by the keyword). Only single point calculations are currently possible with such multilayer 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,
  coreembed = 'LMP2',
  coreembedatoms = 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`, `coreembed`, `scfguess`. The interface will take care of adding other keywords and writing the coordinate input file section.

### 5.2.6.7 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

<code>basis</code>	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: <code>basis = '6-31G'</code> )
<code>method</code>	Method to be used in the calculation, can be either 'RHF' or some supported DFT functional. Popular choices are 'BLYP', 'PBE' and 'B3LYP'. (Default: <code>method = 'BLYP'</code> )
<code>dftd</code>	Determines whether dispersion corrections are applied in the case of DFT calculations. (Default: <code>dftd = 'no'</code> )
<code>precision</code>	Precision model setting (single vs double precision). (Default: <code>precision = 'mixed'</code> )
<code>dynamicgrid</code>	Use coarse grid during early SCF iterations. (Default: <code>dynamicgrid = 'yes'</code> )
<code>threall</code>	Determines a variety of thresholds. (Default: <code>threall = 1.0E-11</code> )

<code>convthre</code>	SCF convergence threshold for the wavefunction. (Default: <code>convthre = 3.0E-05</code> , which leads to SCF energy convergence of approximately $10^{-7}$ au or $10^{-4}$ kcal/mol)
<code>maxit</code>	Maximum number of SCF iterations. (Default: <code>maxit = 100</code> )
<code>dftgrid</code>	DFT grid to be employed for the numerical XC quadrature in DFT calculations. (Default: <code>dftgrid = 1</code> )
<code>ngpus</code>	Determines how many GPUs are to be used. (Default: <code>ngpus = 0</code> , which uses all available GPUs)
<code>gpuids</code>	If <code>ngpus</code> has a value other than zero, this determines the IDs of the GPUs to be used for the calculation. (Default: <code>gpuids = 0, 1, 2</code> , etc.)
<code>executable</code>	Name of the TeraChem executable. (Default: <code>executable = terachem</code> )
<code>use_template</code>	Determine whether or not to use a user-provided template file for running external programs. (Default: <code>use_template = 0</code> )
<code>ntpr</code>	Controls frequency of printing for dipole moment and atomic charges to files <code>tc_job.ext</code> . (Defaults to <code>&amp;cntrl</code> namelist variable <code>ntpr</code> )
<code>charge_analysis</code>	Toggles writing of atomic charges to file <code>tc_job.chg</code> (Options: 'none' or 'Mulliken'. Default: <code>dipole = 'none'</code> )
<code>dipole</code>	Toggles writing of dipole moment to file <code>tc_job.dip</code> (Default: <code>dipole = 0</code> )

**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
```

## 5.2.6.8 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.io>. 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.io>). 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**

<i>basis</i>	Path to the Fdata directory. (Default: <i>basis</i> = './Fdata')
<i>max_scf_iterations</i>	Maximum number of iterations in the loop for the calculation of the self-consistent charges. (Default: <i>max_scf_iterations</i> = 70)
<i>sigmatol</i>	Threshold for self-consistency in the electronic structure calculations. (Default: <i>sigmatol</i> = 1.0E-08)
<i>idftd3</i>	DFTD3 dispersion correction. (No correction: <i>idftd3</i> = 0; Dispersion correction for BLYP: <i>idftd3</i> = 1; Default: <i>idftd3</i> = 0)
<i>iwrtcharges</i>	Writes atomic charges in fireball output. (Default: <i>iwrtcharges</i> = 0)
<i>iwrteigen</i>	Writes energy levels in fireball output. (Default: <i>iwrteigen</i> = 0)

For a 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* namelist

```
&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
```

## 5.3 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), [146–148, 155, 156] 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 5.2 or 2) an application programming interface (API). As shown in reference [135], 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 ?? for installation instructions. Additionally, a list of *QUICK* features and limitations has been presented in section ??.

**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 5.2).

### 5.3.1 Usage

As discussed in chapter ??, 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 environment). This step ensures that the location of the necessary executables and libraries are set in the environmental variables of the operating system.

#### 5.3.1.1 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.

#### 5.3.1.2 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 required 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.

### 5.3.1.3 &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 separately 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 chosen 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.





## 6 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.

### 6.1 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.

#### 6.1.1 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 exist 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.

## 6.2 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 be 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 1.7.

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 is 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 (-CH<sub>2</sub>-O-CH<sub>2</sub>-) units. Let us suppose that one created three residues in the PREP unit: CRA, CRB, CRC. Each of these is a (-CH<sub>2</sub>-O-CH<sub>2</sub>-) 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  $10.0 \text{ kcal} \cdot \text{mol}^{-1} \cdot \text{\AA}^{-2}$ ; CA atoms in residues 61 to 127 will have a weaker restraint force constant.





## 7 Sampling configuration space

The "middle" scheme [Section ??] 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.

### 7.1 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. [157? –160]

**Overview:** The input parameter, *tsgavg*, defines the lower limit period of the low frequency motion. Typically, *tsgavg*=0.2 ps is suitable to define bond stretching and bending motions as high frequency motion and is recommended to enhance motions like phase separation, secondary structure folding, and ligand docking, while *tsgavg*=1.0 ps is suitable to include more local motions such as bond rotation and solvent relaxation and is recommended for protein domain motion, and protein-protein docking. The input parameter, *sgft* or *sgff*, defines the strength of the guiding effect. *sgft* between  $-1$  and  $+1$  sets the momentum guiding effect, with 0 for regular LD or MD simulations. A larger *sgft* will enhance low frequency motion to accelerate conformational search. *sgff* between  $-0.32$  and  $+0.32$  defines a force guiding factor to target energy barriers. A negative value will flatten energy barriers. Normally, *sgft* and *sgff* have opposite effects on conformational distribution and there exist balanced values to conserve the canonical ensemble. SGLD accelerates slow events to an affordable time scale while minimizing the perturbation 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[158, 161? ]. The current implementation is mainly based on the most recent reformulation named the generalized self-guided molecular simulation method (SGMDg or SGLDg)[? ]. The previous method[161, 162], SGLDfp is no longer available. As an alternative to Langevin dynamics (LD), self-guided Langevin dynamics via generalized Langevin equation (SGLD-GLE) [160] can exactly preserve canonical ensemble while avoiding the slow down by friction forces. SGLD-GLE method can be turned on by setting *sgfg* between  $-1$  and  $+1$ . SGLD-GLE can be used with SGLDg by setting *sgft*, *sgff*, and *sgfg*.

<i>isgld</i>	SGLD algorithm index. Default <i>isgld</i> = 0, SGLD is disabled; <i>isgld</i> >0 will turn on SGLDg/SGMDg and/or SGLD-GLE method.
<i>tsgavg</i>	Local averaging time ( <i>psec</i> ) for the guiding force calculation. Default 0.2 <i>psec</i> . A larger value defines slower motion to be enhanced.
<i>sgft</i>	Momentum guiding factor. Defines the strength of the guiding effect. Default 0.0. Suggested value is 1.0 when <i>sgff</i> =0. Its value range is $-1 \sim 1$ with larger value leads to stronger low frequency motion. When <i>sgft</i> is set $<-1$ or $>1$ , it is reset to the balanced value of <i>sgff</i> to preserve canonical ensemble.

## 7 Sampling configuration space

<code>sgff</code>	Force guiding factor. <i>sgff</i> is used to scale down low frequency energy surface by a factor, $(1+sgff)$ . <i>sgff</i> is suggested to take values between -0.32 and 0.32, with default value of 0. Suggested value is -0.3 when <i>sgft</i> =0. When <i>sgff</i> is set <-1 or >1, it is reset to the balanced value of <i>sgft</i> to preserv canonical ensemble.
<code>sgfg</code>	momentum guiding factor for SGLD-GLE. Degault is 0. Its value range is -1~1 with larger value resulting in faster low frequency motion. Suggested value is 1 when <i>sgft</i> =0 and <i>sgff</i> =0.
<code>isgsta</code>	The first atom index of SGLD region. Default is 1.
<code>isgend</code>	The last atom index of SGLD region. Default is <i>natom</i> .
<code>nsgsize</code>	size of the bonded substructure for each atom used to calculate average guiding forces. 1 for no bonded structure is considered; 2 for all atoms connected through bonds or angles ; 3 for all atoms connected through bonds, angles, and dihedral angles. Default is 1.

The output of SGMD/SGLD simulations contains the following properties related to the enhancement in conformational search and reweighting of conformational distribution:

METHOD: GAMM TEMPLF TEMPHF EPOTLF EPOTHF EPOTLLF SGWT

These quantities are instantaneous values defined as below:

METHOD: SGLD when *gamma\_ln*>0 or SGMD when *gamma\_ln*=0

GAMM: Average atomic friction constant based on atom interactions.

TEMPLF: low frequency temperature

TEMPHF: high frequency temperature. Apparent temperature=TEMPLF+TEMPHF

EPOTLF: low frequency potential energy

EPOTHF: high frequency potential energy, EPOT=EPOTLF+EPOTHF

EPOTLLF: Average of low frequency potential energy. It is needed for reweighting.

SGWT: Weighting number. exp(SGWT) is the weighting factor of current frame.

The weight of a conformation is calculated by

$$\text{Weight}=\exp(\text{SGWT})=\exp((p(\text{sgft})-\text{sgff})*(\text{EPOTLF}-\text{EPOTLLF})/(\text{KBOLTZ}*\text{Temp}))$$

$$\text{where: sgft}=(1+p(\text{sgft}))^2-1/(1+p(\text{sgft}))$$

or:

$$w_i = \exp\left(\frac{\lambda_{FP} - \lambda_F}{kT}(E_{LF} - E_{LLF})\right)$$

where  $p(\text{sgft})$  or  $\lambda_{FP}$  is called the balanced force guiding factor of  $\lambda_p$ . The relation between  $\text{sgft}$ ,  $\lambda_p$ , and  $p(\text{sgft})$ ,  $\lambda_{FP}$ , is:  $\lambda_p = (1 + \lambda_{FP})^2 - \frac{1}{1 + \lambda_{FP}}$ . Similarly,  $\lambda_{PF}$  is called the balanced momentum guiding factor of  $\lambda_F$ , where  $\lambda_{PF} = (1 + \lambda_F)^2 - \frac{1}{1 + \lambda_F}$ . When both  $\lambda_p$  and  $\lambda_F$  are used in a SGMDg/SGLDg simulation, it is recommended that  $-1 \leq \lambda_p - \lambda_{PF} \leq 1$ .

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, averages and reweighted averages of the SGMD/SGLD properties. Ensemble average properties can be calculated through reweighting:

$$\langle P \rangle = \frac{\sum_{i=N_0}^N w_i P_i}{\sum_{i=N_0}^N w_i}$$

Below are example input files to run SGMD/SGLD simulations.

1. a SGLD simulation with sgft only:

```
Sample SGLD simulation for enhanced conformational search
&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,
ntc=2, ntf=2, tol=0.000001,
dt=0.002, ntb=0, tempi=300., temp0=300.,
isgld=1, tsgavg=0.2, sgft=1.0,
/
```

2. a SGLD simulation using bonded substructure(nsgsize=2) to estimate low frequency motion, which reduces noises due to bond stretching and bending.

```
Sample SGLD simulation using bonded substructures
&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,
ntc=2, ntf=2, tol=0.000001,
dt=0.002, ntb=0, tempi=300., temp0=300.,
isgld=1, tsgavg=0.2, sgft=1.0, nsgsize=2,
/
```

3. a SGLDg simulation using both *sgft* and *sgff* factors. When both guiding factors are used, it is recommend that  $-1 \leq \lambda_p - (1 + \lambda_F)^2 + \frac{1}{1 + \lambda_F} \leq 1$ .

```
Sample SGLDg simulation using both sgft and sgff
&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,
ntc=2, ntf=2, tol=0.000001,
dt=0.002, ntb=0, tempi=300., temp0=300.,
isgld=1, tsgavg=0.2, sgft=0.5, sgff=-0.1,
/
```

4. a SGMDg simulation when gamma\_ln=0:

```
Sample SGMDg simulation using both sgft and sgff
&cntrl
nstlim=1000, cut=99.0, igb=1, saltcon=0.1,
ntpr=100, ntwr=100000, ntt=1, tautp=1.0,
ntx=5, irest=1,
ntc=2, ntf=2, tol=0.000001,
dt=0.002, ntb=0, tempi=300., temp0=300.,
isgld=1, tsgavg=0.2, sgft=0.5, sgff=-0.1,
/
```

5. a SGLD-GLE simulation using sgfg to achieve enhanced conformational search while conserve canonical ensemble exactly:

```
Sample SGLD-GLE simulation to conserve canonical ensemble
```

```
&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,
ntc=2, ntf=2, tol=0.000001,
dt=0.002, ntb=0, tempi=300., temp0=300.,
isgld=1, tsavg=0.2, sgfg=1.0,
/
```

## 7.2 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 *tgfitmask* 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:

$$E = 0.5 * \text{TGTMDFRC} * \text{NATTGTRMS} * (\text{RMSD} - \text{TGTRMSD}) ** 2$$

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 *tgtdmfrc* 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	<p>= 0 no targeted MD (default)</p> <p>= 1 use targeted MD</p> <p>= 2 use targeted MD to multiple targets (Multiply-targeted MD, or MTMD, see next section below)</p>
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.
tgtdmfrc	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.
tgfitmask	Define the atoms that will be used for the rms superposition between the current structure and the reference structure. Syntax is in Chapter 6.1.1.
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 6.1.1.

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 *tgmdfrc* 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.

## 7.3 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.

### 7.3.1 Variables in the &tgt namelist:

<i>refin</i>	The file name of the reference structure used. The input and reference coordinates are expected to match the <i>prmtop</i> file and have atoms in the same sequence. <i>Default for refin is “”, no reference structure given.</i>
<i>mtmdform</i>	If MTMDFORM > 0, then the reference coordinate file is formatted. Otherwise, the reference coordinate file is an unformatted (binary) file. <i>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.</i>
<i>mtmdstep1</i> , <i>mtmdstep2</i>	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. <i>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.</i>
<i>mtmdvari</i>	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). <i>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.</i>

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 `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	<p>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>i.e.</i></p>
----------	---

$$\text{MTMDFORCE2} = \text{R} \times \text{INCREMENTS} \times \text{MTMDFORCE}$$

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	<p>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 6.1.1. <i>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.</i></p>
----------	--

Namelist &gt; 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.

## 7.4 Low-MODe (LMOD) methods

István Kolossváry’s LMOD methods for minimization, conformational searching, and flexible docking[163–166] 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.

### 7.4.1 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,[167] LBFGS is a Limited-memory Broyden-Fletcher-Goldfarb-Shanno quasi-Newton algorithm,[168] and TNCG is a Truncated Newton linear Conjugate Gradient method with optional LBFGS preconditioning.[169]

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`.

<code>maxcyc</code>	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.
<code>ntmin</code>	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.
<code>drms</code>	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 $10^{-4} \text{ kcal} \cdot \text{mol}^{-1} \cdot \text{\AA}^{-1}$ . 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 <code>lmod_minimize_grms</code> below.

Other options that control XMIN are in the scope of the &lmod namelist. These parameters enable expert control of XMIN.

<code>lbfgs_memory_depth</code>	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.
<code>matrix_vector_product_method</code>	The finite difference Hv matrix-vector product method: "forward" = forward difference, "central" = central difference. Default is forward difference.
<code>xmin_method</code>	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.
<code>xmin_verbosity</code>	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.

### 7.4.2 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  $\text{total\_low\_modes} + 1 \leq \text{arnoldi\_dimension} \leq 3 \times \text{three times the number of atoms}$ . 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  $\leq \text{conflib\_size}$ . 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  $[\text{global\_min}, \text{global\_min} + \text{energy\_window}]$ . 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_rottrans_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_rottrans` The number of rigid-body rotational and translational motions applied to the ligand(s). Such applications occur at each frequency\_ligand\_rottrans-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_rottrans_modes`).

## 7 Sampling configuration space

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`  $\geq$  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.

## 8 Free energies

### 8.1 Thermodynamic integration

In a free energy calculation, the system evolves according to a mixed potential (such as in Eqs. 8.3 or 8.4, 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.[170–172] There are also plenty of recent examples to consult.[173, 174] 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 1.13 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  $\lambda$  is used to interpolate between the "unperturbed" and "perturbed" potential functions.

#### 8.1.1 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 1.13, 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 mdl.o -r mdl.x -inf mdinfo
-O -i mdin -p prmtop.1 -c eq1.x -o mdlb.o -r mdlb.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 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.)

### 8.1.2 Basic inputs for thermodynamic integration

<b>icfe</b>	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. 8.3, below.
<b>clambda</b>	The value of $\lambda$ for this run, as in Eqs. 8.3 and 8.4, below. Zero corresponds to the unperturbed Hamiltonian $V_0$ . $\lambda=1$ corresponds to the perturbed Hamiltonian $V_1$ .
<b>klambda</b>	The exponent in Eq. 8.4, below.
<b>tishake</b>	Flag that determines how SHAKE is handled: <b>= 0</b> Coordinates are synchronized after SHAKE, no constraints removed (default). <b>= 1</b> SHAKE is removed between bonds containing one common and one unique atom. This was the default in previous versions of <i>sander</i> . Note that disabling SHAKE requires the use of a 1 fs timestep.

#### 8.1.2.1 Input flags specific to Sander

<b>idecomp</b>	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 <i>mdout</i> 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.
----------------	---

### 8.1.3 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  $\lambda$ ) and to numerically estimate the integral:

$$\Delta A = A(\lambda = 1) - A(\lambda = 0) = \int_0^1 \langle \partial V / \partial \lambda \rangle_\lambda d\lambda \quad (8.1)$$

If you understand how free energies work, this should not be at all difficult. However, since the actual values of  $\lambda$  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:

$$\Delta A = \sum_i w_i \langle \partial V / \partial \lambda \rangle_i \quad (8.2)$$

Some weights and quadrature points are given in the accompanying table; other formulas are possible,[175] but the Gaussian ones listed there are probably the most useful. The formulas are always symmetrical about  $\lambda = 0.5$ , so that  $\lambda$  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  $\lambda$  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:

$$V(\lambda) = (1 - \lambda)V_0 + \lambda V_1 \quad (8.3)$$

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. 8.1 diverges at  $\lambda = 1$ ; 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.[171] *Sander* and *pmemd* implement one simple way of handling this problem: if you set *klambda* > 1, the mixing rules are

$$V(\lambda) = (1 - \lambda)^k V_0 + [1 - (1 - \lambda)^k] V_1 \quad (8.4)$$

where  $k$  is given by *klambda*. Note that this reduces to Eq. 8.3 when  $k = 1$ , which is the default. If  $k \geq 4$ , the integrand remains finite as  $\lambda \rightarrow 1$ . [171] 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.[176] Note that the behavior of  $\langle \partial V / \partial \lambda \rangle$  as a function of  $\lambda$  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. 8.4 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  $\kappa > 1$ : 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.

$n$	$\lambda_i$	$1 - \lambda_i$	$w_i$
1	0.5		1.0
2	0.21132	0.78867	0.5
3	0.1127 0.5	0.88729	0.27777 0.44444
5	0.04691 0.23076 0.5	0.95308 0.76923	0.11846 0.23931 0.28444
7	0.02544 0.12923 0.29707 0.5	0.97455 0.87076 0.70292	0.06474 0.13985 0.19091 0.20897
9	0.01592 0.08198 0.19331 0.33787 0.5	0.98408 0.91802 0.80669 0.66213	0.04064 0.09032 0.13031 0.15617 0.16512
12	0.00922 0.04794 0.11505 0.20634 0.31608 0.43738	0.99078 0.95206 0.88495 0.79366 0.68392 0.56262	0.02359 0.05347 0.08004 0.10158 0.11675 0.12457

Table 8.1: Abscissas and weights for Gaussian integration.

### 8.1.4 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:

$$V_{V_0,disappearing} = 4\epsilon(1-\lambda) \left[ \frac{1}{\left[ \alpha\lambda + \left(\frac{r_{ij}}{\sigma}\right)^6 \right]^2} - \frac{1}{\alpha\lambda + \left(\frac{r_{ij}}{\sigma}\right)^6} \right] \quad (8.5)$$

$$V_{V_1,appearing} = 4\epsilon\lambda \left[ \frac{1}{\left[ \alpha(1-\lambda) + \left(\frac{r_{ij}}{\sigma}\right)^6 \right]^2} - \frac{1}{\alpha(1-\lambda) + \left(\frac{r_{ij}}{\sigma}\right)^6} \right] \quad (8.6)$$

Please refer to Ref [176] 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 [177]. 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  $\lambda$  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*,  $\lambda$  values should be picked so that  $0.01 < \text{clambda} < 0.99$ . 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)
```

## 8 Free energies

scalpha	The $\alpha$ parameter in 8.5 and 8.6, 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.

### 8.1.4.1 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.
--------	---

### 8.1.4.2 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:

$$V_{0,disappearing} = (1 - \lambda) \frac{q_i q_j}{4\pi\epsilon_o \sqrt{\beta\lambda + r_{ij}^2}} \quad (8.7)$$

for disappearing atoms. Replace  $\lambda$  by  $(1 - \lambda)$  and vice versa for the form for appearing atoms. This introduces a new parameter  $\beta$  which controls the 'softness' of the potential. This is set in the input file via:

scbeta	The parameter $\beta$ in 8.7. Default value is $12\text{\AA}^2$ , 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. [178] contains detailed information on the performance of such single step TI calculations.

## 8.1.5 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  $\lambda$ -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  $\lambda$ -value (Default = 0.1)

`bar_l_max` Maximum  $\lambda$ -value (Default = 0.9)

`bar_l_incr` The increment to increase  $\lambda$  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  $\lambda$ -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.

## 8.2 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.

## 8.3 Adaptively Biased MD, Steered MD, Umbrella Sampling with REMD and String Method

### 8.3.1 Overview

The following describes a suite of modules useful for the calculation of the free energy associated with a reaction coordinate  $\sigma(\mathbf{r}_1, \dots, \mathbf{r}_N)$  (which is defined as a smooth function of the atomic positions  $\mathbf{r}_1, \dots, \mathbf{r}_N$ ):

$$f(\xi) = -k_B T \ln \langle \delta[\xi - \sigma(\mathbf{r}_1, \dots, \mathbf{r}_N)] \rangle,$$

(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 [179] 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 [180]) in the molecular dynamics (MD) context, and by Wang and Landau in the context of Monte Carlo simulations [181]. More recent approaches include the metadynamics method [182, 183]. 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:

$$m_a \frac{d^2 \mathbf{r}_a}{dt^2} = \mathbf{F}_a + \frac{\partial}{\partial \mathbf{r}_a} U[t|\sigma(\mathbf{r}_1, \dots, \mathbf{r}_N)],$$

$$\frac{\partial U(t|\xi)}{\partial t} = \frac{k_B T}{\tau_F} G[\xi - \sigma(\mathbf{r}_1, \dots, \mathbf{r}_N)],$$

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  $\tau_F$  (the flooding timescale) and small kernel width, the biasing potential  $U(t|\xi)$  converges towards  $-f(\xi)$  as  $t \rightarrow \infty$ .

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.  $\xi$ , 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* [184, 185]. It amounts to carrying out several different MD simulations biased by the same  $U(t|\xi)$ , which evolves via:

$$\frac{\partial U(t|\xi)}{\partial t} = \frac{k_B T}{\tau_F} \sum_{\alpha} G[\xi - \sigma(\mathbf{r}_1^{\alpha}, \dots, \mathbf{r}_N^{\alpha})],$$

where  $\alpha$  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 [186–188]. 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:

$$p^B(\xi) = \langle \delta[\xi - \sigma(\mathbf{r}_1, \dots, \mathbf{r}_N)] \rangle_B.$$

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 8.3.7. This methodology has been applied to a variety of biomolecular systems, including small peptides [179, 189, 190], sugar puckering [191], polyproline systems [192–194], guest-host systems [195, 196], polyglutamine systems [197, 198], and DNA systems [199–203]. In addition, SMD simulations (discussed below) have been used to examine transition pathways and mechanisms, to estimate free energy differences [193, 204], and to calculate transition rates [205–207].

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)[208] – all of which enhance the stability and convergence of an ABMD simulation. Also implemented is the Swarms-of-Trajectories String Method (STSM) [209], which gives a way of exploring the Minimum Free Energy Path (MFEP) on free energy landscape. Current version of these codes were implemented 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 [210]. In the original ABMD implementation, the history dependent biasing potential is built up at a fixed rate:

$$U(\xi, t) = U^0(\xi) + \int_0^t dt' \omega G(\xi - \xi'), \quad (8.8)$$

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  $\langle U(\xi, t \rightarrow \infty) \rangle_a \approx U^s(\xi) + u(t)$ , in which  $\langle \cdot \rangle_a$  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 [210]. 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  $\omega$ .

One way to resolve this problem is to update the kernel at a non-uniform rate by means of a "well-tempered"  $\omega$ :

$$U(\xi, t) = U^0(\xi) + \int_0^t dt' \omega(\xi', t') G(\xi - \xi'), \quad (8.9)$$

in which  $\omega(\xi, t)$  is a time-dependent, non-uniform rate chosen to be  $\omega_0 e^{-\beta' U(\xi, t)}$  ( $1/\beta' = k_B T'$  where  $T'$  is a pseudo-temperature) that reduces to a constant  $\omega_0$  in the  $\beta' \rightarrow 0$  limit (*i.e.*, resulting in conventional ABMD). With this choice, one can show that  $\langle U(\xi, t \rightarrow \infty) \rangle_a \approx U^s(\xi) + u(t)$ , ( $u(t)$  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 [211].

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[211]. 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  $\tau$ . 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  $(r_1^n, \dots, r_N^n)$ . The weights are then tested and updated every fixed time interval of length  $\tau$ . 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  $\xi_n^t$  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  $\rho$ . 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  $\Delta t$  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 [211]. 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  $-\epsilon \log(1/N_w)$ . Here,  $\log(1/N_w)$  represents the entropy of uniform weights, and the stopping parameter  $\epsilon$  varies between  $0 \leq \epsilon \leq 1$ . When  $\epsilon = 0$ , the algorithm never stops, while  $\epsilon = 1$  forces a stop irrespective of the values of the weights.

In addition to  $\epsilon$ , there are also two other user-defined variables in the selection algorithm, including the constant  $C$  and the interval time  $\tau$ . While the physical interpretation of  $\tau$  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  $\rho$ . 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 8.1: 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.[208] 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 ( $n = 1, 2, \dots$ ), 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  $\xi$  space), and:

$$U_a^n(\xi, t) = U^{n-1}(\xi) + \int_0^t dt' \omega(\xi', t') K(\xi - \xi') e^{-\beta \omega'}$$

in which  $\omega'$  is either defined as the accumulated work or the transferred work. The accumulated and transferred works are defined as  $\omega_{ac}^t = \int_0^t dt' \frac{\partial}{\partial t'} U_d(\xi', t')$  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 unstable for large biasing potentials. To avoid the instability in either case a cutoff for  $\omega'$  is used (i.e., the algorithm will not be applied if  $\omega'$  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*, *&pmmd*, *&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 8.3.7.

### 8.3.2 Reaction Coordinates

A reaction coordinate is defined in the *colvar* namelist in a separate file. (see Fig. 8.1). 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 (specified 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  $\{\alpha_1, \dots, \alpha_N\}$ , **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.,  $\omega$  backbone dihedral angle). For a list of dihedral angles such as  $\{\alpha_1, \dots, \alpha_N\}$ , **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 RMSDs 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. [212]. An example of variable of this type is presented in Fig. 8.2. Two groups are defined here: one comprises the atoms with indexes 1, 2, 3, 4 (line 3 in Fig. 8.2, 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. 8.2). 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 8.2: 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. 8.2) 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:

$$\text{value} = \sum_p \frac{1 - (r_p/r_0)^6}{1 - (r_p/r_0)^{12}},$$

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 intends 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  $w$  ( $0 \leq w \leq 1$ , the default is zero).

#### N\_OF\_STRUCTURES:

$$\text{value} = \sum_g \frac{1 - (R_g/R_{0,g})^6}{1 - (R_g/R_{0,g})^{12}},$$

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 8.3: 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. 8.3. The atomic groups and reference coordinates are the same as the ones shown in Fig. 8.2. Lines 7 and 11 in Fig. 8.3 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.

$$\text{value} = \frac{1 - \left(R_1/R_{0,1}\right)^6}{1 - \left(R_1/R_{0,1}\right)^{12}} + \frac{1 - \left(R_2/R_{0,2}\right)^6}{1 - \left(R_2/R_{0,2}\right)^{12}}.$$

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[213–216] 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<sup>1</sup> by a unit quaternion,  $\hat{q} = (q_0, q_1, q_2, q_3)$ , that minimize  $\langle ||\hat{q}\mathbf{X}_k\hat{q}^* - \mathbf{Y}_k||^2 \rangle$  in which  $\langle . \rangle$  denotes an average over  $k$ ,  $q^*$  is the conjugate of  $q$  and  $||q||^2 = q^*q$  (see Ref.[213] for more details). The optimal rotation unit quaternion (or orientation quaternion) $\hat{q}$  can be written as  $(\cos(\theta/2), \sin(\theta/2)\hat{u})$ , where  $\theta$  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`.

<sup>1</sup> 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 = alx, aly, alz, a2x, a2y, a2z, a3x, a3y, a3z, ...

  ! an arbitrary integer between 1 to 100
  q_index = n
/

```

Figure 8.4: 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.[213, 214]. 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  $(\cos(\theta/2), \sin(\theta/2)\hat{u})$ , where  $\theta$  is the rotation angle and  $\hat{u}$  is a unit vector associated with the axis of rotation; for example, a rotation of  $90^\circ$  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. 8.4 is an example of Quaternion CVs syntax. An example this type of CVs is presented in the Fig. 8.5. 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,... and another one of 24 atoms with indexes 12,16,46,55,75,... 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  $\theta = 2\cos^{-1}(q_0)$  between the current and the reference positions. This angle is between  $0^\circ$  to  $180^\circ$ . The *cv\_i* list is interpreted as a list of indexes of participating atoms.

$$\text{orientation angle: } \theta = 2\cos^{-1}(q_0)$$

**ORIENTATION\_PROJ:** The cosine of the angle of rotation  $\theta$  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.

$$\text{orientation proj: } 2q_0^2 - 1$$

**SPINANGLE:** Angle of rotation  $\phi$  around a given unit axis  $\hat{e}$ . The axis  $\hat{e}$  is being used to decompose a complete orientation rotation in two sub-rotations, spin  $\phi$  and tilt  $\omega$ . An advantage of this decomposition is  $\phi$  and  $\omega$  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<sup>2</sup>  $\hat{e}$  in  $A^\circ$ . 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*.

$$\text{spin angle: } \phi = 2\tan^{-1}(\mathbf{q} \cdot \mathbf{e} / q_0)$$

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. 8.6. 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 a unit given axis. The tilt angle  $\omega$ , shows a rotation away from the direction  $\hat{e}$ . 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  $0^\circ$  and  $180^\circ$ , 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  $\hat{e}$  ( $\omega = 0^\circ$ ), and the value  $-1$  represents an anti-parallel orientation.

$$\text{tilt: } t = \cos(\omega) = 2\left(\frac{q_0}{\cos(\frac{\tan^{-1}(\mathbf{q} \cdot \mathbf{e})}{q_0})}\right)^2 - 1$$

### 8.3.3 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. [217] 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:

<sup>2</sup>The axis is from the origin(0.0,0.0,0.0) to that point.

<b>path</b>	the steering path whose elements must be real numbers. The <code>path</code> 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.
<b>npath</b>	sets the number of elements in <code>path</code> . Default is 0.
<b>path_mode</b>	The way steering paths are constructed. There are two modes available. In <code>SPLINE</code> mode (default) the path is approximated by a spline that passes through the given points; in <code>LINES</code> mode the path is represented by the line segments joining the control points.
<b>harm</b>	specifies the harmonic constant. If a single number is provided, e.g., <code>harm = 10.0</code> , then it is constant throughout the run. If two or more numbers are provided, e.g., <code>harm = 10.0, 20.0</code> , then the harmonic constant follows a Catmull-Rom spline built upon the provided values.
<b>nharm</b>	sets the number of elements in <code>harm</code> . Default is 0.
<b>harm_mode</b>	The way harmonical paths are constructed, similar with <code>path_mode</code> .

An example of MDIN file and CV.IN file for steered MD is shown in Fig. 8.7. 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.

### 8.3.4 Umbrella sampling

To activate the umbrella sampling code, the `&pmd` namelist 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:

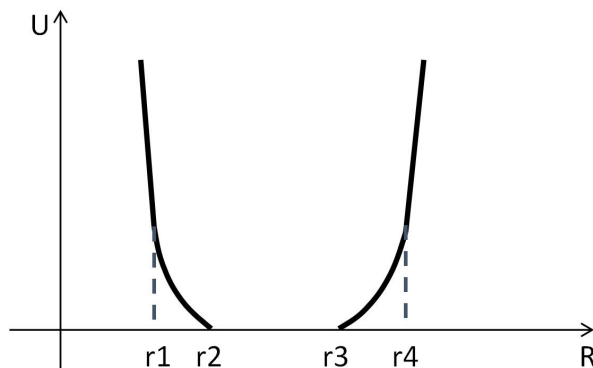
**anchor\_position:** this consists of four real numbers ( $r_1, r_2, r_3, r_4$ ) that determine the rectangle of the umbrella (harmonic) potential. The default value is that all of the  $r$ 's is set to zero.

**anchor\_strength:** two non-negative real numbers ( $k_1, k_2$ ) 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)

- $U = k_1 * (r_1 - r_2) * R \quad (R \leq r_1)$
- $U = 0.5 * k_1 * (R - r_2)^2 \quad (r_1 < R \leq r_2)$
- $U = 0 \quad (r_2 < R \leq r_3)$
- $U = 0.5 * k_2 * (R - r_3)^2 \quad (r_3 < R \leq r_4)$
- $U = k_2 * (r_4 - r_3) * R \quad (R > r_4)$

A plot of the umbrella potential is shown below



**eg1:** if  $r_2 = r_3$ ,  $r_1 \ll r_2$  and  $r_4 \gg r_3$ , then the generated  $U$  is simply the traditional harmonic potential.

**eg2:** if  $r_1$  is slightly less than  $r_2$  and  $r_4$  is slightly larger than  $r_3$ , also with very large  $k_1, k_2$ , the reaction coordinate is restrained in the range  $(r_2, r_3)$  with no potential added.

An example of an MDIN file and CV.IN file for an umbrella sampling simulation is shown in Fig. 8.8. 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<sup>2</sup> (anchor\_strength entry). The second reaction coordinate requested in Fig. 8.8 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<sup>2</sup>. 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. 8.9. The reaction coordinates here are  $\varphi$  and  $\psi$  angles of dialanine.  $\varphi$  is restrained between -2.0 rad and 2.0 rad,  $\psi$  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.

### 8.3.5 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**  $\tau_F$ , 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  $\epsilon$ . Parameter  $\epsilon$  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  $\beta' \rightarrow 0$ ). 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. 8.10.

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  $-1 \text{ \AA}$  and  $10 \text{ \AA}$  and the resolution is set to  $0.5 \text{ \AA}$ . The lower bound ( $-1 \text{ \AA}$ ) 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 `RESTART` 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. 8.10, 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. 8.11). 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 [218]).

**mt19937\_file** the name of the file to which the state of the Mersenne twister is dumped periodically (for restarts).

The MDOUT, MDCRD, RESTART, 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. 191.

### 8.3.6 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"[219][209] 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)[209] 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 implementation 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  $\xi$ . For the  $i^{th}$  iteration, first  $M$  copies of the  $n^{th}$  image are generated around the old center  $\xi_n^{i-1}$  by MD equilibration lasting  $\tau_E$  timesteps. The generated  $M$  copies of the  $n^{th}$  image are expected to be close to  $\xi_n^{i-1}$  for time  $\tau_E$ , assuming that the invoked harmonic constant  $k$  for the restraining potential is large enough. The parameters  $\tau_E$  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  $\tau_R$  timesteps. The newly shifted center  $\xi_n^i$  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  $\varepsilon$  with  $0 \leq \varepsilon \leq 1$  determines the smoothness of the curve; it is recommended that  $\varepsilon$  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  $\tau_R$ . Generally, the large the  $M$  and the shorter  $\tau_R$ , 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  $\xi_n^{i-1}$  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 multiplied 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 equilibration (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_position` 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. 8.12. Here we run 8 images along the path, with `I` defining the image ID. We run 980 MD steps for equilibration 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 be 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. 8.13. 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. 8.14. 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 and reparametrized centers can be reported. Always the reparametrized centers will be extracted to draw the MFEP in the phase space.

### 8.3.7 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
/

```

```

&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 8.6: 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 8.7: 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 8.8: 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 8.9: An example CV.IN file to restrain the  $\varphi$  and  $\psi$  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 8.10: 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 8.11: *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 8.12: 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 8.13: 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 8.14: An example of MDOUT file for STSM run (only part is presented, and some centers are also omitted)



## 9 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 9.7 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.[220]
  - 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](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.

## 9.1 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.]

### 9.1.1 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 generalized 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  $(\mathbf{r}_1 - \mathbf{r}_2) \times (\mathbf{r}_3 - \mathbf{r}_4)$ , 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  $(\mathbf{r}_1 - \mathbf{r}_2) \times (\mathbf{r}_3 - \mathbf{r}_4)$  and  $(\mathbf{r}_5 - \mathbf{r}_6) \times (\mathbf{r}_7 - \mathbf{r}_8)$ . 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 *IAT*1 < 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 ' '*.

## 9 NMR refinement

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. <i>Defaults for ATNAM(1)→ATNAM(8) are ' '</i> .
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. <i>Default = 0.</i>
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). <i>Defaults for NSTEP1, NSTEP2 are both 0.</i>
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. <i>Default is IRSTYP=0.</i>
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 <i>should</i> 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 <i>sander</i> . Default value is 0, or the last value that was explicitly set in a previous restraint. This value is set to 1 if <i>makeDIST_RST</i> is called with the <i>-altdis</i> 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). <i>Default is IFVARI=0.</i>
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). <i>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.</i>
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>i.e.</i> <div style="margin-left: 40px;"> <b>rk2a = R**INCREMENTS * rk2</b>  <b>rk3a = R**INCREMENTS * rk3.</b> </div> INCREMENTS is the number of times the target value changes, which is determined by NSTEP1, NSTEP2, and NINC. <i>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.</i>
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: <ul style="list-style-type: none"> <li>• R &lt; r1 Linear, with the slope of the "left-hand" parabola at the point R=r1.</li> </ul>

- $r1 \leq R < r2$  Parabolic, with restraint energy  $k_2(R - r_2)^2$ .
- $r2 \leq R < r3$   $E = 0$ .
- $r3 \leq R < r4$  Parabolic, with restraint energy  $k_3(R - r_3)^2$ .
- $r4 \leq R$  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  $\pm n \times 360$ , 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-Å<sup>2</sup> Force constants for angles are in kcal/mol-rad<sup>2</sup>. (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  $k_2(R - r_2)^2$ .
- $r2 \leq R < r3$   $E = 0$ .
- $r3 \leq R < r4$  Parabolic, with restraint energy  $k_3(R - r_3)^2$ .
- $r4 \leq R$  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  $ak_3$  at 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  $r1 \rightarrow r4$ ,  $rk2$ , 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  $r1 \rightarrow r4$  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.  $r1(\text{NSTEP1}) = r1$  and  $r1(\text{NSTEP2}) = r1a$ . Note that you *must* specify an explicit value for *nstep1* and *nstep2* if you use this option. Defaults for  $r1 \rightarrow r4, rk2, rk3, r1a \rightarrow r4a, rk2a$  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 desired parabolic well into  $r1 \rightarrow r4$ ,  $rk2$ ,  $rk3$ ,  $r1a \rightarrow r4a$ ,  $rk2a$ , 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)  $\rightarrow$  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  $\tau$  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  $A = \text{RJCOEF}(1)$ ,  $B = \text{RJCOEF}(2)$  and  $C = \text{RJCOEF}(3)$ . In this case, the target values ( $R1 \rightarrow R4$ ,  $R1A \rightarrow R4A$ ) 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→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 outxyz=0. When outxyz is set to 1, the components of the distance and total distance are outputted 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  $\langle r^{-6} \rangle^{-1/6}$  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.

## 9.2 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 "&noexp", followed immediately by standard Amber "group" cards defining the atoms in the submolecule. In addition to the submolecule input ("&noexp"), 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 &noexp namelist:

For each submolecule, the namelist "&noexp" 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.

## 9 NMR refinement

<code>invwt1, invwt2</code>	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 <i>oscale</i> . 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 INVWGT1=INVWGT2=1.0, placing equal weights on all peaks.
<code>omega</code>	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 <i>omega</i> to 600. in the first submolecule.
<code>taurot</code>	Rotational tumbling time of the molecule, in nsec. Default is 1.0 nsec. Like <i>omega</i> , this value is "sticky", so that a value set in one submolecule will remain until it is explicitly reset.
<code>taumet</code>	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.[221] 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.[222] As with <i>omega</i> , <i>taumet</i> can be different for different sub-molecules, but will only change when it is explicitly re-set.
<code>id2o</code>	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 <i>indexn</i> . Alternatively, you can manually rename hydrogens in the <i>prmtop</i> 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 <i>id2o</i> variable retains its value across namelist reads, <i>i.e.</i> its value will only change if it is explicitly reset.
<code>oscale</code>	overall scaling factor between experimental and computed volume units. The experimental intensities are multiplied by <i>oscale</i> 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 <i>oscale</i> variable retains its value across namelist reads, <i>i.e.</i> 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 `&noexp` namelist, give the Amber "group" cards that identify this submolecule. This combination of `&noexp` 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.



## 9.3 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.)

<i>nring</i>	Number of rings in the system.
<i>natr(i)</i>	Number of atoms in the <i>i</i> -th ring.
<i>iatr(j,i)</i>	Absolute atom number for the <i>j</i> -th atom of the <i>i</i> -th ring.
<i>namr(i)</i>	Eight-character string that labels the <i>i</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.
<i>str(i)</i>	Ring current intensity factor for the <i>i</i> -th ring. Older values are summarized by Cross and Wright; <a href="#">[223]</a> more recent empirical parametrizations seem to give improved results. <a href="#">[224, 225]</a>
<i>nprot</i>	Number of protons for which penalty functions are to be set up.
<i>iprot(i)</i>	Absolute atom number of the <i>i</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 <i>wt</i> parameter, described below.
<i>obs(i)</i>	Observed secondary shift for the <i>i</i> -th proton. This is typically calculated as the observed value minus a random coil reference value.
<i>shrang(i)</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.
<i>wt(i)</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 <i>obs</i> entered for the last proton.
<i>shcut</i>	Values of calculated shifts will be printed only if the absolute error between calculated and observed shifts is greater than this value. <i>Default = 0.3 ppm.</i>
<i>nter</i>	Residue number of the N-terminus, for protein shift calculations; <i>default = 1.</i>
<i>cter</i>	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.

## 9.4 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:

$$\delta_{pc}^i = \sum_j \frac{1}{12\pi r_{ij}^3} \left[ \Delta\chi_{ax}^j (3n_{ij}^2 - 1) + (3/2)\Delta\chi_{rh}^j (l_{ij}^2 - m_{ij}^2) \right]$$

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. [226].

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

```

## 9.5 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 *scal*m 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  $\theta_x$  is the angle between the  $x$  axis and the spectrometer field.[227] 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  $\phi_x$  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. [228] 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  $(D_{calc} - D_{obs,u})^2$ . 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:  

$$E_{align}^i = D_{wt}^i (D_{calc}^i - D_{obs(u,l)}^i)^2$$
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.[228]
- 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  $S_{33}$  is calculated as  $-(S_{11}+S_{22})$ . 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 ratios by  $\gamma_N = g_N \beta_N / \hbar$ . Common values are  $^1\text{H} = 5.5856$ ,  $^{13}\text{C} = 1.4048$ ,  $^{15}\text{N} = -0.5663$ ,  $^{31}\text{P} = 2.2632$ .

<code>dij</code>	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 <i>dij</i> 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. <a href="#">[229]</a>
<code>dcut</code>	Controls printing of calculated and observed dipolar couplings. Only values where $\text{abs}(\text{dobs}(u,l) - \text{dcalc})$ is greater than <i>dcut</i> will be printed. Default is 0.1 Hz. Set to a negative value to print all dipolar restraint information.
<code>freezemol</code>	If this is set to <i>.true.</i> , 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 <i>.false.</i> .

## 9.6 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 constraints. This section shows how to input these sorts of restraints. The entry of the alignment tensor is done as in Section 9.5, 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.

<code>ncsa</code>	Number of observed residual CSA peaks to be used as restraints.
<code>icsa, jcsa, kcsa</code>	Atom numbers for the csa of interest: <i>jcsa</i> is the atom whose $\Delta\sigma$ value has been measured; <i>icsa</i> and <i>kcsa</i> are two atoms bonded to it, used to define the local axis frame for the CSA tensor. See <i>amber12/test/pcsa/RST.csa</i> for examples of how to set these.
<code>cobsl, cobsu</code>	Limiting values for the observed residual CSA, in Hz (not ppm or ppb!). If the calculated value of $\Delta\sigma$ is less than <i>cobsl</i> , the energy penalty is proportional to $(\Delta\sigma_{\text{calc}} - \Delta\sigma_{\text{obs},l})^2$ ; if it is larger than <i>cobsu</i> , the penalty is proportional to $(\Delta\sigma_{\text{calc}} - \Delta\sigma_{\text{obs},u})^2$ . Calculated values between <i>cobsl</i> and <i>cobsu</i> are not penalized. Note that <i>cobsl</i> must be less than <i>cobsu</i> .
<code>cwt</code>	The relative weight of each observed value. Default is 1.0. The penalty function is thus: $E_{\text{csa}}^i = C_{\text{wt}}^i (\Delta\sigma_{\text{calc}}^i - \Delta\sigma_{\text{obs}(u,l)}^i)^2$ <p>where <math>C_{\text{wt}}</math> 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.</p>
<code>datasetc</code>	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.
<code>field</code>	Magnetic field (in MHz) for the residual CSA being considered here. This is indexed from 1 to <i>ncsa</i> , and is nucleus dependent. For example, if the proton frequency is 600 MHz, then <i>field</i> for $^{13}\text{C}$ would be 150, and that for $^{15}\text{N}$ would be 60.
<code>sigma11, sigma22, sigma12, sigma13, sigma23</code>	Values of the CSA tensor (in ppm) for atom <i>icsa</i> , in the local coordinate frame defined by atoms <i>icsa</i> , <i>jcsa</i> and <i>kcsa</i> . See <i>\$AMBERHOME/test/pcsa/RST.csa</i> for examples of how to set these.

`ccut` Controls printing of calculated and observed residual CSAs. Only values where  $\text{abs}(\text{cobs}(u,l) - \text{ccalc})$  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.

## 9.7 Preparing restraint files for Sander

Fig. 9.1 shows the general information flow for auxiliary programs that help prepare the restraint files. Once the restraint files are made, Fig. 9.2 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.[220, 230–234] 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.

### 9.7.1 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
-uall <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:

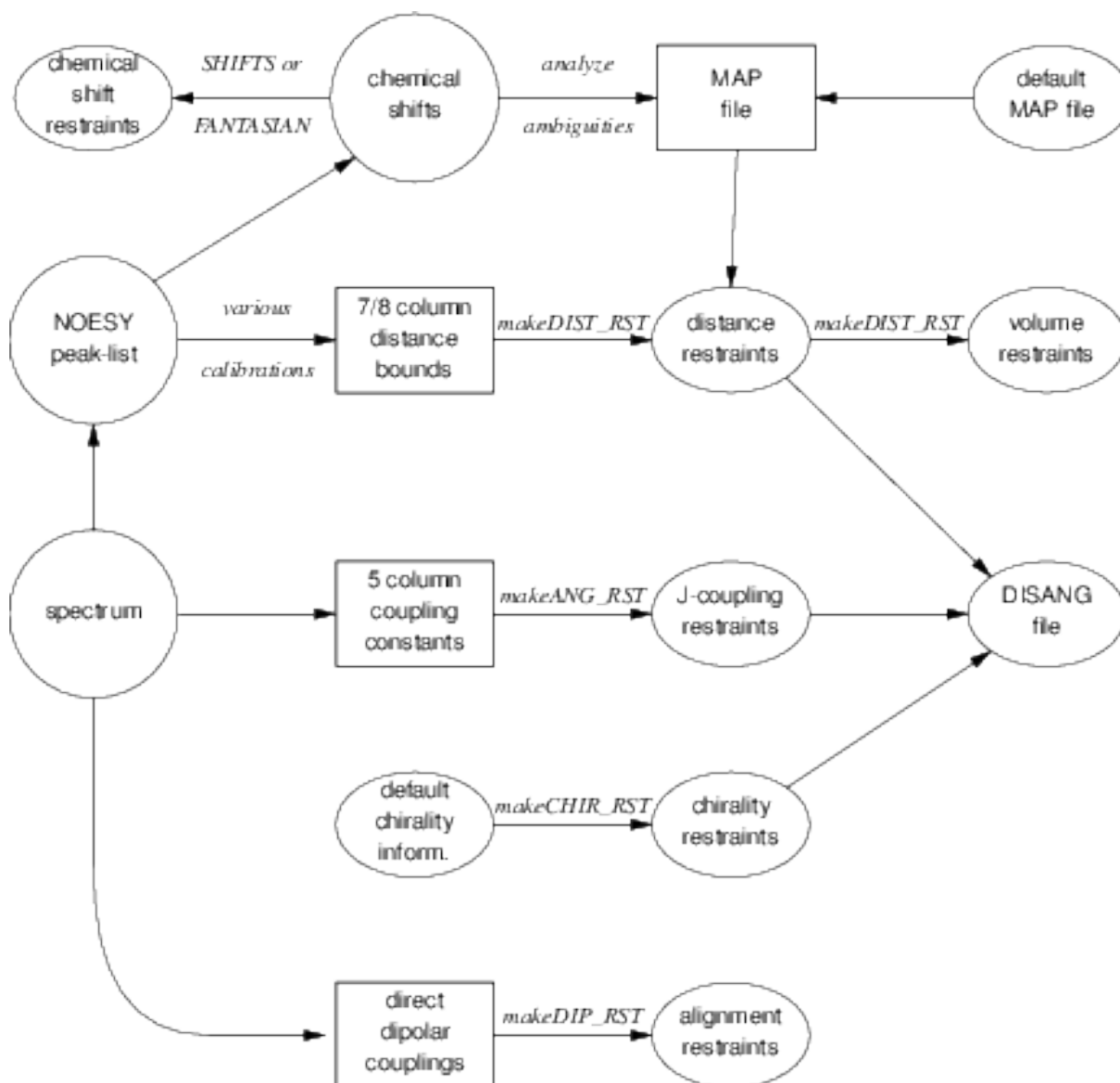


Figure 9.1: Notation: circles represent logical information, whose format might differ from one project to the next; solid rectangles are in a specific format (largely compatible with DIANA and other programs), and are intended to be read and edited by the user; ellipses are specific to sander, and are 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

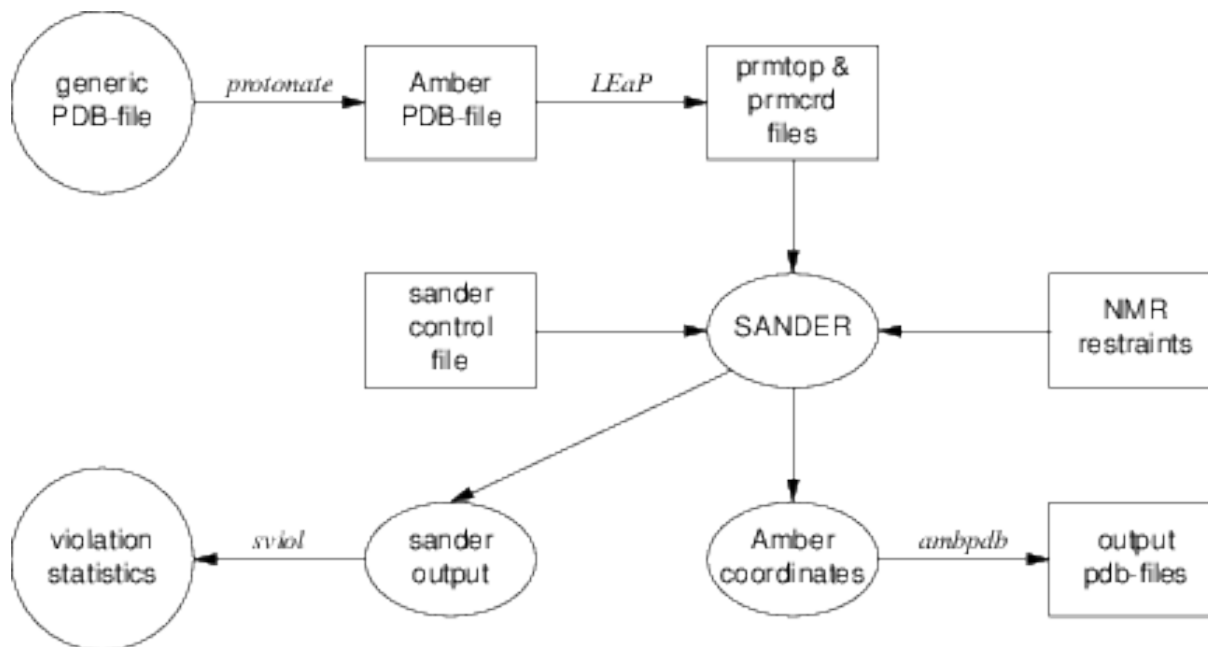


Figure 9.2: General organization of NMR refinement calculations.

**23 ALA HA 52 VAL H 3.2 3.8 # comments go here**

Here the lower bound is 3.2 Å and 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 *n timer>* 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 -a atm* 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 Å upper 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*.

### 9.7.2 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 number in the  $ub$  column,  $360^\circ$  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.

### 9.7.3 Chirality restraints: makeCHIR\_RST

**Usage:** `makeCHIR_RST <pdb-file> <output-constraint-file>`

We also find it useful to add chirality constraints and *trans*-peptide  $\omega$  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.

### 9.7.4 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 C $\alpha$ H 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 ( $0 \leq S_2 \leq 1$ ), 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.

### 9.7.5 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.
```

## 9.8 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.)

## 9.9 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

$$\bar{r} = (1/C) \left\{ \int_0^t e^{(t'-t)/\tau} r(t')^{-i} dt' \right\}^{-1/i} \quad (9.1)$$

where

- $\bar{r}$  = time-averaged value of the internal coordinate (distance or angle)
- $t$  = the current time
- $\tau$  = 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

$$\langle \phi \rangle = \tan^{-1} (\langle \sin(\phi) \rangle / \langle \cos(\phi) \rangle)$$

where  $\phi$  is the torsion, and  $\langle \sin(\phi) \rangle$  and  $\langle \cos(\phi) \rangle$  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),

$$\partial E / \partial x = (\partial E / \partial \bar{r})(\partial \bar{r} / \partial r(t))(\partial r(t) / \partial x) \quad (9.2)$$

(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

$$\partial E / \partial x = (\partial E / \partial \bar{r})(\partial r(t) / \partial x) \quad (9.3)$$

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. 9.2, 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. 9.2], 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. 9.3 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.[235–239]

## 9.10 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.

## 9.11 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.

### 9.11.1 1. Simulated annealing NMR refinement

```
15ps simulated annealing protocol
&cntrl
  nstlim=15000, ntt=1, !(time limit, temp. control)
  ntpcr=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.

## 9.11.2 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

```

### 9.11.3 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*.

### 9.11.4 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.

### 9.11.5 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*',
/

```



## 10 Xray and cryoEM refinement

### 10.1 EMAP restraints for rigid and flexible fitting into EM maps

EMAP restrained simulation[19, 240] 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[240–250].

With EMAP restraints, Sander and PMEMD can be used to perform both rigid[240] and flexible[19] 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 7.1 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, <i>mapfile</i> =”, 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 <i>move</i> >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 <i>ifit</i> >0. One would do this when the initial coordinates don’t match those of the map (default=0). When <i>ifit</i> =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 ( <i>mapfit</i> =...) that matches the (final) simulation coordinates, <i>and/or</i> output of the coordinates ( <i>molfit</i> =...) that would match the orientation of the original map. When <i>ifit</i> =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, <i>ifit</i> =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[240]. For example, <i>grids</i> =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).

**mapfit** The filename for the final constraint map after rigid fitting and/or moving. The filename must have 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/lgb1.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/gbln_1.ccp4',  (final map)
molfit='scratch/gbln_1.pdb', /  (final restrained atoms related to initial map)
&emap          (EMAP restraint 2)
mapfile='data/lgb1.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/gblh_1.ccp4',  (final map, same as initial)
molfit='scratch/gblh_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/gblc_1.ccp4',  (final map)
molfit='scratch/gblc_1.pdb', /  (final restrained atoms related to initial map)
```

## 10.2 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 is 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.[251]

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.

### 10.2.1 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
```

### 10.2.2 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
```

### 10.2.3 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 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 Ångströms 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.

### 10.2.4 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.

## 10.3 X-ray functionality and diffraction-based restraints

The *msander* program includes a new 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”[252]. For information on how to set up a crystal simulation, including periodic boundary conditions to emulate crystalline lattice, see Chapter 10.2. 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[253]. This was accomplished by using the python API to *sander*, and uses locally-enhanced sampling (see Chap. ??) 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.

### 10.3.1 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[254]; 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[255], which calculates the contribution from interstitial solvent into SFs using two generic parameters (*k\_sol* and *b\_sol*), as reported by Fokine and Urzhumtsev[256]. The implementation, including the scheme to build solvent mask, is analogous to the one in cctbx library[257]. The more advanced version (*bulk\_solvent\_model* = ‘afonine-2013’) employs the variable *k\_mask*, which replaces *k\_sol* and *b\_sol*; this variable is automatically optimized over the individual resolution bins[258]. 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. 10.2) 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*).

### 10.3.2 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):

$$E_{total} = E_{force-field} + w_{xray} E_{xray} \quad (10.1)$$

$E_{xray}$  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 amplitudes and of calculated and observed structure factors:

$$E_{xray} = \frac{\sum_{h,k,l} (F_{calc}(h,k,l) - F_{obs}(h,k,l))^2}{\sum_{h,k,l} F_{obs}^2(h,k,l)} \quad (10.2)$$

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’)[259, 260]:

$$E_{xray} = \sum_{h,k,l} \left( -\ln \left( \frac{2F_{obs}(h,k,l)}{\epsilon\beta} \right) + \frac{F_{obs}^2(h,k,l)}{\epsilon\beta} + \frac{\alpha^2 F_{calc}^2(h,k,l)}{\epsilon\beta} - \ln I_0 \left( \frac{2\alpha F_{obs}(h,k,l) F_{calc}(h,k,l)}{\epsilon\beta} \right) \right) \quad (10.3)$$

where  $\alpha$  and  $\beta$  are (resolution-shell-dependent) ML likelihood distribution parameters,  $\epsilon$  is the symmetry coefficient ( $\epsilon = 1$  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.9, 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[261, 262]. Another promising application is refinement of crystallographic structures. Such an Amber-based protocol has been reported by Mikhailovskii et al.[263] (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.

### 10.3.3 Inputs and file formats

System setup follows the general procedures outlined in Chap. 10.2. 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. 10.2) that contains the B-factors. The structure factors have to be listed in the *reflection\_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:**

<code>pdb_infile</code>	name of the PDB input file containing B-factors
<code>pdb_read_coordinates</code>	if true, use coordinates from the PDB file, not inpcrd, as starting coordinates
<code>pdb_outfile</code>	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 <i>pdb_infile</i>
<code>reflection_infile</code>	name of the input file containing experimental SFs and R-flags
<code>fmtz_outfile</code>	name of the file with calculated SFs, in “formatted MTZ” format

**Bulk solvent parameters:**

<code>bulk_solvent_model</code>	the type of bulk solvent to use. Possible values: ‘none’ for disabled bulk solvent contribution, ‘simple’ or ‘afonine-2013’ (default)
<code>k_sol</code>	solvent electron density (default $0.35 \text{ e } \text{\AA}^{-3}$ )
<code>b_sol</code>	determines the blurring of the boundary between the solvent region and the macromolecule (default $46 \text{ \AA}^2$ )
<code>solvent_mask_adjustment</code>	increment to be added to atomic radii of the atoms selected by <i>atom_selection_mask</i> as a part of the algorithm to build bulk solvent mask (default $1.11 \text{ \AA}$ )
<code>solvent_mask_probe_radius</code>	the radius of solvent probe to apply as a part of the algorithm to build bulk solvent mask (default $0.9 \text{ \AA}$ )
<code>mask_update_period</code>	time interval for bulk solvent grid update (expressed as a multiple of integration step, default 100 steps)

**Structure-factor-based potential:**

<code>atom_selection_mask</code>	ambmask-format mask to specify the atoms that contribute to $F_{calc}$ via direct summation formula (default ‘!@H=’)
----------------------------------	--

## 10 Xray and cryoEM refinement

<code>scale_update_period</code>	time interval to re-scale $F_{calc}$ to $F_{obs}$ (expressed as a multiple of integration step, default 100 steps)
<code>target</code>	the type of crystallographic target function. Possible values: 'ls' for Least Squares, 'ml' for Maximum Likelihood (default)
<code>ml_update_period</code>	time interval to update ML parameters $\alpha$ and $\beta$ (expressed as a multiple of integration step, default 100 steps)
<code>xray_weight</code>	Sets the value of $w_{xray}$ (default 1.0) This value can be variable: see <a href="#">Section 1.10</a> for more information.
<code>xray_weight_final</code>	final value that defines linear scaling of $w_{xray}$ weight along the trajectory (if unassigned, assumed to be equal to <code>xray_weight_initial</code> )

### 10.4 Preparing inputs X-ray refinement

Coming(?)....

## Bibliography

- [1] 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.
- [2] 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.
- [3] 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.
- [4] 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.
- [5] 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.
- [6] 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.
- [7] T. Morishita. Fluctuation formulas in molecular-dynamics simulations with the weak coupling heat bath. *J. Chem. Phys.*, **2000**, *113*, 2976.
- [8] A. Mudi; C. Chakravarty. Effect of the Berendsen thermostat on the dynamical properties of water. *Mol. Phys.*, **2004**, *102*, 681–685.
- [9] 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.
- [10] 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.
- [11] 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.
- [12] Y. M. Rhee; V. S. Pande. Solvent viscosity dependence of the protein folding dynamics. *J. Phys. Chem. B*, **2008**, *112*, 6221–6227. PMID: 18229911.
- [13] J. A. Izaguirre; D. P. Catarello; J. M. Wozniak; R. D. Skeel. Langevin stabilization of molecular dynamics. *J. Chem. Phys.*, **2001**, *114*, 2090–2098.
- [14] 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.
- [15] H. C. Andersen. Molecular dynamics simulations at constant pressure and/or temperature. *J. Chem. Phys.*, **1980**, *72*, 2384–2393.
- [16] 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.

## BIBLIOGRAPHY

- [17] 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.
- [18] 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.
- [19] 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.
- [20] 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.
- [21] 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.
- [22] 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.
- [23] 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.
- [24] 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.
- [25] 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.
- [26] 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.
- [27] X. Wu; B. R. Brooks. Isotropic periodic sum: A method for the calculation of long-range interactions. *J. Chem. Phys.*, **2005**, 122, 044107.
- [28] 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.
- [29] K. Takahashi; K. Yasuoka; T. Narumi. Cutoff radius effect of isotropic periodic sum method for transport. *J. Chem. Phys.*, **2007**, 127, 114511.
- [30] 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.
- [31] X. Wu; B. R. Brooks. Isotropic periodic sum of electrostatic interactions for polar systems. *J. Chem. Phys.*, **2009**, 131, 024107.
- [32] 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.
- [33] 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.
- [34] M. Feig. Kinetics from Implicit Solvent Simulations of Biomolecules as a Function of Viscosity. *J. Chem. Theory Comput.*, **2007**, 3, 1734–1748.
- [35] 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.

- [36] 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.
- [37] A. V. Onufriev; D. A. Case. Generalized Born implicit solvent models for biomolecules. *Annu. Rev. Biophys.*, **2019**, *48*, 275–296.
- [38] 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.
- [39] 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.
- [40] M. Schaefer; M. Karplus. A comprehensive analytical treatment of continuum electrostatics. *J. Phys. Chem.*, **1996**, *100*, 1578–1599.
- [41] 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.
- [42] 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.
- [43] C. J. Cramer; D. G. Truhlar. Implicit solvation models: Equilibria, structure, spectra, and dynamics. *Chem. Rev.*, **1999**, *99*, 2161–2200.
- [44] D. Bashford; D. A. Case. Generalized Born models of macromolecular solvation effects. *Annu. Rev. Phys. Chem.*, **2000**, *51*, 129–152.
- [45] A. Onufriev; D. Bashford; D. A. Case. Modification of the generalized Born model suitable for macromolecules. *J. Phys. Chem. B*, **2000**, *104*, 3712–3720.
- [46] M. S. Lee; F. R. Salsbury, Jr.; C. L. Brooks, III. Novel generalized Born methods. *J. Chem. Phys.*, **2002**, *116*, 10606–10614.
- [47] 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.
- [48] 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.
- [49] N. Calimet; M. Schaefer; T. Simonson. Protein molecular dynamics with the generalized Born/ACE solvent model. *Proteins*, **2001**, *45*, 144–158.
- [50] 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.
- [51] 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.
- [52] 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.
- [53] 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.
- [54] F. M. Richards. Areas, volumes, packing, and protein structure. *Ann. Rev. Biophys. Bioeng.*, **1977**, *6*, 151–176.

## BIBLIOGRAPHY

- [55] 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.
- [56] 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.
- [57] 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.
- [58] 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.
- [59] 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.
- [60] 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.
- [61] H. Nguyen; D. R. Roe; C. Simmerling. Improved Generalized Born Solvent Model Parameters for Protein Simulations. *J. Chem. Theory Comput.*, **2013**, 9, 2020–2034.
- [62] 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.
- [63] 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.
- [64] A. Onufriev. in *Modeling Solvent Environments*, M. Feig, Ed., (Wiley, USA). pp 127–165. 2010.
- [65] 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.
- [66] M. Schaefer; H. W. T. Van Vlijmen; M. Karplus. Electrostatic contributions to molecular free energies in solution. *Adv. Protein Chem.*, **1998**, 51, 1–57.
- [67] 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.
- [68] 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.
- [69] 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.
- [70] 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.
- [71] D. Sitkoff; K. A. Sharp; B. Honig. Accurate calculation of hydration free energies using macroscopic solvent models. *J. Phys. Chem.*, **1994**, 98, 1978–1988.
- [72] 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.

- [73] D. Chandler; H. C. Andersen. Optimized cluster expansions for classical fluids. ii. theory of molecular liquids. *J. Chem. Phys.*, **1972**, 57, 1930–1937.
- [74] F. Hirata; P. J. Rossky. An extended RISM equation for molecular polar fluids. *Chem. Phys. Lett.*, **1981**, pp 329–334.
- [75] 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.
- [76] 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.
- [77] D. Chandler; J. McCoy; S. Singer. Density functional theory of nonuniform polyatomic systems. i. general formulation. *J. Chem. Phys.*, **1986**, 85, 5971–5976.
- [78] 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.
- [79] 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.
- [80] 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.
- [81] 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.
- [82] 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.
- [83] A. Kovalenko. In Hirata [644], chapter 4.
- [84] 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.
- [85] 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.
- [86] J.-P. Hansen; I. R. McDonald. *Theory of simple liquids*. Academic Press, London, 1990.
- [87] F. Hirata. In *Molecular Theory of Solvation* [644], chapter 1.
- [88] 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.
- [89] 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.
- [90] 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.
- [91] S. J. Singer; D. Chandler. Free energy functions in the extended RISM approximation. *Mol. Phys.*, **1985**, 55, 621–625.
- [92] 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.

## BIBLIOGRAPHY

- [93] S. M. Kast. Free energies from integral equation theories: Enforcing path independence. *Phys. Rev. E*, **2003**, 67, 041203.
- [94] 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.
- [95] 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.
- [96] 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.
- [97] 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.
- [98] 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.
- [99] 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.
- [100] 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.
- [101] S. Genheden; T. Luchko; S. Gusarov; A. Kovalenko; U. Ryde. An MM/3D-RISM approach for ligand-binding affinities. *J. Phys. Chem.*, **2010**. Accepted.
- [102] 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**.
- [103] 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.
- [104] 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.
- [105] 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.
- [106] T. Kruger; M. Elstner; P. Schiffels; T. Frauenheim. Validation of the density-functional based tight-binding approximation. *J. Chem. Phys.*, **2005**, 122, 114110.
- [107] 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.
- [108] J. J. P. Stewart. Optimization of parameters for semiempirical methods I. Method. *J. Comput. Chem.*, **1989**, 10, 209–220.
- [109] 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.
- [110] 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.
- [111] 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.



- [112] M. P. Repasky; J. Chandrasekhar; W. L. Jorgensen. PDDG/PM3 and PDDG/MNDO: Improved semiempirical methods. *J. Comput. Chem.*, **2002**, 23, 1601–1622.
- [113] 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.
- [114] 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.
- [115] 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.
- [116] 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.
- [117] 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.
- [118] 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.
- [119] P. Imhof; F. Noé; S. Fischer; J. C. Smith. AM1/d Parameters for Magnesium in Metalloenzymes. *J. Chem. Theory Comput.*, **2006**, 2, 1050–1056.
- [120] 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.
- [121] 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.
- [122] 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.
- [123] 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.
- [124] 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.
- [125] 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.
- [126] 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.
- [127] 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.
- [128] M. Korth. Third-generation hydrogen-bonding corrections for semiempirical qm methods and force fields. *J. Chem. Theory Comput.*, **2010**, 6, 3808.
- [129] 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.
- [130] A. Bondi. van der Waals volumes and radii. *J. Phys. Chem.*, **1964**, 68, 441–451.

## BIBLIOGRAPHY

- [131] 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.
- [132] E. Pellegrini; M. J. Field. A generalized-Born solvation model for macromolecular hybrid-potential calculations. *J. Phys. Chem. A.*, **2002**, 106, 1316–1326.
- [133] 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.
- [134] 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.
- [135] 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>.
- [136] 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.
- [137] 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.
- [138] ADF2011, SCM, Theoretical Chemistry, Vrije Universiteit, Amsterdam, The Netherlands, <http://www.scm.com>, 2012.
- [139] 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.
- [140] 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, Amsterdam, 2005.
- [141] 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.
- [142] 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.
- [143] F. Neese. ORCA - an ab initio, Density Functional and Semiempirical program package, Version 2.8.0, University of Bonn, 2010.
- [144] 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.
- [145] 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.
- [146] 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.
- [147] 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.
- [148] 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>.
- [149] 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](http://www.mrcc.hu).
- [150] Z. Rolik; L. Szegedy; I. Ladjánszki; B. Ladoćzki; M. Kállay. An efficient linear-scaling CCSD(T) method based on local natural orbitals. *J. Chem. Phys.*, **2013**, 139, 094105.
- [151] 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.
- [152] 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.
- [153] 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.
- [154] B. Hégyel; 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.
- [155] 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.
- [156] 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.
- [157] X. Wu; B. R. Brooks. Self-guided Langevin dynamics simulation method. *Chem. Phys. Lett.*, **2003**, 381, 512–518.
- [158] 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.
- [159] 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.

## BIBLIOGRAPHY

- [160] X. Wu; B. R. Brooks. Self-guided langevin dynamics via generalized langevin equation. *J. Comput. Chem.*, **2016**, 37, 595–601.
- [161] X. Wu; B. R. Brooks. Toward canonical ensemble distribution from self-guided Langevin dynamics simulation. *J. Chem. Phys.*, **2011**, 134, 134108.
- [162] 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.
- [163] 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.
- [164] I. Kolossváry; W. C. Guida. Low-mode conformational search elucidated: Application to C<sub>39</sub>H<sub>80</sub> and flexible docking of 9-deazaguanine inhibitors into PNP. *J. Comput. Chem.*, **1999**, 20, 1671–1684.
- [165] 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.
- [166] 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.
- [167] 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.
- [168] D. C. Liu; J. Nocedal. On the limited memory method for large scale optimization. *Math. Programming B*, **1989**, 45, 503–528.
- [169] J. Nocedal; J. L. Morales. Automatic preconditioning by limited memory quasi-Newton updating. *SIAM J. Opt.*, **2000**, 10, 1079–1096.
- [170] P. Kollman. Free energy calculations: Applications to chemical and biochemical phenomena. *Chem. Rev.*, **1993**, 93, 2395–2417.
- [171] T. Simonson. in *Computational Biochemistry and Biophysics*, O. Becker; A. D. MacKerell; B. Roux; M. Watanabe, Eds. Marcel Dekker, New York, 2001.
- [172] D. Frenkel; B. Smit. *Understanding Molecular Simulation: From Algorithms to Applications. Second edition*. Academic Press, San Diego, 2002.
- [173] 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.
- [174] 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.
- [175] G. Hummer; A. Szabo. Calculation of free-energy differences from computer simulations of initial and final states. *J. Chem. Phys.*, **1996**, 105, 2004–2010.
- [176] 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.
- [177] J. Kaus; L. C. T. Pierce; R. C. Walker; J. A. McCammon. PMEMD TI: Placeholder. *J. Chem. Theory Comput.*, **2013**.
- [178] 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.

- [179] V. Babin; C. Roland; C. Sagui. Adaptively biased molecular dynamics for free energy calculations. *J. Chem. Phys.*, **2008**, *128*, 134101.
- [180] 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.
- [181] F. Wang; D. P. Landau. Efficient, multiple-range random walk algorithm to calculate the density of states. *Phys. Rev. Lett.*, **2001**, *86*, 2050–2053.
- [182] A. Laio; M. Parrinello. Escaping free-energy minima. *Proc. Natl. Acad. Sci.*, **2002**, *99*, 12562–12566.
- [183] 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.
- [184] T. Lelièvre; M. Rousset; G. Stoltz. Computation of free energy profiles with parallel adaptive dynamics. *J. Chem. Phys.*, **2007**, *126*, 134111.
- [185] 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.
- [186] Y. Sugita; A. Kitao; Y. Okamoto. Multidimensional replica-exchange method for free-energy calculations. *J. Chem. Phys.*, **2000**, *113*, 6042–6051.
- [187] G. Bussi; F. L. Gervasio; A. Laio; M. Parrinello. Free-energy landscape for  $\beta$  hairpin folding from combined parallel tempering and metadynamics. *J. Am. Chem. Soc.*, **2006**, *128*, 13435–13441.
- [188] S. Piana; A. Laio. A bias-exchange approach to protein folding. *J. Phys. Chem. B*, **2007**, *111*, 4553–4559.
- [189] 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.
- [190] V. Babin; V. Karpusenko; 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.
- [191] 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.
- [192] 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.
- [193] 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.
- [194] M. Moradi; V. Babin; C. Sagui; C. Roland. in *Proline: Biosynthesis, Regulation and Health Benefits*, B. Nedjimi, Ed., pp 67–110. Nova Publishers, 2013.
- [195] 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.
- [196] 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.
- [197] V. Babin; C. Roland; C. Sagui. The alpha-sheet: A missing-in-action secondary structure? *Proteins –Structure Function and Bioinformatics*, **2011**, *79*, 937–946.
- [198] 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.

## BIBLIOGRAPHY

- [199] 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.
- [200] 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.
- [201] 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.
- [202] 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.
- [203] 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.
- [204] 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.
- [205] M. Moradi; C. Sagui; C. Roland. Calculating relative transition rates with driven nonequilibrium simulations. *Chem. Phys. Lett.*, **2011**, *518*, 109.
- [206] M. Moradi; C. Sagui; C. Roland. Investigating rare events with nonequilibrium work measurements: I. Nonequilibrium transition paths. *J. Chem. Phys.*, **2014**, *140*, 034114.
- [207] M. Moradi; C. Sagui; C. Roland. Investigating rare events with nonequilibrium work measurements: II. Transition and reaction rates. *J. Chem. Phys.*, **2014**, *140*, 034115.
- [208] M. Moradi; E. Tajkhorshid. Driven Metadynamics: Reconstructing equilibrium free energies from driven adaptive-bias simulations. *J. Phys. Chem. Lett.*, **2013**, *4*, 1882.
- [209] 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.
- [210] 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.
- [211] 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.
- [212] E. A. Coutsiyas; C. Seok; K. A. Dill. Using quaternions to calculate RMSD. *J. Comput. Chem.*, **2004**, *25*, 1849–1857.
- [213] D. K. Coutsiyas EA, Seok C. Using quaternions to calculate rmsd. *J Comput Chem*, **2004 Nov 30**, *25*, 1849–57.
- [214] G. Fiorin; M. L. Klein; J. Hénin. Using collective variables to drive molecular dynamics simulations. *Molecular Physics*, **2013**, *111*, 3345–3362.
- [215] 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.
- [216] 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.
- [217] 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.
- [218] M. Matsumoto; T. Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, **1998**, *8*, 3–30.

- [219] 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.
- [220] 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.
- [221] A. Kalk; H. J. C. Berendsen. Proton magnetic relaxation and spin diffusion in proteins. *J. Magn. Reson.*, **1976**, *24*, 343–366.
- [222] E. T. Olejniczak; M. A. Weiss. Are methyl groups relaxation sinks in small proteins? *J. Magn. Reson.*, **1990**, *86*, 148–155.
- [223] K. J. Cross; P. E. Wright. Calibration of ring-current models for the heme ring. *J. Magn. Reson.*, **1985**, *64*, 220–231.
- [224] K. Ösabay; D. A. Case. A new analysis of proton chemical shifts in proteins. *J. Am. Chem. Soc.*, **1991**, *113*, 9436–9444.
- [225] D. A. Case. Calibration of ring-current effects in proteins and nucleic acids. *J. Biomol. NMR*, **1995**, *6*, 341–346.
- [226] 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.
- [227] 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.
- [228] 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.
- [229] D. A. Case. Calculations of NMR dipolar coupling strengths in model peptides. *J. Biomol. NMR*, **1999**, *15*, 95–102.
- [230] 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.
- [231] D. A. Case; P. E. Wright. in *NMR in Proteins*, G. M. Clore; A. Gronenborn, Eds., pp 53–91. MacMillan, New York, 1993.
- [232] 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.
- [233] R. Brüschweiler; D. A. Case. Characterization of biomolecular structure and dynamics by NMR cross-relaxation. *Prog. NMR Spectr.*, **1994**, *26*, 27–58.
- [234] D. A. Case. The use of chemical shifts and their anisotropies in biomolecular structure determination. *Curr. Opin. Struct. Biol.*, **1998**, *8*, 624–630.
- [235] A. E. Torda; R. M. Scheek; W. F. VanGunsteren. Time-dependent distance restraints in molecular dynamics simulations. *Chem. Phys. Lett.*, **1989**, *157*, 289–294.
- [236] 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.
- [237] 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.
- [238] D. A. Pearlman. How well to time-averaged J-coupling restraints work? *J. Biomol. NMR*, **1994**, *4*, 279–299.

## BIBLIOGRAPHY

- [239] 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.
- [240] 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.
- [241] 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.
- [242] X. Wu; B. R. Brooks. Modeling of Macromolecular assemblies with map objects. *Proc. 2007 Int. Conf. Bioinform. Comput. Biol.*, **2007**, II, 411–417.
- [243] 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.
- [244] 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.
- [245] 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.
- [246] 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.
- [247] 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.
- [248] 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.
- [249] X. Wu; B. R. Brooks. in *Modern electron microscopy in physical and life science*, M. Janecek, Ed., chapter 12, pp 243–262. InTech, 2016.
- [250] 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.
- [251] D. S. Cerutti; D. A. Case. Molecular dynamics simulations of macromolecular crystals. *Wires Comput. Mol. Sci.*, **2018**, 8, e1402.
- [252] 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.
- [253] 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.
- [254] P. Afonine; A. Urzhumtsev. On a fast calculation of structure factors at a subatomic resolution. *Acta Cryst. A*, **2004**, 60, 19–32.
- [255] 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.



- [256] A. Fokine; A. Urzhumtsev. Flat bulk-solvent model: obtaining optimal parameters. *Acta Cryst. D*, **2002**, 58, 1387–1392.
- [257] 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.
- [258] 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.
- [259] V. Lunin; T. Skovoroda. R-free likelihood-based estimates of errors for phases calculated from atomic models. *Acta Cryst. A*, **1995**, 51, 880–887.
- [260] 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.
- [261] Y. Xue; N. Skrynnikov. Ensemble MD simulations restrained via crystallographic data: accurate structure leads to accurate dynamics. *Prot. Sci.*, **2014**, 23, 488–507.
- [262] 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.
- [263] 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.
- [264] J. I. Mendieta-Moreno; R. C. Walker; J. P. Lewis; P. Gómez-Puertas; J. Mendieta; J. Ortega. FIREBAL-L/AMBER: An efficient local-orbital DFT QM/MM method for biomolecular systems. *J. Chem. Theory Comput.*, **2014**, 10, 2185–2193.
- [265] S. N. Steinmann; R. Ferreira de Moraes; A. W. Götz; P. Fleurat-Lessard; M. Iannuzzi; P. Sautet; C. Michel. *J. Chem. Theory Comput.*, **2018**.
- [266] S. N. Steinmann; P. Fleurat-Lessard; A. W. Götz; C. Michel; R. Ferreira de Moraes; 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.
- [267] 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.
- [268] 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.
- [269] 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.
- [270] 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.
- [271] 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.
- [272] 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.
- [273] R. E. Buló; 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.
- [274] R. E. Buló; B. Ensing; J. Sikkema; L. Visscher. Toward a practical method for adaptive qm/mm simulations. *J. Chem. Theory Comput.*, **2009**, 9, 2212–2221.

## BIBLIOGRAPHY

- [275] 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.
- [276] 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.
- [277] 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.
- [278] 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.
- [279] 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.
- [280] 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.
- [281] D. A. Pearlman; S.-H. Kim. Conformational Studies of Nucleic Acids. V. Sequence Specificities of in the Conformational Energetics of Oligonucleotides: The Homo-Tetramers. *Biopolymers*, **1988**, 27, 59–77.
- [282] 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.
- [283] 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.
- [284] R. Lavery. in *Unusual DNA Structures*, R. D. Wells; S. C. Harvey, Eds. Springer-Verlag, New York, 1988.
- [285] 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.
- [286] J. M. Hubbard; J. E. Hearst. Predicting the Three-Dimensional Folding of Transfer RNA with a Computer Modeling Protocol. *Biochemistry*, **1991**, 30, 5458–5465.
- [287] S.-H. Chou; L. Zhu; B. R. Reid. The Unusual Structure of the Human Centromere (GGA)<sub>2</sub> Motif. *J. Mol. Biol.*, **1994**, 244, 259–268.
- [288] M. Levitt. Detailed Molecular Model for Transfer Ribonucleic Acid. *Nature*, **1969**, 224, 759–763.
- [289] M. S. Babcock; E. P. D. Pednault; W. K. Olson. Nucleic Acid Structure Analysis. *J. Mol. Biol.*, **1994**, 237, 125–156.
- [290] J. M. Hubbard; J. E. Hearst. Computer Modeling 16S Ribosomal RNA. *J. Mol. Biol.*, **1991**, 221, 889–907.
- [291] 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.
- [292] T. Schlick; W. K. Olson. Supercoiled DNA Energetics and Dynamics by Computer Simulation. *J. Mol. Biol.*, **1992**, 223, 1089–1119.
- [293] R. E. Dickerson. Definitions and Nomenclature of Nucleic Acid Structure Parameters. *J. Biomol. Struct. Dyn.*, **1989**, 6, 627–634.
- [294] S. R. Holbrook; J. L. Sussman; R. W. Warrant; S.-H. Kim. Crystal Structure of Yeast Phenylalanine Transfer RNA II. Structural Features and Functional Implications. *J. Mol. Biol.*, **1978**, 123, 631–660.
- [295] 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.

- [296] 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.
- [297] 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.
- [298] A. T. Brünger. *X-PLOR: A System for Crystallography and NMR, Version 3.1*. Yale University, New Haven, CT, 1992.
- [299] J. R. Wyatt; J. D. Puglisi; I. Tinoco Jr. RNA Pseudoknots. Stability and Loop Size Requirements. *J. Mol. Biol.*, **1990**, 214, 455–470.
- [300] J. D. Puglisi; J. R. Wyatt; I. Tinoco Jr. Conformation of an RNA Pseudoknot. *J. Mol. Biol.*, **1990**, 214, 437–453.
- [301] 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.
- [302] B. Lewin. in *Genes IV*, pp 409–425. Cell Press, Cambridge, Mass., 1990.
- [303] W. H. Press; S. A. Teukolsky; W. T. Vetterling; B. P. Flannery. in *Numerical Recipes in C*, pp 113–117. Cambridge, New York, 1992.
- [304] V. B. Zhurkin; G. Raghunathan; N. B. Ulyanov; 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.
- [305] W. Saenger. in *Principles of Nucleic Acid Structure*, p 120. Springer-Verlag, New York, 1984.
- [306] M. Turcotte; G. Lapalme; F. Major. Exploring the conformations of nucleic acids. *J. Funct. Program.*, **1995**, 5, 443–460.
- [307] 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.
- [308] 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.
- [309] R. Lavery; K. Zakrzewska; H. Skelnar. JUMNA (junction minimisation of nucleic acids). *Comp. Phys. Commun.*, **1995**, 91, 135–158.
- [310] G. M. Crippen; T. F. Havel. *Distance Geometry and Molecular Conformation*. Research Studies Press, Taunton, England, 1988.
- [311] 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.
- [312] 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.
- [313] 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.
- [314] B. C. M. Potts; J. Smith; M. Akke; T. J. Macke; K. Okazaki; H. Hidaka; D. A. Case; W. J. Chazin. The structure of calyculin reveals a novel homodimeric fold S100 Ca<sup>2+</sup>-binding proteins. *Nature Struct. Biol.*, **1995**, 2, 790–796.

## BIBLIOGRAPHY

- [315] 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.
- [316] R. J. Gurbriel; 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.
- [317] T. J. Macke. *NAB, a Language for Molecular Manipulation*. Ph.D. thesis, The Scripps Research Institute, 1996.
- [318] 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.
- [319] R. Tan; S. Harvey. Molecular Mechanics Model of Supercoiled DNA. *J. Mol. Biol.*, **1989**, 205, 573–591.
- [320] 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.
- [321] 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.
- [322] 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.
- [323] T. F. Havel; I. D. Kuntz; G. M. Crippen. The theory and practice of distance geometry. *Bull. Math. Biol.*, **1983**, 45, 665–720.
- [324] D. K. Agrafiotis. Stochastic Proximity Embedding. *J. Comput. Chem.*, **2003**, 24, 1215–1221.
- [325] 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.
- [326] C. Brooks; A. Brünger; M. Karplus. Active site dynamics in protein molecules: A stochastic boundary molecular-dynamics approach. *Biopolymers*, **1985**, 24, 843–865.
- [327] D. T. Nguyen; D. A. Case. On finding stationary states on large-molecule potential energy surfaces. *J. Phys. Chem.*, **1985**, 89, 4020–4026.
- [328] 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.
- [329] S. Harvey; J. A. McCammon. *Dynamics of Proteins and Nucleic Acids*. Cambridge University Press, Cambridge, 1987.
- [330] M. P. Allen; D. J. Tildesley. *Computer Simulation of Liquids*. Clarendon Press, Oxford, 1987.
- [331] W. F. van Gunsteren; P. K. Weiner; A. J. Wilkinson, eds. *Computer Simulations of Biomolecular Systems, Vol. 3*. ESCOM Science Publishers, Leiden, 1997.
- [332] W. F. van Gunsteren; P. K. Weiner; A. J. Wilkinson, eds. *Computer Simulations of Biomolecular Systems, Vol. 2*. ESCOM Science Publishers, Leiden, 1993.
- [333] L. R. Pratt; G. Hummer, eds. *Simulation and Theory of Electrostatic Interactions in Solution*. American Institute of Physics, Melville, NY, 1999.

- [334] 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.
- [335] 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.
- [336] 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.
- [337] M. D. Beachy; R. A. Friesner. Accurate ab initio quantum chemical determination of the relative energies of peptide conformations and assessment of empirical force fields. *J. Am. Chem. Soc.*, **1997**, *119*, 5908–5920.
- [338] 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.
- [339] J. Higo; N. Ito; M. Kuroda; S. Ono; N. Nakajima; H. Nakamura. Energy landscape of a peptide consisting of  $\alpha$ -helix,  $3_{10}$  helix,  $\beta$ -turn,  $\beta$ -hairpin and other disordered conformations. *Prot. Sci.*, **2001**, *10*, 1160–1171.
- [340] 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.
- [341] 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.
- [342] 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.
- [343] 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.
- [344] J. Åqvist. Ion-water interaction potentials derived from free energy perturbation simulations. *J. Phys. Chem.*, **1990**, *94*, 8021–8024.
- [345] J. Åqvist; A. Warshel. Computer simulation of the initial proton-transfer step in human carbonic anhydrase-I. *J. Mol. Biol.*, **1992**, *224*, 7–14.
- [346] 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.
- [347] W. L. Jorgensen; J. D. Madura. Temperature and size dependence for Monte Carlo simulations of TIP4P water. *Mol. Phys.*, **1985**, *56*, 1381–1392.
- [348] 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.
- [349] H. J. C. Berendsen; J. R. Grigera; T. P. Straatsma. The missing term in effective pair potentials. *J. Phys. Chem.*, **1987**, *91*, 6269–6271.
- [350] 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.
- [351] D. S. Wishart; D. A. Case. Use of chemical shifts in macromolecular structure determination. *Meth. Enzymol.*, **2001**, *338*, 3–34.
- [352] 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.

## BIBLIOGRAPHY

- [353] B. Honig; A. Nicholls. Classical electrostatics in biology and chemistry. *Science*, **1995**, 268, 1144–1149.
- [354] 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.
- [355] 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.
- [356] 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.
- [357] M. L. Connolly. Analytical molecular surface calculation. *J. Appl. Cryst.*, **1983**, 16, 548–558.
- [358] 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.
- [359] 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.
- [360] 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.
- [361] 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.
- [362] 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.
- [363] 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.
- [364] A. Miranker; M. Karplus. Functionality maps of binding sites: A multiple copy simultaneous search method. *Proteins: Str. Funct. Gen.*, **1991**, 11, 29–34.
- [365] J. E. Straub; M. Karplus. Energy partitioning in the classical time-dependent Hartree approximation. *J. Chem. Phys.*, **1991**, 94, 6737.
- [366] 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.
- [367] 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.
- [368] A. Vedani; D. W. Huhta. A new force field for modeling metalloproteins. *J. Am. Chem. Soc.*, **1990**, 112, 4759–4767.
- [369] 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.
- [370] 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.

- [371] 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.
- [372] A. J. Hopfinger; R. A. Pearlstein. Molecular mechanics force-field parameterization procedures. *J. Comput. Chem.*, **1985**, 5, 486–499.
- [373] J. F. Cannon. AMBER force-field parameters for guanosine triphosphate and its imido and methylene analogs. *J. Comput. Chem.*, **1993**, 14, 995–1005.
- [374] 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.
- [375] 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.
- [376] 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.
- [377] 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.
- [378] 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.
- [379] T. A. Halgren. Merck Molecular Force Field (MMFF94). Part I-V. *J. Comput. Chem.*, **1996**, 17, 490–641.
- [380] 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.
- [381] 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.
- [382] 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.
- [383] 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.
- [384] 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.
- [385] 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.
- [386] J. Wang; P. A. Kollman. Automatic parameterization of force field by systematic search and genetic algorithms. *J. Comput. Chem.*, **2001**, 22, 1219–1228.
- [387] 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.

## BIBLIOGRAPHY

- [388] 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.
- [389] D. A. Pearlman; P. A. Kollman. The overlooked bond-stretching contribution in free energy perturbation calculations. *J. Chem. Phys.*, **1991**, *94*, 4532–4545.
- [390] D. A. Pearlman. Determining the contributions of constraints in free energy calculations: Development, characterization, and recommendations. *J. Chem. Phys.*, **1993**, *98*, 8946–8957.
- [391] 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.
- [392] D. A. Pearlman. A comparison of alternative approaches to free energy calculations. *J. Phys. Chem.*, **1994**, *98*, 1487–1493.
- [393] 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.
- [394] 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.
- [395] 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.
- [396] H.-A. Yu; M. Karplus. A thermodynamic analysis of solvation. *J. Chem. Phys.*, **1988**, *89*, 2366–2379.
- [397] G. Hummer. Fast-growth thermodynamic integration: Error and efficiency analysis. *J. Chem. Phys.*, **2001**, *114*, 7330–7337.
- [398] 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.
- [399] 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.
- [400] 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.
- [401] 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.
- [402] 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.
- [403] S. R. Niketic; K. Rasmussen. *The Consistent Force Field: A Documentation*. Springer-Verlag, New York, 1977.
- [404] C. Cerjan; W. H. Miller. On finding transition states. *J. Chem. Phys.*, **1981**, *75*, 2800.
- [405] D. T. Nguyen; D. A. Case. On finding stationary states on large-molecule potential energy surfaces. *J. Phys. Chem.*, **1985**, *89*, 4020–4026.
- [406] G. Lamm; A. Szabo. Langevin modes of macromolecules. *J. Chem. Phys.*, **1986**, *85*, 7334–7348.
- [407] J. Kottalam; D. A. Case. Langevin modes of macromolecules: application to crambin and DNA hexamers. *Biopolymers*, **1990**, *29*, 1409–1421.



- [408] 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.
- [409] T. Darden; D. Pearlman; L. G. Pedersen. Ionic charging free energies: Spherical versus periodic boundary conditions. *J. Chem. Phys.*, **1998**, 109, 10921–10935.
- [410] R. M. Levy; M. Karplus; J. Kushick; D. Perahia. Evaluation of the configurational entropy for proteins: Application to molecular dynamics simulations of an  $\alpha$ -helix. *Macromolecules*, **1984**, 17, 1370–1374.
- [411] 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.
- [412] 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.
- [413] O. Becker; A. D. MacKerell; B. Roux; M. Watanabe, eds. *Computational Biochemistry and Biophysics*. Marcel Dekker, New York, 2001.
- [414] A. R. Leach. *Molecular Modelling. Principles and Applications, Second Edition*. Prentice-Hall, Harlow, England, 2001.
- [415] 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.
- [416] G. Sigalov; P. Scheffel; A. Onufriev. Incorporating variable environments into the generalized Born model. *J. Chem. Phys.*, **2005**, 122, 094511.
- [417] G. Sigalov; A. Fenley; A. Onufriev. Analytical electrostatics for biomolecules: Beyond the generalized Born approximation. *J. Chem. Phys.*, **2006**, 124, 124902.
- [418] A. Mitsutake; Y. Sugita; Y. Okamoto. Generalized-ensemble algorithms for molecular simulations of biopolymers. *Biopolymers*, **2001**, 60, 96–123.
- [419] J. J. Prompers; R. Brüschweiler. Dynamic and structural analysis of isotropically distributed molecular ensembles. *Proteins*, **2002**, 46, 177–189.
- [420] 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.
- [421] 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.
- [422] 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.
- [423] 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.
- [424] 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.
- [425] 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.
- [426] J. Kottalam; D. A. Case. Dynamics of ligand escape from the heme pocket of myoglobin. *J. Am. Chem. Soc.*, **1988**, 110, 7690–7697.

## BIBLIOGRAPHY

- [427] B. Roux. The calculation of the potential of mean force using computer simulations. *Comput. Phys. Comm.*, **1995**, *91*, 275–282.
- [428] J. W. Ponder; D. A. Case. Force fields for protein simulations. *Adv. Prot. Chem.*, **2003**, *66*, 27–85.
- [429] 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.
- [430] P. Beroza; D. A. Case. Calculations of proton-binding thermodynamics in proteins. *Meth. Enzymol.*, **1998**, *295*, 170–189.
- [431] 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.
- [432] M. K. Gilson. Theory of electrostatic interactions in macromolecules. *Curr. Opin. Struct. Biol.*, **1995**, *5*, 216–23.
- [433] M. Scarsi; J. Apostolakis; A. Caffisch. Continuum electrostatic energies of macromolecules in aqueous solutions. *J. Phys. Chem. A*, **1997**, *101*, 8098–8106.
- [434] R. Luo; L. David; M. K. Gilson. Accelerated Poisson-Boltzmann calculations for static and dynamic systems. *J. Comput. Chem.*, **2002**, *23*, 1244–1253.
- [435] 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.
- [436] 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.
- [437] H. Wei; R. Luo; R. Qi. An Efficient Second-Order Poisson-Boltzmann Method. *J. Comput. Chem.*, **2019**, *40*, 1257.
- [438] 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.
- [439] 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.
- [440] C. Wang; P. Ren; R. Luo. Ionic solution: What goes right and wrong with continuum solvation modeling. *J. Phys. Chem. B*, **2017**, *121*, 11169.
- [441] 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.
- [442] Z. Li; I. K. *The Immersed Interface Method: Numerical Solutions of PDEs Involving Interfaces and Irregular Domains*. SIAM Frontiers in Applied Mathematics, Philadelphia, 2006.
- [443] 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.
- [444] J. Wang; R. Luo. Assessment of Linear Finite-Difference Poisson-Boltzmann solvers. *J. Comput. Chem.*, **2010**, *31*, 1689–1698.
- [445] Q. Cai; J. Wang; H. Zhao; R. Luo. On removal of charge singularity in Poisson-Boltzmann equation. *J. Chem. Phys.*, **2009**, *130*, 145101.
- [446] Q. Cai; M.-J. Hsieh; J. Wang; R. Luo. Performance of Nonlinear Finite-Difference Poisson-Boltzmann Solvers. *J. Chem. Theory Comput.*, **2010**, *6*, 203.

- [447] 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.
- [448] 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.
- [449] Q. Cai; X. Ye; R. Luo. Dielectric Pressure in Continuum Electrostatic Solvation of Biomolecules. *Phys. Chem. Chem. Phys.*, **2012**, 14, 15917–15925.
- [450] 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.
- [451] 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.
- [452] 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.
- [453] 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.
- [454] R. Qi; R. Luo. Robustness and Efficiency of Poisson-Boltzmann Modeling on Graphics Processing Units. *J. Chem. Inf. Model.*, **2019**, 59, 409–420.
- [455] T. Simonson. Electrostatics and dynamics of proteins. *Rep. Prog. Phys.*, **2003**, 66, 737–787.
- [456] D. Bashford; M. Karplus. pK sub a's of ionizable groups in proteins: Atomic detail from a continuum electrostatic model. *Biochemistry*, **1990**, 29, 10219–10225.
- [457] 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.
- [458] 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.
- [459] J. D. Jackson. *Classical Electrodynamics*. Wiley and Sons, New York, 1975.
- [460] Q. Lu; R. Luo. A Poisson-Boltzmann dynamics method with nonperiodic boundary condition. *J. Chem. Phys.*, **2003**, 119, 11035–11047.
- [461] 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.
- [462] C. H. Tan; Y. H. Tan; R. Luo. Implicit nonpolar solvent models. *J. Phys. Chem. B*, **2007**, 111, 12263–12274.
- [463] 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.
- [464] 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.
- [465] A. E. García; K. Y. Sanbonmatsu.  $\alpha$ -helical stabilization by side chain shielding of backbone hydrogen bonds. *Proc. Natl. Acad. Sci. USA*, **2002**, 99, 2782–2787.
- [466] 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.

## BIBLIOGRAPHY

- [467] K. N. Kirschner; R. J. Woods. Solvent interactions determine carbohydrate conformation. *Proc. Natl. Acad. Sci. USA*, **2001**, 98, 10541–10545.
- [468] 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.
- [469] 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.
- [470] K. A. Sharp; B. Honig. Electrostatic interactions in macromolecules: Theory and experiment. *Annu. Rev. Biophys. Biophys. Chem.*, **1990**, 19, 301–332.
- [471] 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.
- [472] J. Warwicker; H. C. Watson. Calculation of the electric potential in the active site cleft due to. *J. Mol. Biol.*, **1982**, 157, 671–679.
- [473] 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.
- [474] 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.
- [475] 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.
- [476] M. E. Davis; J. A. McCammon. Electrostatics in biomolecular structure and dynamics. *Chem. Rev.*, **1990**, 90, 509–521.
- [477] 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.
- [478] D. Bashford. An object-oriented programming suite for electrostatic effects in biological molecules. *Lect. Notes Comput. Sci.*, **1997**, 1343, 233–240.
- [479] 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.
- [480] 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.
- [481] 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.
- [482] 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.
- [483] 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.
- [484] 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.
- [485] 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.
- [486] 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.

- [487] 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.
- [488] S. L. Dixon; K. M. Merz, Jr. Semiempirical molecular orbital calculations with linear system size scaling. *J. Chem. Phys.*, **1996**, *104*, 6643–6649.
- [489] S. L. Dixon; K. M. Merz, Jr. Fast, accurate semiempirical molecular orbital calculations for macro-molecules. *J. Chem. Phys.*, **1997**, *107*, 879–893.
- [490] J. Nocedal; S. J. Wright. *Numerical Optimization*. Springer-Verlag, New York, 1999.
- [491] S. G. Nash. A survey of truncated-Newton methods. *J. of Computational and Applied Mathematics*, **2000**, *124*, 45–59.
- [492] X. Cheng; V. Hornak; C. Simmerling. Improved conformational sampling through an efficient combination of mean-field simulation approaches. *J. Phys. Chem. B*, **2004**, *108*.
- [493] 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.
- [494] 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.
- [495] R. Elber; M. Karplus M. A method for determining reaction paths in large molecules: Application to myoglobin. *Chem. Phys. Lett.*, **1987**, *139*, 375–380.
- [496] 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.
- [497] 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.
- [498] H. Jönsson; G. Mills; K. W. Jacobsen. in *Classical and Quantum Dynamics in Condensed Phase Simulations*, B. J. Berne; G. Cicciotti; D. F. Coker, Eds., pp 385–404. World Scientific, Singapore, 1998.
- [499] G. Mills; H. Jönsson. Quantum and thermal effects in H<sub>2</sub> dissociative adsorption: Evaluation of free energy barriers in multidimensional quantum systems. *Phys. Rev. Lett.*, **1994**, *72*, 1124–1127.
- [500] J. Mongan; D. A. Case; J. A. McCammon. Constant pH molecular dynamics in generalized Born implicit solvent. *J. Comput. Chem.*, **2004**, *25*, 2038–2048.
- [501] 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.
- [502] E. J. Sorin; V. S. Pande. Exploring the helix-coil transition via all-atom equilibrium ensemble simulations. *Biophys. J.*, **2005**, *88*, 2472–2493.
- [503] P. Ren; J. W. Ponder. Temperature and pressure dependence of the AMOEBA water model. *J. Phys. Chem. B*, **2004**, *108*, 13427–13437.
- [504] P. Ren; J. W. Ponder. Consistent treatment of inter- and intramolecular polarization in molecular mechanics calculations. *J. Comput. Chem.*, **2002**, *23*, 1497–1506.
- [505] 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.

## BIBLIOGRAPHY

- [506] 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.
- [507] 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.
- [508] 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.
- [509] 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.
- [510] R. P. Feynman; A. R. Hibbs. *Quantum Mechanics and Path Integrals*. McGraw-Hill, New York, 1965.
- [511] R. P. Feynman. *Statistical Mechanics*. Benjamin, Reading, MA, 1972.
- [512] H. Kleinert. *Path Integrals in Quantum Mechanics, Statistics, and Polymer Physics*. World Scientific, Singapore, 1995.
- [513] L. S. Schulman. *Techniques and Applications of Path Integration*. Wiley & Sons, New York, 1996.
- [514] A. Messiah. *Quantum Mechanics*. Wiley & Sons, New York, 1958.
- [515] 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.
- [516] D. M. Ceperley. Path integrals in the theory of condensed helium. *Rev. Mod. Phys.*, **1995**, 67, 279–355.
- [517] 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.
- [518] 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.
- [519] 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.
- [520] A. van der Vaart; K. M. Merz, Jr. Divide and conquer interaction energy decomposition. *J. Phys. Chem. A*, **1999**, 103, 3321–3329.
- [521] A. V. Mitin. The dynamic level shift method for improving the convergence of the SCF procedure. *J. Comput. Chem.*, **1988**, 9, 107–110.
- [522] 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.
- [523] 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.
- [524] B. Wang; K. Raha; K. M. Merz Jr. Pose scoring by NMR. *J. Am. Chem. Soc.*, **2004**, 126, 11430–11431.
- [525] 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.

- [526] 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.
- [527] 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  $\text{CH}_3\text{Cl} + \text{Cl}^-$  exchange-reaction and gas-phase protonation of polyethers. *J. Comp. Chem.*, **1986**, *7*, 718–730.
- [528] 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.
- [529] 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.
- [530] 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.
- [531] 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.
- [532] 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.
- [533] 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.
- [534] 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.
- [535] 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.
- [536] 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.
- [537] H. Nymeyer; S. Gnanakaran; A. García. Atomic simulations of protein folding using the replica exchange algorithm. *Meth. Enzymol.*, **2004**, *383*, 119–149.
- [538] 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.
- [539] 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.
- [540] 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.
- [541] 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.
- [542] 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.
- [543] C. Jarzynski. Nonequilibrium equality for free energy differences. *Phys. Rev. Lett.*, **1997**, *78*, 2690–2693.
- [544] G. Hummer; A. Szabo. Free energy reconstruction from nonequilibrium single-molecule pulling experiments. *Proc. Natl. Acad. Sci. USA*, **2001**, *98*, 3658.

## BIBLIOGRAPHY

- [545] G. Hummer; A. Szabo. Kinetics from nonequilibrium single-molecule pulling experiments. *Biophys. J.*, **2003**, 85, 5–15.
- [546] 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.
- [547] 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.
- [548] P. Y. Ren; J. W. Ponder. Polarizable atomic multipole water model for molecular mechanics simulation. *J. Phys. Chem. B*, **2003**, 107, 5933–5947.
- [549] P. Y. Ren; J. W. Ponder. Tinker polarizable atomic multipole force field for proteins. *to be published.*, **2006**.
- [550] 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.
- [551] 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.
- [552] A. Warshel. *Computer Modeling of Chemical Reactions in Enzymes and Solutions*. John Wiley and Sons, New York, 1991.
- [553] 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.
- [554] 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.
- [555] W. Kabsch; C. Sander. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, **1983**, 22, 2577–2637.
- [556] T. E. Cheatham, III; M. A. Young. Molecular dynamics simulation of nucleic acids: Successes, limitations and promise. *Biopolymers*, **2001**, 56, 232–256.
- [557] 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.
- [558] 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.
- [559] 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.
- [560] 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.
- [561] 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.
- [562] 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.
- [563] 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.



- [564] 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.
- [565] 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.
- [566] 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.
- [567] Y. Saad; M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, **1986**, *7*, 856.
- [568] P. Pulay. Convergence acceleration of iterative sequences. The case of SCF iteration. *Chem. Phys. Lett.*, **1980**, *73*, 393.
- [569] P. Pulay. Improved SCF convergence acceleration. *J. Comput. Chem.*, **1982**, *3*, 556.
- [570] 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.
- [571] 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.
- [572] G. J. Martyna; M. L. Klein; M. Tuckerman. Nosé-Hoover chains: The canonical ensemble via continuous dynamics. *J. Chem. Phys.*, **1992**, *97*, 2635.
- [573] G. J. Martyna; A. Hughes; M. E. Tuckerman. Molecular dynamics algorithms for path integrals at constant pressure. *J. Chem. Phys.*, **1999**, *110*, 3275.
- [574] B. J. Berne; D. Thirumalai. On the simulation of quantum systems: path integral methods. *Annu. Rev. Phys. Chem.*, **1986**, *37*, 401.
- [575] G. A. Voth. Path-integral centroid methods in quantum statistical mechanics and dynamics. *Adv. Chem. Phys.*, **1996**, *93*, 135.
- [576] I. R. Craig; D. E. Manolopoulos. Quantum statistics and classical mechanics: Real time correlation functions from ring polymer molecular dynamics. *J. Chem. Phys.*, **2004**, *121*, 3368.
- [577] T. F. Miller; D. E. Manolopoulos. Quantum diffusion in liquid water from ring polymer molecular dynamics. *J. Chem. Phys.*, **2005**, *123*, 154504.
- [578] 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.
- [579] J. Vaníček; W. H. Miller; J. F. Castillo; F. J. Aoiz. Quantum-instanton evaluation of the kinetic isotope effects. *J. Chem. Phys.*, **2005**, *123*, 054108.
- [580] 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.
- [581] 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.
- [582] W. H. Miller. Semiclassical limit of quantum mechanical transition state theory for nonseparable systems. *J. Chem. Phys.*, **1975**, *62*, 1899.

## BIBLIOGRAPHY

- [583] 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.
- [584] 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.
- [585] W. H. Miller; S. D. Schwartz; J. W. Tromp. Quantum mechanical rate constants for bimolecular reactions. *J. Chem. Phys.*, **1983**, *79*, 4889–4898.
- [586] 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.
- [587] 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.
- [588] 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.
- [589] H. Kopitz; A. Zivkovic; J. W. Engels; H. Gohlke. Determinants of the unexpected stability of RNA fluorobenzene self pairs. *ChemBioChem*, **2008**, *9*, 2619–2622.
- [590] S. Fulle; H. Gohlke. Analyzing the flexibility of RNA structures by constraint counting. *Biophys. J.*, **2008**, DOI:10.1529/biophysj.107.113415.
- [591] 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.
- [592] A. Ahmed; H. Gohlke. Multiscale modeling of macromolecular conformational changes combining concepts from rigidity and elastic network theory. *Proteins*, **2006**, *63*, 1038–1051.
- [593] 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.
- [594] D. J. Price; C. L. Brooks. A modified TIP3P water potential for simulation with Ewald summation. *J. Chem. Phys.*, **2004**, *121*, 10096–10103.
- [595] 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.
- [596] 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.
- [597] R. J. Woods. Derivation of net atomic charges from molecular electrostatic potentials. *J. Comput. Chem.*, **1990**, *11*, 29–310.
- [598] M. L. DeMarco; R. J. Woods. Bridging computational biology and glycobiology: A game of snakes and ladders. *Glycobiology*, **2008**, *18*, 426–440.
- [599] R. J. Woods. Restrained electrostatic potential charges for condensed phase simulations of carbohydrates. *J. Mol. Struct (Theochem)*, **2000**, *527*, 149–156.
- [600] 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.

- [601] 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.
- [602] 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.
- [603] 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.
- [604] 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.
- [605] 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.
- [606] 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.
- [607] 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.
- [608] 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.
- [609] E. Darve; A. Pohorille. Calculating free energies using average force. *J. Chem. Phys.*, **2001**, 115, 9169–9183.
- [610] 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.
- [611] 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.
- [612] S. Joing; 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.
- [613] 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.
- [614] 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.
- [615] 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.
- [616] P. Cieplak; F.-Y. Dupradeau; Y. Duan; J. Wang. Polarization effects in molecular mechanical force fields. *J. Phys.: Condens. Matter*, **2009**, 21, 333102.
- [617] Z. J. Shi; J. Shen. New inexact line search method for unconstrained optimization. *J. Optim. Theory Appl.*, **2005**, 127, 425–446.

## BIBLIOGRAPHY

- [618] 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. Wiorcikiewicz-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.
- [619] 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.
- [620] 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.
- [621] 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. Hodoseck; 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.
- [622] 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.
- [623] 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.
- [624] 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.
- [625] B. J. Berne; G. D. Harp. *Adv. Chem. Phys.*, **1970**, *17*, 63.
- [626] R. Kubo; M. Toda; N. Hashitsume. *Statistical Physics II: Nonequilibrium Statistical Mechanics*, 2nd ed. Springer-Verlag, Heidelberg, 1991.
- [627] W. H. Miller. *Adv. Chem. Phys.*, **1974**, *25*, 69.
- [628] W. H. Miller. Including quantum effects in the dynamics of complex (i.e., large) molecular systems. *J. Chem. Phys.*, **2006**, *125*, 132305.
- [629] 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.
- [630] 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.
- [631] 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.
- [632] 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.
- [633] 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.
- [634] 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.

- [635] 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.
- [636] 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.
- [637] 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.
- [638] Q. Shi; E. Giva. *J. Chem. Phys. A*, **2003**, *107*, 9059.
- [639] 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.
- [640] 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.
- [641] 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.
- [642] 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.
- [643] V. Sergiievskiy; 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.
- [644] F. Hirata, Ed. *Molecular Theory of Solvation*. Kluwer Academic Publishers, 2003.
- [645] 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.
- [646] 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.
- [647] M. Tuckerman; B. J. Berne; G. J. Martyna. Reversible multiple time scale molecular dynamics. *J. Chem. Phys.*, **1992**, *97*, 1990–2001.
- [648] H. Grubmuller; H. Heller; A. Windemuth; K. Schulten. Generalized Verlet algorithm for efficient molecular dynamics simulations with long-range interactions. *Mol. Simulat.*, **1991**, *6*, 121–142.
- [649] J. S. Perkyns; B. M. Pettitt. A site-site theory for finite concentration saline solutions. *J. Chem. Phys.*, **1992**, *97*, 7656–7666.
- [650] 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.
- [651] 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*.
- [652] T. Schlick. *Molecular modeling and simulation: an interdisciplinary guide*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.
- [653] 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.

## BIBLIOGRAPHY

- [654] D. W. Li; R. Brüschweiler. NMR-based protein potentials. *Angew. Chem. Int. Ed.*, **2010**, 49, 6778–6780.
- [655] 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.
- [656] 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.
- [657] 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.
- [658] R. Anandakrishnan; M. Daga; A. V. Onufriev. An n log n Generalized Born Approximation. *J. Chem. Theory Comput.*, **2011**, 7, 544–559.
- [659] 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.
- [660] 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 Comput.*, **2009**, 5, 2322–2338.
- [661] 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.
- [662] 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.
- [663] 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.
- [664] M.-J. Hsieh; R. Luo. Exploring a coarse-grained distributive strategy for finite-difference poisson-boltzmann calculations. *J. Molec. Model.*, **2011**.
- [665] 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.
- [666] 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.
- [667] 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.
- [668] 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.
- [669] 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.

- [670] M. Zgarbova; M. Otyepka; J. Spöner; 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.
- [671] C. Perez; F. Lohr; H. Rüterjans; 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.
- [672] 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.
- [673] H.-A. Yu; B. Roux; M. Karplus. Solvation thermodynamics: An approach from analytic temperature derivatives. *J. Chem. Phys.*, **1990**, 92, 5020–5033.
- [674] T. Yamazaki; N. Blinov; D. Wishart; A. Kovalenko. Hydration effects on the HET-s prion and amyloid- $\beta$ 2 fibrillous aggregates, studied with three-dimensional molecular theory of solvation. *Biophys. J.*, **2008**, 95, 4540–4548.
- [675] 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.
- [676] N. Homeyer; H. Gohlke. Free energy calculations by the molecular mechanics poisson-boltzmann surface area method. *Mol. Informatics*, **2012**, DOI: 10.1002/minf.201100135.
- [677] V. Wong; D. A. Case. Evaluating rotational diffusion from protein md simulations. *J. Phys. Chem. B*, **2008**, 112, 6013–6024.
- [678] 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.
- [679] 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.
- [680] 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.
- [681] 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.
- [682] 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.
- [683] 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.
- [684] 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.
- [685] 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.
- [686] S. Harvey; M. Prabhakaran. Ribose puckering - structure, dynamics, energetics, and the pseudorotation cycle. *J Am Chem Soc*, **1986**, 108, 6128–6136.
- [687] D. Cremer; J. Pople. A general definition of ring puckering coordinates. *J Am Chem Soc*, **1975**, 97, 1354–1358.

## BIBLIOGRAPHY

- [688] T. Gaillard; D. A. Case. Evaluation of DNA Force Fields in Implicit Solvation. *J. Chem. Theory Comput.*, **2011**, 7, 3181–3198.
- [689] 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.
- [690] 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.
- [691] 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.
- [692] 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**.
- [693] 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.
- [694] A. Pohorille; C. Jarzynski; C. Chipot. Good practices in Free-Energy calculations. *J. Phys. Chem. B*, **2010**, 114, 10235–10253.
- [695] M. R. Shirts; J. D. Chodera. Statistically optimal analysis of samples from multiple equilibrium states. *J. Chem. Phys.*, **2008**, 129, 124105–124105–10.
- [696] Å. Skjerve; 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.
- [697] 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.
- [698] 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.
- [699] 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.
- [700] 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.
- [701] 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.
- [702] 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.
- [703] 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.
- [704] 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.



- [705] B. T. Thole. Molecular polarizabilities calculated with a modified dipole interaction. *Chem. Phys.*, **1981**, 59, 341–350.
- [706] R. Bosque; J. Sales. Polarizabilities of solvents from the chemical composition. *J. Chem. Inf. Comput. Sci.*, **2002**, 42, 1154–1163.
- [707] 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.
- [708] 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.
- [709] T. E. Cheatham, III. Simulation and modeling of nucleic acid structure, dynamics and interactions. *Curr. Opin. Struct. Biol.*, **2004**, 14, 360–367.
- [710] 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.
- [711] M. Krepl; M. Zgarbova; P. Stadlbauer; M. Otyepka; P. Banas; J. Koca; T. E. Cheatham, III; J. Spomer. 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.
- [712] J. Wang; T. Hou. Application of Molecular Dynamics Simulations in Molecular Property Prediction. II. Diffusion coefficient. *J. Comput. Chem.*, **2011**, 32, 3509–3519.
- [713] C. F. Fu; S. X. Tian. A Comparative Study for Molecular Dynamics Simulations of Liquid Benzene. *J. Chem. Theory Comput.*, **2011**, 7, 2240–2252.
- [714] 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.
- [715] 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.
- [716] 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.
- [717] 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.
- [718] 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.
- [719] 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 Neu5Ac $\alpha$ 2-6Gal $\beta$ 1-4GlcNAc human-type influenza receptor. *Glycobiology*, **2011**, 21, 973–984.
- [720] B. L. Foley; M. B. Tessier; R. J. Woods. Carbohydrate force fields. *WIREs Comput. Mol. Sci.*, **2012**, 2, 652–697.
- [721] 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  $\alpha$ -N-Acetylglucosaminidase from *Clostridium perfringens*. *PLoS ONE*, **2012**, 7, e33524.

## BIBLIOGRAPHY

- [722] 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.
- [723] 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.
- [724] P. Varnai; D. Djuranovic; R. Lavery; B. Hartmann.  $\alpha/\gamma$  Transitions in the B-DNA backbone. *Nucl. Acids Res.*, **2002**, *30*, 5398–5406.
- [725] 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.
- [726] 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.
- [727] T. Steinbrecher; J. Latzer; D. A. Case. Revised AMBER Parameters for Bioorganic Phosphates. *J. Chem. Theory Comput.*, **2012**, *8*, 4405–4412.
- [728] 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.
- [729] S. E. Feller. Molecular dynamics simulations of lipid bilayers. *Curr. Opin. Colloid Interface Sci.*, **2000**, *5*, 217–223.
- [730] 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.
- [731] 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.
- [732] R. M. Betz; N. A. DeBardleben; 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.
- [733] D. E. Warschawski; P. F. Devaux. Order parameters of unsaturated phospholipids in membranes and the effect of cholesterol: a  $^1\text{H}$ - $^{13}\text{C}$  solid-state NMR study at natural abundance. *Eur. Biophys. J.*, **2005**, *34*, 987–996.
- [734] L. Saiz; M. L. Klein. Computer Simulation Studies of Model Biological Membranes. *Acc. Chem. Res.*, **2002**, *35*, 482–489.
- [735] 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.
- [736] J. T. Berryman; T. Schilling. Absolute Free Energies for Biomolecules in Implicit or Explicit Solvent. *Physics Procedia*, **2014**, *57*, 7–15.
- [737] T. Schilling; F. Schmid. Computing absolute free energies of disordered structures by molecular simulation. *J. Chem. Phys.*, **2009**, *131*, 231102.
- [738] F. Schmid; T. Schilling. A method to compute absolute free energies or enthalpies of fluids. *Physics Procedia*, **2010**, *4*, 131–143.

- [739] 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.
- [740] 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.
- [741] 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.
- [742] S. G. Itoh; A. Damjanovic; B. R. Brooks. pH replica-exchange method based on discrete protonation states. *Proteins*, **2011**, *79*, 3420–3436.
- [743] 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.
- [744] R. B. Best; N.-V. Buchete; G. Hummer. Are Current Molecular Dynamics Force Fields too Helical? *Biophys. J.*, **2008**, *95*, L07–L09; 4494.
- [745] 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.
- [746] 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.
- [747] K. K. Patapati; N. M. Glykos. Three force fields views of the 3-10 helix. *Biophys. J.*, **2011**, *101*, 1766–1771.
- [748] 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.
- [749] 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.
- [750] PDB Current Holdings Breakdown. **2013**.
- [751] M. L. Lundstrom, K. H.; Chiu. *G Protein-Coupled Receptors in Drug Discovery*. Taylor & Francis, London, 2005.
- [752] C. Chipot; A. Pohorille, eds. *Free energy calculations. Theory and Applications in Chemistry and Biology*. Springer, Berlin, 2007.
- [753] M. Griebel; S. Knapek; G. Zumbusch. *Numerical Simulation in Molecular Dynamics. Numerical Algorithms, Parallelization, Applications*. Springer-Verlag, Berlin, 2010.
- [754] M. E. Tuckerman. *Statistical Mechanics: Theory and Molecular Simulation*. Oxford University Press, Oxford, 2010.
- [755] 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.
- [756] 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.
- [757] 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.

## BIBLIOGRAPHY

- [758] 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.
- [759] 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.
- [760] M. Fajer; D. Hamelberg; J. A. McCammon. Replica-Exchange Accelerated Molecular Dynamics (REX-AMD) Applied to Thermodynamic Integration. *J. Chem. Theory Comput.*, **2008**, *4*, 1565–1569.
- [761] 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.
- [762] 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.
- [763] P. Li; L. F. Song; K. M. Merz, Jr. Systematic Parameterization of Monovalent Ions Employing the Non-bonded Model. *J. Chem. Theory Comput.*, **2015**, *11*, 1645–1657.
- [764] M. T. Panteva; G. M. Giambasu; D. M. York. Comparison of Structural, Thermodynamic, Kinetic and Mass Transport Properties of Mg<sup>2+</sup> Models Commonly Used in Biomolecular Simulations. *J. Comput. Chem.*, **2015**, *36*, 970–982.
- [765] M. T. Panteva; G. M. Giambasu; D. M. York. Force Field for Mg<sup>2+</sup>, Mn<sup>2+</sup>, Zn<sup>2+</sup> and Cd<sup>2+</sup> Ions That Have Balanced Interactions with Nucleic Acids. *J. Phys. Chem. B*, **2015**, *119*, 15460–15470.
- [766] 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.
- [767] P. Li; K. M. Merz, Jr. MCPB.py: A Python Based Metal Center Parameter Builder. *J. Chem. Inf. Model.*, **2016**, *56*, 599–604.
- [768] P. Li; K. M. Merz, Jr. Metal Ion Modeling Using Classical Mechanics. *Chem. Rev.*, **2017**, *117*, 1564–1686.
- [769] Z. Yu; P. Li; K. M. Merz, Jr. Extended Zinc AMBER Force Field (EZAFF). *J. Chem. Theory Comput.*, **2018**, *14*, 242–254.
- [770] A. Sengupta; A. Seitz; K. M. Merz, Jr. Simulating the Chelate Effect. *J. Am. Chem. Soc.*, **2018**, *140*, 15166–15169.
- [771] 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.
- [772] 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.
- [773] P. Li; K. M. Merz, Jr. in *Methods Mol. Biol.*, (Humana Press, New York, NY). volume 2199. pp 257–275. 2021.
- [774] 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.
- [775] 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.
- [776] J. M. Seminario. Calculation of Intramolecular Force Fields from Second-Derivative Tensors. *Int. J. Quantum Chem.*, **1996**, *30*, 1271–1277.

- [777] P. Eastman; V. S. Pande. OpenMM: A Hardware-Independent Framework for Molecular Simulations. *Computing in Science and Engineering*, **2010**, 12, 34–39.
- [778] 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.
- [779] J. M. Swails. *Free Energy Simulations of Complex Biological Systems at Constant pH*. PhD thesis, University of Florida, 2013.
- [780] 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.
- [781] 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.
- [782] A. Bakan; L. M. Meireles; I. Bahar. ProDy: Protein Dynamics Inferred from Theory and Experiments. *Bioinformatics*, **2011**, 27, 1575–1577.
- [783] 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.
- [784] 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.
- [785] 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.
- [786] 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.
- [787] N. Bernstein; J. R. Kermode; G. Csányi. Hybrid atomistic simulation methods for materials systems. *Rep. Prog. Phys.*, **2009**, 72, 026501.
- [788] A. Jones; B. Leimkuhler. Adaptive stochastic methods for sampling driven molecular systems. *J. Chem. Phys.*, **2011**, 135, 084125.
- [789] T. Kerdcharoen; B. M. Rode. A QM/MM simulation method applied to the solution of Li<sup>+</sup> in liquid ammonia. *Chem. Phys.*, **1996**, 211, 313–323.
- [790] N. Homeyer; H. Gohlke. FEW - A workflow tool for free energy calculations of ligand binding. *J. Comput. Chem.*, **2013**, 34, 965–973.
- [791] A. Metz. *Goethe University (Frankfurt am Main)*, **2006**.
- [792] 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.
- [793] J. Åqvist; V. B. Luzhkov; B. O. Brandsdal. Ligand binding affinities from MD simulations. *Acc. Chem. Res.*, **2002**, 35, 358–365.
- [794] H. G. Wallnoefer; K. R. Liedl; T. Fox. A challenging system: free energy prediction for factor Xa. *J. Comput. Chem.*, **2011**, 32, 1743–1752.
- [795] 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.

## BIBLIOGRAPHY

- [796] 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.
- [797] 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.
- [798] 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.
- [799] 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.
- [800] 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.
- [801] J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proc. Amer. Math. Soc.*, **1956**, 7, 48–50.
- [802] *ROCS, OpenEye Scientific Software, Santa Fe*, <http://www.eyesopen.com>.
- [803] 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.
- [804] T. E. Cheatham; D. A. Case. Twenty-five years of nucleic acid simulations. *Biopolymers*, **2013**, 99, 969–977.
- [805] C. J. Cramer. *Essentials of Computational Chemistry: Theories and Models*. John Wiley & Sons, New York, 2002.
- [806] T. Lazaridis. Inhomogeneous Fluid Approach to Solvation Thermodynamics. 1 Theory. *J. Phys. Chem. B*, **1998**, 102, 3531–3541.
- [807] 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.
- [808] 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.
- [809] W. Humphrey; A. Dalke; K. Schulten. VMD Visual Molecular Dynamics. *J. Molec. Graph.*, **1996**, 14, 33–38.
- [810] 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.
- [811] 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.
- [812] 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.
- [813] 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.
- [814] 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.

- [815] R. Mills. Self-diffusion in normal and heavy water in the range 1-45. deg. *J. Phys. Chem.*, **1973**, 77, 685–688.
- [816] 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.
- [817] 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+.
- [818] G. S. Kell. Precise representation of volume properties of water at one atmosphere. *J. Chem. Eng. Data*, **1967**, 12, 66–69.
- [819] S. Niu; M. L. Tan; T. Ichiye. The large quadrupole of water molecules. *J. Chem. Phys.*, **2011**, 134, 134501+.
- [820] S. Izadi; R. Anandakrishnan; A. V. Onufriev. Building Water Models: A Different Approach. *J. Phys. Chem. Lett.*, **2014**, 5, 3863–3871.
- [821] 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.
- [822] 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.
- [823] 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.
- [824] 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.
- [825] B. Schneider; S. Neidle; H. M. Berman. Conformations of the sugar-phosphate backbone in helical dna crystal structures. *Biopolymers*, **1997**, 42, 113–124.
- [826] B. Schneider; Z. Moravek; H. M. Berman. Rna conformational classes. *Nucleic Acids Res.*, **2004**, 32, 1666–1677.
- [827] 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.
- [828] 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.
- [829] 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.
- [830] 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.
- [831] W. Harb; M. I. Bernal-Uruchurtu; M. F. Ruiz-López. An improved semiempirical method for hydrated systems. *Theor. Chem. Acc.*, **2004**, 112, 204–216.
- [832] 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.
- [833] 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.

## BIBLIOGRAPHY

- [834] 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.
- [835] 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.
- [836] 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.
- [837] 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.
- [838] 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.
- [839] C. Vega; J. L. F. Abascal. Simulating water with rigid non-polarizable models: a general perspective. *Phys Chem Chem Phys*, **2011**, 13, 19663–19688.
- [840] 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.
- [841] 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.
- [842] 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.
- [843] 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.
- [844] 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.
- [845] S. Jo; T. Kim; W. Im. Automated builder and database of protein/membrane complexes for molecular dynamics simulations. *PLoS One*, **2007**, 2, e880.
- [846] 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.
- [847] 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.
- [848] 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.
- [849] 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.
- [850] M. Holst; F. Saied. Multigrid solution of the Poisson-Boltzmann equation. *J. Comput. Chem.*, **1993**, 14, 105–113.
- [851] 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.
- [852] M. Holst. Adaptive numerical treatment of elliptic systems on manifolds. *Adv. Comput. Math.*, **2001**, 15, 139–191.



- [853] R. Bank; M. Holst. A New Paradigm for Parallel Adaptive Meshing Algorithms. *SIAM Review*, **2003**, *45*, 291–323.
- [854] 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.
- [855] H. Nymeyer; H. X. Zhou. A method to determine dielectric constants in nonhomogeneous systems: application to biological membranes. *Biophys. J.*, **2008**, *94*, 1185–1193.
- [856] 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.
- [857] 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.
- [858] 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*.
- [859] 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.
- [860] 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.
- [861] 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.
- [862] 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.
- [863] 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.
- [864] 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.
- [865] 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.
- [866] 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.
- [867] H. Nguyen; D. R. Roe; J. M. Swails; D. A. Case. PYTRAJ: Interactive data analysis for molecular dynamics simulations. *Manuscript in preparation*, **2016**.
- [868] 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**.
- [869] 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. Gelpi; 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.
- [870] 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.

## BIBLIOGRAPHY

- [871] 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.
- [872] 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.
- [873] 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.
- [874] 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.
- [875] 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.
- [876] C. Torrence; G. P. Compo. A practical guide to wavelet analysis. *Bull. Am. Meteorol. Soc.*, **1998**, *79*, 61–78.
- [877] N. C. Benson; V. Dagget. Wavelet analysis of protein motion. *Int. J. Wavelets Multi.*, **2012**, *10*.
- [878] A. Rodriguez; A. Laio. Clustering by fast search and find of density peaks. *Science*, **2014**, *344*, 1492–1496.
- [879] 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.
- [880] M. A. E. Hassan; C. R. Calladine. Two distinct modes of protein-induced bending in dna. *J. Mol. Biol.*, **1998**, *282*, 331–343.
- [881] 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.
- [882] R. Galindo-Murillo; J. C. Robertson; M. Zgarbovic; J. Spomer; 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.
- [883] S. Izadi; A. V. Onufriev. Accuracy limit of rigid 3-point water models. *J. Chem. Phys.*, **2016**, *145*, 074501.
- [884] S. Izadi; R. Anandkrishnan; 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.
- [885] 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.
- [886] 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.
- [887] 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.
- [888] 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.

- [889] L. Wang; K. A. McKiernan; J. Gomes; K. A. Beauchamp; T. Head-Gordon; J. E. Rice; W. C. Swope; T. J. Martı́nez; 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.
- [890] 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.
- [891] 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.
- [892] 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.
- [893] 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**.
- [894] J. Khandogin; C. L. Brooks, III. Constant pH molecular dynamics with proton tautomerism. *Biophys. J.*, **2005**, *89*, 141–157.
- [895] J. Khandogin; C. L. Brooks, III. Toward the accurate first-principles prediction of ionization equilibria in proteins. *Biochemistry*, **2006**, *45*, 9363–9373.
- [896] 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.
- [897] 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.
- [898] 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.
- [899] 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.
- [900] 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.
- [901] 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.
- [902] 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.
- [903] 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 trifluoromethylsulfate ([emim][otf]). *J. Chem. Phys.*, **2018**, *148*, 024503.
- [904] 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.
- [905] 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.
- [906] 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.

## BIBLIOGRAPHY

- [907] G. A. Cisneros. Application of gaussian electrostatic model (gem) distributed multipoles in the amoeba force field. *J. Chem. Theo. Comput.*, **2012**, 12, 5072–5080.
- [908] 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.
- [909] 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.
- [910] 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.
- [911] 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.
- [912] 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.
- [913] B. Leimkuhler; C. Matthews. Rational construction of stochastic numerical methods for molecular sampling. *Appl. Math. Res. eXpress*, **2013**, 2013, 34–56.
- [914] 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.
- [915] 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.
- [916] H. C. Andersen. RATTLE: A "velocity" version of the shake algorithm for molecular dynamics calculations. *J. Computat. Phys.*, **1983**, 52, 24 – 34.
- [917] N. Gr"onbech-Jensen; O. Farago. A simple and effective Verlet-type algorithm for simulating Langevin dynamics. *Mol. Phys.*, **2013**, 111, 983–991.
- [918] T. Nugent; D. T. Jones. Membrane protein orientation and refinement using a knowledge-based statistical potential. *BMC Bioinformatics*, **2013**, 14, 276.
- [919] 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.
- [920] 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.
- [921] 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.
- [922] B. Ho. pdbremix. <https://github.com/boscoh/pdbremix>, 2018.
- [923] X. Liu; J. Liu. Critical role of quantum dynamical effects in the Raman spectroscopy of liquid water. *Mol. Phys.*, **2018**, 116, 755–779.
- [924] H. C. Andersen. Molecular dynamics simulations at constant pressure and/or temperature. *J. Chem. Phys.*, **1980**, 72, 2384–2393.
- [925] 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.
- [926] D. A. McQuarrie. *Statistical Thermodynamics*. Harper and Row, New York, 1973.

- [927] 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.
- [928] H. Nguyen; D. A. Case; A. S. Rose. Nglview–interactive molecular graphics for jupyter notebooks. *Bioinformatics*, **2017**, *34*, 1241–1242.
- [929] 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.
- [930] 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.
- [931] Z.-H. Duan; R. Krasny. An adaptive treecode for computing nonbonded potential energy in classical molecular systems. *J. Computat. Chem.*, **2001**, *22*, 184–195.
- [932] P. Li; H. Johnston; R. Krasny. A Cartesian treecode for screened Coulomb interactions. *J. Computat. Phys.*, **2009**, *228*, 3858–3868.
- [933] 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.
- [934] B. Schepers; H. Gohlke. Amber-dyes in amber: Implementation of fluorophore and linker parameters into ambertools. *Journal of Chemical Physics*, **2020**, *152*.
- [935] 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.
- [936] 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.
- [937] M. Dimura; T. O. Peulen; H. Sanabria; D. Rodnin; K. Hemmen; C. A. Seidel; H. Gohlke. Automated and optimally fret-assisted structural modeling. **2019**.
- [938] 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.
- [939] A. Patriksson; D. van der Spoel. A temperature predictor for parallel tempering simulations. *Phys. Chem. Chem. Phys.*, **2008**, *10*, 2073–2077.
- [940] 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.
- [941] 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.
- [942] 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.
- [943] 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**.
- [944] 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.

## BIBLIOGRAPHY

- [945] 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.
- [946] 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.
- [947] 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.
- [948] 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.
- [949] 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.
- [950] O. H. S. Ollila; H. A. Heikkinen; H. Iwamura. Rotational dynamics of proteins from spin relaxation times and molecular dynamics simulations. *The Journal of Physical Chemistry B*, **2018**, 122, 6559–6569. PMID: 29812937.
- [951] 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.
- [952] R. G. Fernández; 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.
- [953] 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.
- [954] 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.
- [955] 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.
- [956] G. Bussi; D. Donadio; M. Parrinello. Canonical sampling through velocity rescaling. *The Journal of chemical physics*, **2007**, 126, 014101.
- [957] 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.
- [958] 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.
- [959] 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.
- [960] 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.
- [961] 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.
- [962] 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.

- [963] M. R. Machado; S. Pantano. SIRAH tools: mapping, backmapping and visualization of coarse-grained models. *Bioinformatics*, **2016**, 32, 1568–1570.
- [964] 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.
- [965] 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.
- [966] 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.
- [967] 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.
- [968] 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.
- [969] 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.
- [970] 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.
- [971] S. Meisburger; D. Case; N. Ando. Correlated motions in a protein crystal. *Nature Commun.*, **2020**, 11, 1271.
- [972] S. Meisburger; N. Ando. Correlated Motions from Crystallography beyond Diffraction. *Acc. Chem. Res.*, **2017**, 50, 580–583.
- [973] S. Boresch; M. Karplus. The role of bonded terms in free energy simulations. 1. theoretical analysis. *J. Phys. Chem. A*, **1999**, 103, 103–118.
- [974] 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.
- [975] S. Boresch. The role of bonded energy terms in free energy simulations - insights from analytical results. *Mol. Simul.*, **2002**, 28, 13–37.
- [976] 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.
- [977] 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.
- [978] 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.
- [979] T. J. Giese; D. M. York. A GPU-Accelerated Parameter Interpolation Thermodynamic Integration Free Energy Method. *J. Chem. Theory Comput.*, **2018**, 14, 1564–1582.
- [980] 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.
- [981] C. Tian; K. Kasavajhala; K. Belfon; L. Raguette; H. Huang; A. Migués; J. Bickel; Y. Wang; J. Pin-cay; 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.

## BIBLIOGRAPHY

- [982] 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**.
- [983] K. Belfon; C. Tian; J. Maier; L. Raguette; Q. Wu; C. Simmerling. RAGTAG: Rapid Amber Gpu Torsion pArameter Generator. . *In Prep*, **2020**.
- [984] 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**.
- [985] Y. Miao; A. Bhattarai; J. Wang. Ligand Gaussian accelerated molecular dynamics (LiGaMD): Characterization of ligand binding thermodynamics and kinetics. *bioRxiv*, **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- $\beta$  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. Leherite; 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.

## BIBLIOGRAPHY

- [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 cr<sub>2</sub>o<sub>3</sub> 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

1D-RISM, 37  
3D-RISM, 37, 38, 42

## A

adjust\_q, 68  
aexp, 143  
alpha, 112  
anchor\_position, 123  
anchor\_strength, 123  
apply\_rism\_force, 45  
arange, 143  
arnoldi\_dimension, 104  
asymptKSpaceTolerance, 45  
atmask, 163  
atnam, 140  
atom\_selection\_mask, 169  
atomn, 27  
auxbasis, 79  
awt, 143

## B

b\_sol, 169  
bar\_intervall, 113  
bar\_l\_incr, 113  
bar\_l\_max, 113  
bar\_l\_min, 113  
barostat, 15  
basis, 73, 75–77, 79, 80, 82, 84, 87  
beckegrid, 74  
bellymask, 12  
bridge function, 38  
buffer, 44, 45  
bulk\_solvent\_model, 169

## C

calc, 80  
cbasis, 77  
ccut, 150  
charge density, 43  
charge distribution, 43  
charge\_analysis, 83  
chelpg, 75  
chkvir, 27  
chnghmask, 22  
clambda, 108  
closure, 41, 44

cobsl, 149  
column\_fft, 20  
conflib\_filename, 104  
conflib\_size, 104  
convkey, 78  
convthre, 83  
core, 73  
corembed, 81  
corembedatoms, 81  
correlation, 79  
crgmask, 112  
csurften, 15  
cter, 145  
cut, 18  
cuv, 43  
cv\_file, 122  
cv\_i, 117  
cv\_max, 125  
cv\_min, 125  
cv\_ni, 117  
cv\_nr, 117  
cv\_r, 117  
cv\_type, 117  
cwt, 149

## D

dataset, 148  
datasetc, 149  
dcut, 149  
debug, 80, 87  
dft, 80  
dftb\_3rd\_order, 67  
dftb\_3rd\_order, 53  
dftb\_chg, 53  
dftb\_disper, 53  
dftb\_maxiter, 53  
dftb\_slko\_path, 53  
dftb\_telec, 53  
dftb\_chg, 67  
dftb\_maxiter, 67  
dftb\_telec, 67  
dftd, 82  
dftgrid, 83  
diag\_routine, 54, 67  
dielc, 18

## INDEX

dij, 149  
dipole, 74, 75, 77–79, 83  
direct correlation function, 37  
do\_dipole, 80  
do\_parallel, 87  
dobsl, 148  
do\_debugf, 27  
driven\_cutoff, 126  
driven\_weight, 126  
drms, 12, 103  
dsum\_tol, 19  
dt, 13  
dtfb\_disper, 67  
dumpfrc, 27  
dvbips, 21  
dvdl\_norest, 112  
dwt, 148  
dx0, 12  
dynamicgrid, 82  
dynlmb, 112

## E

eedmeth, 20  
eedtb dns, 20  
embed, 81  
embedatoms, 81  
emix, 143  
energy\_window, 104  
entropy, 43  
equilibration, 128  
errconv, 56  
ew\_coeff, 20  
exactdensity, 74  
excess chemical potential, 39  
exchange, 79  
exchange\_freq, 126  
exchange\_log\_file, 126  
exchange\_log\_freq, 126  
exchem, 43  
executable, 76, 83, 87  
explored\_low\_modes, 104  
extdiel, 35

## F

fcons, 163  
finalgrid, 78  
fit\_type, 73  
frameon, 21  
freezemol, 149  
frequency\_eigenvector\_recalc, 104  
frequency\_ligand\_rottrans, 104  
fswitch, 18  
fxyz, 142

## G

gamma\_ten, 15  
gamma\_ln, 14  
gammamap, 17  
Gaussian fluctuation, 39  
gbsa, 36  
gfCorrection, 44  
gigj, 148  
gms\_version, 75  
gpuids, 83  
grid, 78  
gridips, 21  
grids, 164  
grms\_tol, 55  
grnam1, 142  
guess, 79  
guv, 43

## H

harm, 123  
harm\_mode, 123  
HNC, 38, 39  
huv, 43  
hypernetted-chain approximation, 38

## I

ialtd, 140  
iat, 138  
iatr, 145  
ibelly, 11  
icfe, 108  
iconstr, 143  
icsa, 149  
id, 148  
id2o, 144  
idecomp, 108  
idftd3, 84  
iemap, 17  
ifit, 163  
ifmbar, 112  
ifntyp, 142  
ifqnt, 18  
ifsc, 111  
ifvari, 140, 141  
ig, 14  
igb, 18, 33  
igr1, 142  
ihp, 143  
image, 127  
imin, 9  
imult, 140  
infe, 117  
intdiel, 35

integration, 74  
 invwt1, 144  
 ionstepvelocities, 11  
 ioutfm, 11  
 ipnlty, 16  
 iprot, 145, 146  
 ips, 21  
 iqmatoms, 64  
 ir6, 142  
 iresid, 140  
 irest, 10  
**irism**, 19, 43  
 irstyp, 140  
 iscale, 16  
 isgend, 98  
 isgld, 97  
 isgsta, 98  
 itgtmd, 100  
 itrmax, 55, 67  
 iwrcharges, 84  
 iwrteigen, 84  
 ixpk, 142

**J**

jbasis, 77  
 jfastw, 16

**K**

k\_sol, 169  
 keywords, 87  
 KH, 38, 39  
 klambda, 108  
 kmaxqx, 65  
 Kovalenko-Hirata, 38  
 ksqmaxq, 65

**L**

lbfgs\_memory\_depth, 103  
 ligcent\_list, 106  
 ligstart\_list, 106  
 lj1264, 19  
 ljTolerance, 44, 45  
 lmod\_job\_title, 104  
 lmod\_minimize\_grms, 104  
 lmod\_relax\_grms, 104  
 lmod\_restart\_frequency, 104  
 lmod\_step\_size\_max, 105  
 lmod\_step\_size\_min, 105  
 lmod\_trajectory\_filename, 105  
 lmod\_verbosity, 105  
 lnk\_atomic\_no, 67  
 lnk\_dis, 67  
 lnk\_method, 67

logdvd1, 112  
 long-range asymptotics, 43

**M**

mapfile, 163  
 mapfit, 164  
 mask\\_update\\_period, 169  
 matrix\_vector\_product\_method, 103  
 max\_scf\_iterations, 84  
 maxcore, 78  
 maxcyc, 12, 55, 103  
 maxit, 75, 83  
 maxiter, 78  
 maxstep, 45  
 mbar\_lambda, 112  
 mbar\_states, 112  
 mcbartint, 15  
 MDIIS, 41  
 mdiis\_del, 42, 45  
 mdiis\_method, 45  
 mdiis\_nvec, 42, 45  
 mdiis\_restart, 42, 45  
 mean solvation force, 39  
 mem, 77, 80  
 method, 75, 76, 78, 79, 82, 87  
 mipso, 21  
 mipsx, 21  
 ml\_update\_period, 170  
 mlimit, 20  
 mltpro, 146  
 mode, 124  
 molfit, 164  
 molReconstruction, 46  
 monitor\_file, 124  
 monitor\_freq, 125  
 monte\_carlo\_method, 105  
 move, 163  
 mt19937\_file, 127  
 mt19937\_seed, 126  
 mtmdforce, 102  
 mtmdform, 101  
 mtmdmask, 102  
 mtmdmult, 102  
 mtmdninc, 102  
 mtmdrmsd, 102  
 mtmdstep1, 101  
 mtmdvari, 101  
 multisander, 29  
 mwords, 75  
 mxsub, 17

**N**

namr, 145

## INDEX

natr, 145  
nbflag, 20  
ndiis\_attempts, 56  
ndiis\_matrices, 56  
ndip, 148, 149  
neglgdel, 27  
netfrc, 20  
nfe\_abmd, 124  
nfe\_bbmd, 126  
nfe\_pmd, 123  
nfe\_smd, 122  
nfe\_stsm, 127  
nfft3, 19  
ng3, 44  
ngpus, 83  
nharm, 123  
ninc, 140  
ninterface, 15  
nleb, 75  
nme, 146  
nmo\_corembd, 81  
nmo\_embed, 81  
nmpmc, 146  
nmropt, 10  
noeskp, 16  
noshakemask, 16, 110  
npath, 123  
npeak, 143  
nprintlog, 80  
npropagate, 40, 45  
nprot, 145, 146  
nrad, 75  
nranatm, 27  
nrespa, 13  
nring, 145  
nscm, 12  
nsgsize, 98  
nsnb, 18  
nstep1, 140  
nstlim, 12  
ntb, 18  
ntc, 15  
nter, 145  
ntf, 17  
ntmin, 12, 103  
ntp, 15  
ntpr, 10, 55, 74, 75, 77–80, 83  
ntr, 12  
ntt, 13  
ntwe, 11  
ntwf, 11  
ntwprr, 11  
ntwr, 10

ntwrism, 46  
ntwv, 11  
ntwx, 11  
ntx, 10  
ntxo, 10  
num\_mpi\_prcs, 79  
num\_threads, 74–76, 78, 79  
number\_free\_rotrans\_modes, 105  
number\_ligand\_rotrans, 105  
number\_ligands, 105  
number\_lmod\_iterations, 105  
number\_lmod\_moves, 105  
num\_datasets, 148  
nvec, 40  
nxpk, 142

## O

obs, 145, 146  
offset, 36  
omega, 144  
optkon, 146  
optphi, 146  
order, 19  
Ornstein-Zernike, 37  
oscale, 144  
outfprefix, 87  
output\_file, 122  
output\_freq, 122  
outxyz, 142

## P

pair-distribution function, 37  
parameter\_file, 67  
parameterfile, 55  
path, 123  
path\_mode, 123  
PC+/3D-RISM, 44  
pdb\_infile, 169  
pdb\_outfile, 169  
pdb\_read\_coordinates, 169  
pencut, 17  
peptide\_corr, 55, 67  
polarDecomp, 46  
polardecomp, 40  
potUVroot, 43  
precision, 82  
pres0, 15  
print\_eigenvalues, 55  
printcharges, 55, 67  
printdipole, 66  
profile\_mpi, 22  
progress, 46  
PSE-n, 38, 39

pseudo\_diag, 54, 67

pseudo\_diag\_criteria, 54, 67

## Q

qmcharge, 53, 67

qmcut, 64

qm\_ewald, 64

qmgb, 65

qmmask, 64

qmmm\_int, 56, 66

qmmm\_switch, 65

qm\_pme, 65

qmqmdx, 53, 67

qmshake, 66

qm\_theory, 52, 66, 67

qxd, 55, 67

## R

r0, 141

raips, 21

random\_seed, 105

ranseed, 27

rbornstat, 36

rdt, 36

refin, 101

reflection\_infile, 169

release, 128

repeats, 128

report\_centers, 128

resolution, 125, 163

restart\_pool\_size, 105

restraint, 139

restraintmask, 12

restraint\_wt, 12

rgbmax, 35

RISM, 37

rismld, 37

rjcoef, 141

rmsfrc, 27

rotmin\_list, 106

rstwt, 138

rsum\_tol, 19

r\_switch\_hi, 65

r\_switch\_lo, 65

rtemperature, 105

## S

s11, 148

saltcon, 35

sander, 42

scale\_update\_period, 170

scalm, 17

scf\_conv, 74–76, 79

scf\_cyc, 87

scf\_iter, 73

scfconv, 54, 67, 78

scmask, 112

selection\_constant, 125

selection\_epsilon, 125

selection\_freq, 125

sf\_outfile, 169

sgff, 98

sgft, 97, 98

shcut, 145

shrang, 145

sigmatol, 84

skinnb, 20

smoothing, 128

snapshots\_basename, 125

snapshots\_freq, 125

solvation, 37, 39

solvation free energy, 39

solvbox, 44, 45

solvene, 43

solvent\_mask\_adjustment, 169

solvent\_mask\_probe\_radius, 169

spin, 53, 67

str, 145

surften, 36

## T

t, 13

target, 170

taumet, 144

taurot, 144

tausw, 17

temp0, 13

tempi, 14

tgfitmask, 100

tgtdmfr, 100

tgtrmsd, 100

tgtrmsmask, 100

thermodynamics, 39

threall, 82

tight\_p\_conv, 54, 67

timescale, 125

tishake, 108, 110, 111

tol, 16

tolerance, 41, 44, 45

tolpro, 146

total correlation function, 37

total\_low\_modes, 105

trmin\_list, 106

tsgavg, 97

## INDEX

### U

uccoeff, [44](#)  
use\_dftb, [74](#)  
use\_template, [74](#), [75](#), [77–79](#), [81](#), [83](#), [87](#)

### V

vdwmeth, [20](#)  
verbose, [19](#), [46](#)  
verbosity, [54](#), [67](#), [80](#)  
vlimit, [14](#)  
volfmt, [46](#)  
vshift, [56](#)  
vsolv, [67](#)

### W

write\_thermo, [46](#)  
writepdb, [66](#)  
wt, [145](#), [146](#)  
wt\_temperature, [125](#)  
wt\_umbrella\_file, [125](#)

### X

xc, [73](#)  
xmin\_method, [103](#)  
xmin\_verbosity, [103](#)  
xray\\_weight\\_final, [170](#)  
xray\\_weight\\_initial, [170](#)  
xvv, [43](#)

### Z

zerochg, [27](#)  
zerodip, [27](#)  
zerofrc, [45](#)  
zerovdw, [27](#)  
zlmfit, [73](#)