RAPTOR

The Voth Group

August 24, 2015

Contents

1 INTRODUCTION

RAPTOR (Rapid Approach for Proton Transfer and Other Reactions) is an add-on package (~82K lines of code) written for the LAMMPS MD code for the large-scale simulation of chemical reactions using a reactive multistate methodology (MS-EVB: Multistate Empirical Valance Bond).

The MS-EVB algorithm has historically been used in the Voth group to simulate proton transfer in a variety of condensed phase systems spanning liquids, proteins, and polymer fuel cell membranes. The main purpose of this code is to provide a flexible framework to enable one to take advantage of the full flexibility of the MS-EVB and related methods and model completely general chemical reactions. Another, equally important, purpose for this code is the efficient simulation utilizing high-performance computing resources with particular emphasis on reaching leadership capability: >100K processors

2 RECENT ADDITIONS

This section tries to highlight recent additions to the manual in terms of functionality and helpful guides.

1. Modified PLUMED interfaces for v1.2 and v2.0 are now included in the USER-RAPTOR/plumed directory for manual installation.
2. A new ”Common Compilation Issues” section has been added: Section 5.2.
3. A new ”in.evb-raptor.out” file that shows output from processing the evb configuration file: Section. 5.5.3.
4. A guide has been organized that may be helpful for setting up the evb configuration file for large molecule systems: Section 5.6.
5. An add-on section has been added to highlight RAPTOR support for other packages (e.g. COLVARS and PLUMED): Section 5.7.
6. The use of eigenvector overlaps can be used as the SCI convergence critieria instead of the default energy differences: Section 6.2.

3 DEVELOPERS

The following is a list of the current authors and maintainers of the RAPTOR code. Please contact anyone on this list if you run into problems running the code or would like assistance in implementing a new method.

- Yuxing Peng (UC)
- Chris Knight (ALCF)

The following is a list of people (roughly chronological) who over the years have contributed to the development and implementation of algorithms in the RAPTOR code.

- Steve Tse (UC)
- Adrian Lange (ALCF)
- Phillip Blood (PSC)
- Lonnie Crosby (NICS)
- Tianying Yan
- Yong Zhang
- Takefumi Yamashita
- Wim R. Cardoen
- Mark C. Maupin

4 LAMMPS

In this section, we briefly discuss general topics on how to compile and run the LAMMPS code. A complete and more detailed explanation of using the code is available on the LAMMPS website.

4.1 Compilation

The source code for LAMMPS, along with some instructions, is available at the following website: http://lammps.sandia.gov/download.html. There are a few options available for downloading the source.

1. Instructions are on the Voth group wiki for obtaining and keeping up-to-date the LAMMPS, PLUMED, and RAPTOR codes using cvs.
2. One can download a tarball by choosing the "LAMMPS" option and then click "Download Now" to obtain the most recent version of the source including any patches and updates. Recent changes to the code can be tracked on the LAMMPS website. LAMMPS is in active development and updates are posted here on a regular basis. After unzipping the downloaded file, you will now have a directory named "lammps-$date", where $date is the date of the most recent version. For example, the directory might be named **lammps-15Mar11**
3. Two other options which are strongly encouraged, is to use **svn** or **git** to checkout and update the source. This will allow you to easily update to the most recent source with minimal effort.

Example Makefiles for compiling the code can be found in the "MAKE" directory. A number of Makefiles for machines used by the Voth group can also be found on the group wiki. Some notable features about the LAMMPS code is that it is always linked against an MPI library, even serial executables. To compile a serial version of the code, you will need to go into the "STUBS" directory and type "make." This generates the dummy library "mpi.a" which, for the most part, contains empty MPI functions that the Lammps code will call. Now you are ready to compile the LAMMPS code. After the Makefile for the specific machine has been generated, the LAMMPS code can be compiled by typing **make foo** where "foo" corresponds to the "Makefile.foo" file located in the MAKE directory. If the compilation was successful, you should now see an executable named "lmp_foo" in the directory. It may be necessary to install certain packages before building the LAMMPS executable. These packages can be installed by typing commands like the following and then recompiling.

```
make yes-MOLECULE
make yes-KSPACE
make yes-REPLICA
make yes-CLASS2
make yes-USER-OMP
```

Note also, that everytime you update the lammps code using cvs, svn, or git, then you must reinstall the packages before recompiling the LAMMPS code to utilize the latest updates.

4.2 Input Files

All of the input filenames given in this section are user specified. The filenames can either be specified on the command line or in the main input file which controls all of the directives for the LAMMPS code. The name of the main input file is specified on the command line. In this document, we will refer to this file as "lmp.in". The filenames of all other input files are defined within this file.

4.2.1 lmp.in

This input file contains all of the control directives for running a LAMMPS simulation. The following is an example input file for a simulation of the SPC/Fw water model in the NVE ensemble. We will only briefly comment on each of the command statements since each one is well documented on the LAMMPS website and in the "lammps/doc" directory. The LAMMPS parser for the input script is very powerful and flexible allowing the user to define variables, reference additional files, and even introduce some logical "if" statements.

```
-------------------------------------------------------------------
variable        output_freq   string 5
variable        input         string lmp

units           real
atom_style      full

pair_style      lj/cut/coul/long 9.0
pair_modify     mix arithmetic

bond_style      harmonic
angle_style     harmonic
dihedral_style  none
improper_style  none

kspace_style    ewald  1e-4

read_data       ${input}.data

bond_coeff      1  529.581   1.012    # OW-HW
```

```
angle_coeff     1   37.95   113.24     #  HW-OW-HW

pair_coeff      * *  0.0         0.0
pair_coeff      1 1  0.1554253   3.165492    9.0

neighbor        2.0 bin

timestep        1.0

thermo_style    custom step temp pe etotal
thermo          ${output_freq}

fix             1 all nve
fix             2 all temp/berendsen 300.0 300.0 500.0

dump            1 all custom ${output_freq} ${input}.lammpstrj id &
                type x y z fx fy fz

velocity        all create 300.0 4928459 rot yes dist gaussian

restart         1000 ${input}.restart

run             10000

write_restart   ${input}.restart
--------------------------------------------------------------------
```

The '&' character on the line with the 'dump' command is a line continuation character.

4.2.2 lmp.data

This file contains all information to define the simulation cell vectors, coordinates, and bonding topology of the system. It can also contain force field parameters not specified in the LAMMPS input script for most forms of interaction potentials. The name of this file is given as an argument to the "read_data" directive in the LAMMPS input file. The following is an example data file for two water molecules plus an excess proton. The "Atoms" section in this example corresponds to the "full" atom_style.

```
--------------------------------------------------------------------
one excess proton in box of 2 waters
```

```
 10  atoms
  7  bonds
  5  angles
  0  dihedrals
  0  impropers

  4  atom types
  2  bond types
  2  angle types
  0  dihedral types
  0  improper types

 -10.00      10.00         xlo xhi
 -10.00      10.00         ylo yhi
 -10.00      10.00         zlo zhi

Masses

  1 16.0000   # OW
  2  1.0000   # HW
  3 16.0000   # O
  4  1.0000   # H

Pair Coeffs

  1    0.155425300  3.165492   # OW
  2    0.000000000  0.000000   # HW
  3    0.098609686  3.118508   # O
  4    0.000040458  0.000000   # H

Bond Coeffs

  1  harmonic 529.581    1.012              # OW-HW
  2  morse     88.960    2.100     1.000    # OH-HO

Angle Coeffs

  1  37.95000     113.2400     # HW-OW-HW
  2  38.74335     111.7269     # HO-OH-HO

Atoms
```

```
 1  1  1  -0.82    5.4395   -0.1280    2.7391
 2  1  2   0.41    5.3411   -0.0372    1.7444
 3  1  2   0.41    6.1952   -0.7593    3.0119
 4  2  1  -0.82    7.4395   -0.1280    2.7391
 5  2  2   0.41    7.3411   -0.0372    1.7444
 6  2  2   0.41    8.1952   -0.7593    3.0119
 7  3  3  -0.50   -2.1992   -5.6561    4.1987
 8  3  4   0.50   -2.7194   -4.8966    4.7869
 9  3  4   0.50   -1.3871   -6.0550    4.6153
10  3  4   0.50   -2.9773   -6.4079    4.0115

Bonds

     1      1       1      2
     2      1       1      3
     3      1       4      5
     4      1       4      6
     5      2       7      8
     6      2       7      9
     7      2       7     10

Angles

     1      1       2      1      3
     2      1       5      4      6
     3      2       8      7      9
     4      2       8      7     10
     5      2       9      7     10
--------------------------------------------------------------------
```

The data file is divided into several sections, each of which is explained in detail in the LAMMPS documentation. In the ”Atoms” section for the “full” atom_style, there is one line of data for each atom with a minimum of seven columns of data.

1. Atom Index
2. Molecule Index
3. Atom Type
4. Charge
5. x-coordinate

6. y-coordinate
7. z-coordinate

Three additional integers can be given for each atom which correspond to the number of integral lattice translations to be applied to the coordinates of a particle to unwrap PBC and form continuous trajectories.

4.3 Running LAMMPS

For a serial executable, the following command would be work.

```
./lmp_mac < lmp.in > lmp.out
```

For a parallel executable, it is necessary to use the “-in” option to define the input file.

```
./lmp_mac-MPI -in lmp.in > lmp.out
./lmp_mac-MPI -in lmp.in -out lmp.out
```

4.4 Output Files

4.4.1 lmp.out

This file contains output generated over the course of the simulation. The contents of this file would normally be printed to standard output. This contains some information regarding the input file and all output generated by the ”thermo_style” command.

4.4.2 log.lammps

This file contains similar output as the ”lmp.out” file, but also has a complete copy of the input file including LAMMPS generated statements as each command is processed. In the event of an error parsing the input file, this file is a good place to start since it will likely indicate the last line read by the code (if I/O is not buffered).

4.4.3 lmp.lammpstrj

This file contains the trajectory related data generated over the course of the simulation. The format and contents of this file are controlled with the “dump” keyword in the LAMMPS input script.

4.5 Tabulated Pairwise Potentials

This section will give a brief discussion on the use of tabulated potentials in LAMMPS. The example below shows the combined use of tabulated potentials with Lennard-Jones and coulomb interactions. The ”hybrid/overlay” command will accumulate the defined interactions for each pair of atoms. For the “pair_coeff” commands, the type of potential must be specified before pair-specific arguments are given. In this example, all atoms interact with one another via coulomb interactions. Atoms of type 1 interact with a Lennard-Jones potentials and atom pairs of type 1-3 and 1-4 interact with a tabulated potential. Near the bottom of the input script, the ”pair_write” command is used to write the total interaction potential between each unique pair of atom types to file. This can be used to confirm that the atoms are interacting with the desired combination of potentials. Note that the ”0.0 0.0” at the end of each ”pair_write” statement sets that atoms charges to zero, so only the non-coulombic interaction potential is written to file.

4.5.1 lmp.in

```
--------------------------------------------------------------------
units           real
atom_style      full

pair_style      hybrid/overlay lj/cut 9.0 coul/long 9.0 table linear 2000
pair_modify     mix arithmetic

bond_style      hybrid harmonic morse
angle_style     harmonic
dihedral_style  none
improper_style  none

kspace_style    ewald 1e-4

read_data       lmp.data

pair_coeff      * * coul/long
pair_coeff      1 1 lj/cut 0.1554253 3.165492  9.0

pair_coeff      1 3 table lmp.TABLE OH-OW 9.0
pair_coeff      1 4 table lmp.TABLE HH-OW 9.0

neighbor        2.0 bin

timestep        1.0
```

```
thermo_style    custom step temp pe etotal
thermo          1

fix             1 all nve
fix             2 all temp/berendsen 300.0 300.0 200.0


dump            1 all custom 20 lmp.lammpstrj id type x y z fx fy fz


velocity        all create 300.0 4928459 rot yes dist gaussian

#pair_write 1 1 101 r 1.0000 9.0000 table_1_1.dat LJ 0.0 0.0
#pair_write 1 2 101 r 1.0000 9.0000 table_1_2.dat LJ 0.0 0.0
#pair_write 1 3 101 r 1.0000 9.0000 table_1_3.dat LJ 0.0 0.0
#pair_write 1 4 101 r 1.0000 9.0000 table_1_4.dat LJ 0.0 0.0

#pair_write 2 2 101 r 1.0000 9.0000 table_2_2.dat LJ 0.0 0.0
#pair_write 2 3 101 r 1.0000 9.0000 table_2_3.dat LJ 0.0 0.0
#pair_write 2 4 101 r 1.0000 9.0000 table_2_4.dat LJ 0.0 0.0

#pair_write 3 3 101 r 1.0000 9.0000 table_3_3.dat LJ 0.0 0.0
#pair_write 3 4 101 r 1.0000 9.0000 table_3_4.dat LJ 0.0 0.0

#pair_write 4 4 101 r 1.0000 9.0000 table_4_4.dat LJ 0.0 0.0

run             20

write_restart   lmp.restart
--------------------------------------------------------------------
```

4.5.2 lmp.tab

This file contains tabulated pairwise potentials for interatomic pairs. For each interaction potential, some header information is given before the tabulated potential is listed. For each radial grid point, there are four columns of data given: index, radial distance, potential energy, and force.

One important point about the format of this file is that internally, LAMMPS will stored the interpolated potential on a grid in r^2, not r. If the table is given as input in units of r, then internally, the grid spacing at the shortest distances may be too large to capture the important features of the interaction potential. This would be seen in output from the "pair_write" command when compared to the tabulated potential used as input. In the example given below, there are only 101 grid points over the distance range 0.5 - 9.0 Å. Internally, LAMMPS will calculate the bin size for a grid on r^2 as $(9^2 - 0.5^2)/101 = 0.7995\text{Å}^2$ to construct a list of evenly spaced control points. The spacing between control points in r will be largest at the shortest distances and decrease as r increases. The increased spacing at shortest distances may lead to an inaccurate representation of the tabulated potential. There are two options to prevent this from becoming a problem: 1) generate the table in "lmp.TABLE" on a grid in r^2 and use the "RSQ" command as discussed in the documentation or 2) use a sufficiently large number of grid points so that the difference between the two spacings (r vs. r^2) does not significantly affect the interpolated table.

```
----------------------------------------------------------------------
# Pair Potential: i, r, energy, force

OH-OW
N 101

         1       0.5000  1877619986.39058000  45063245587.30170000
         2       0.5850   285342406.25474800   5853299485.02925000
         3       0.6700    56016547.54796780   1003328616.39114000
...
        99       8.8300          -0.00100316           -0.00068026
       100       8.9150          -0.00094723           -0.00063629
       101       9.0000          -0.00089490           -0.00059552
# Pair Potential: i, r, energy, force

HH-OW
N 101

         1       0.5000       10159.11665801       243940.08888324
         2       0.5850        1541.51869433        31661.30925852
         3       0.6700         301.66437281         5418.57864023
...
        99       8.8300          -0.00000033           -0.00000023
```

```
         100       8.9150           -0.00000031             -0.00000021
         101       9.0000           -0.00000030             -0.00000020
----------------------------------------------------------------------
```

5 RAPTOR

In this section, we discuss general features of the RAPTOR add-on package for the LAMMPS molecular dynamics software package.[?,?] This will include discussion on topics such as compilation of the code, running the program in serial and parallel, as well as the structure of the input and output files with several examples.

5.1 Compilation

The LAMMPS source code along with instructions to compile serial and parallel versions were discussed above. After having succesfully compiled a working LAMMPS executable, the following steps can be used to compile the RAPTOR package. For the first time usage, a tarball containing the RAPTOR code can be download from the group wiki page. Documentation, examples, and other related files can also be found there. The RAPTOR code is organized similar to all other user packages, so the USER-RAPTOR directory should be placed in the lammps/src directory.

As long as cvs is installed on the machine, you should be able to use the command ”cvs update” inside the USER-RAPTOR directory to grab the most up-to-date version of the raptor code. The RAPTOR code can then be compiled like any other user package by typing the following commands in the lammps/src directory.

```
make yes-USER-RAPTOR
make foo
```

After rerunning the ”make foo” command, one should find a RAPTOR enabled LAMMPS executable ”lmp_foo”.

5.2 Resolving common compilation issues

- Make sure the version of LAMMPS is consistent with that expected from RAPTOR.

  ```
  % make yes-USER-RAPTOR
  Installing package USER-RAPTOR
  [USER-RAPTOR] Check the SVN revision number of LAMMPS...(12990 is required)
  [...]
  ```

- When installing a new version of LAMMPS, make sure to clean up the /src directory before running the ”svn update” command.

  ```
  \\ Check what packages are installed (USER-RAPTOR won't appear)
  cd lammps/src
  make ps | grep 'YES'

  make no-USER-RAPTOR
  make no-all
  svn update -r 12990

  \\ Reinstall desired packages

  \\ Make sure new version of LAMMPS compiles just fine.
  make -j 4 mac

  \\ Reinstall USER-RAPTOR (two calls needed to initialize headers)
  make yes-USER-RAPTOR
  make yes-USER-RAPTOR
  make -j 4 mac
  ```

- When compiling USER-RAPTOR and errors related to TIMER_EVB targets appear (usually after an update), simply run ”make yes-USER-RAPTOR” a second time to generate a correct EVB_timer_const.h file.

  ```
  make mac

  [...]

  ../EVB_effpair.cpp(863): error: identifier "TIMER_EVB_EffPair_compute_finter"
      is undefined TIMER_STAMP(EVB_EffPair,compute_finter);
      ^
  ```

```
../EVB_effpair.cpp(934): error: identifier "TIMER_EVB_EffPair_compute_finter"
    is undefined TIMER_CLICK(EVB_EffPair,compute_finter);

\\ Generate new EVB_timer_const.h and EVB_cracker.h files
make yes-USER-RAPTOR
make mac
```

5.3 DLEVB compatibility

Due to slight differences in implementation, some minor changes are necessary for the RAPTOR code to reproduce the results of the older DLEVB code. These changes affect details of the state search algorithm and the manner in which scaled 1-4 interactions are treated. These changes are particularly important to reproduce results from amino acid models developed with the DLEVB code.

In the DLEVB code, certain pairs of scaled 1-4 interactions (both LJ & Coulomb) were neglected when atoms in the 1-4 pair crossed the boundary between the reactive complex and the environment atoms. These interactions are properly accounted for in the RAPTOR code, but their inclusion with the older models introduces an ”effective” diabatic shift which can easily be seen when calculating potentials of mean force for deprotonating the amino acids.

In the state search algorithm, the use of a hybrid MS-EVB2 & MS-EVB3 algorithm is necessary. In the event that a hydronium or amino acid H-bond is donated to more than one water molecule, only the water with the shortest H-bond distance is retained, but only after a full MS-EVB3 state search has been completed at each reactive shell level. There are also specific distances used for the cutoffs to determine whether a state should be included or not. These are hard-coded in the EVB_Complex::search_state() function to define the ”shell_rcut” variable.

To enable these changes to run simulations with the older amino acid models, it is necessary to add the preprocessor flag “-DDLEVB_MODEL_SUPPORT” to the Lammps makefile.

5.4 Input Files

All of the input filenames given in this section are somewhat arbitrary since they are defined as the arguments to commands in the LAMMPS input script.

5.4.1 lmp.in

This input file contains all of the control directives for running a LAMMPS simulation. With the exception of a few additional directives, the contents of this file are identical to one that is used to run a regular LAMMPS simulations. In fact, only one additional command needs to be given to turn a nonreactive simulation into a reactive simulation assuming, of course, that the necessary MS-EVB related input files are present.

```
--------------------------------------------------------------------
variable        output_freq   string 5
variable        input         string lmp

units           real
```

```
atom_style      full

pair_style      lj/cut/coul/long 9.0
pair_modify     mix arithmetic

bond_style      harmonic
angle_style     harmonic
dihedral_style  none
improper_style  none

kspace_style    ewald  1e-4

read_data       ${input}.data

bond_coeff      1  529.581   1.012    # OW-HW

angle_coeff     1   37.95   113.24     #  HW-OW-HW

pair_coeff      * *  0.0         0.0
pair_coeff      1 1  0.1554253   3.165492    9.0

neighbor        2.0 bin

timestep        1.0

thermo_style    custom step temp pe etotal
thermo          ${output_freq}

fix             evb all evb in.evb evb.out ${input}.top
fix             1 all nve
fix             2 all temp/berendsen 300.0 300.0 500.0

dump            1 all custom ${output_freq} ${input}.lammpstrj id type &
                x y z fx fy fz

velocity        all create 300.0 4928459 rot yes dist gaussian

restart         1000 ${input}.restart

run             10000

write_restart   ${input}.restart
```

```
------------------------------------------------------------------
```

The only statement that was added to turn on chemical reactivity in a multistate simulations was the following.

```
fix             evb all evb in.evb evb.out ${input}.top
```

This proves to be especially convenient in that a reactive simulation can be turned into a nonreactive simulation simply by commenting the one line in the input script that defines the evb fix. No other changes are necessary to switch between reactive and nonreactive simulations. The fix_evb object must be declared before any integrator. The three file names given are those of the input, output, and topology files. The format and contents of each input file will be discussed in detail below.

5.4.2 lmp.data

The format of this file is the same as for a nonreactive simulation. No changes to this file are necessary to run an MS-EVB simulation.

5.4.3 in.evb

This file defines several settings specific to reactive simulations and the topological information for those molecules with variable chemical identity in a reactive simulation, which includes reactive molecule templates, definitions of chemical reactions, rules for the state search algorithm, and model parameters for the off-diagonal couplings and diabatic corrections. The main contents of this input file is to make a one-to-one connection between the LAMMPS and RAPTOR types defining atoms, molecules, bonds, angles, etc... The first section contains options to control the frequency and amount of output written.

Comments can be placed in the file by adding a ":" character. These characters can appear anywhere on the line and will cause all remaining characters on the line to be ignored by the RAPTOR code. Unless stated otherwise, '0' means false/off and '1' means true/on.

```
------------------------------------------------------------------
::::::::::::::::::::
:: Output Settings
::::::::::::::::::::

[segment.output]

0 : output frequency, 0 means obey lammps setting
0 : if output for reaction
```

```
0 : Binary file
0 : flush every output

1 : output center location
1 : output # of states
1 : output states infomation
1 : output energy
1 : decompose energy

[segment.end]
--------------------------------------------------------------------
```

The next section defines a number of variables that will be referenced in later sections of this file.

```
--------------------------------------------------------------------
::::::::::::::::::::
:: Declare Modules
::::::::::::::::::::

::: Define models
#define EVB3
#define WAT
#define HYDRONIUM

::: Set molecule types
settype H2O 1
settype H3O 2

::: Define reaction Types
#define H3O_H2O
--------------------------------------------------------------------
```

At this point, three variables have been defined and the value of two variables have been set.

```
--------------------------------------------------------------------
::::::::::::::::::::::::::::::::::::::::::::
:: Declare atom, bond, angle, etc... types
::::::::::::::::::::::::::::::::::::::::::::

: --------- WATER ---------
#ifdef WAT
```

```
 #define OW    1          : Oxygen of Water
 #define HW    2          : Hydrogen of Water
 #define OW-HW   1        : Water Bond
 #define HW-OW-HW   1     : Water Angle

#endif

#ifdef HYDRONIUM

 #define OH 3          : Oxygen of Hydronium
 #define HH 4          : Hydrogen of Hydronium
 #define OH-HH 2       : Hydronium Bond
 #define HH-OH-HH 2   : Hydronium Angle

#endif

#include "evb.par_4amber"
---------------------------------------------------------------------
```

For each reactive molecule, the atom, bond, angle, etc... types are defined (Figure 1). The variables ”OW”, ”HW”, ”OH-HW”, ”HW-OW-HW”, etc... are not necessary, but prove helpful in keeping the contents of the remaing portion of this file organized. The last line of the ”in.evb” file references the ”evb.par_4amber” file that will be included as part of the ”in.evb” file. The parameter file contains all definitions relating to the chemical reactions that will be simulated and is discussed in the next section.

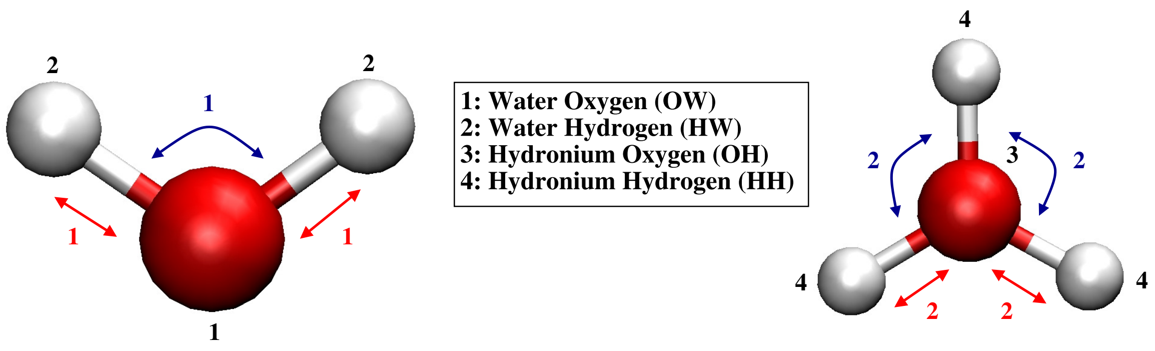

Figure 1: Atom, bond, and angle definitions for the two molecules involved in the chemical reaction. The black numbers beside each atom indicate the atom type as defined in the center text box, the red numbers beside each straight arrow indicate the bond type, and the blue numbers next to the curved arrows indicate the angle type. Each of these molecules provide a template for the EVB kernels.

5.4.4 evb.par_4amber

The contents of this file is divided into several sections.

1. Definitions for the EVB kernels (nonreactive molecule templates)
2. Definitions for changing topologies during a chemical reaction
3. Definitions for state search algorithm
4. Off-diagonal couplings
5. Short-range repulsive interactions
6. Supplemental

Each of the sections is discussed in detail below. Within each of these sections are various "#ifdef" statements and IF-ELSE constructs to determine whether certain sections should be activated. For example, in the "in.evb" file discussed above, only water and hydronium molecules were defined, therefore, only the corresponding sections would be included while all others ignored.

1: Definitions for the EVB kernels
In this section, details on the topology of all molecules involved in chemical reactions are specified. Associated with each unqiue molecule present at any point in a simulation is a template to define the corresponding topology and force field parameters. For the case of the excess proton in water, there are two unqiue molecules to be considered: water and the hydronium cation (Figure 2).

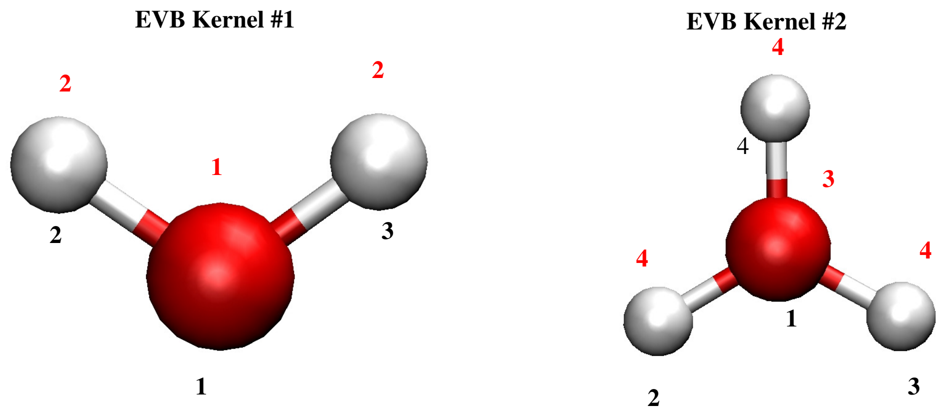

Figure 2: Definition of the two EVB templates for the water molecule and hydronium cation. The black numbers below each atom indicate the atom index within the molecule. The red numbers above each atom indicate the atom type and correspond to the definitions set in the "in.evb" file.

The start and end of this section is indicated with the following statements.

```
[segment.molecule_type]
...
[segment.end]
```

All molecule templates are to be defined in this section. The start and end of each molecule template is denoted by the statments

```
[molecule_type.start.###]
...
[molecule_type.end]
```

, where the ### is the name of a molecule. For each molecule template, the first line consists of a list of integers to specify counters for the topology: number of atoms, bonds, angles, dihedrals, and improper torsions. The last entry in this list is either a 0 or 1 depending on whether this molecule is used to initiate a state search. For each atom in the molecule, a line of information is given defining the atom type, charge, and a 0 or 1 to indicate whether each atom is considered to be ”active” in the sense that its topological information is allowed to change over the course of a simulation. The next sections define each of the bonds, angles, dihedrals, and improper torsions used to define the connectivity of the molecule. If the molecule can be used to initiate a statesearch, then a center-of-charge (COC) is typically calculated. The last couple of lines indicate how many atoms in the molecule are used to compute the COC. The following line gives a list of the atom indices used in the calculation of the COC. For the case of the hydronium cation, all four atoms contribute. Notice that variables for atom, bond, and angle types defined earlier are used here. Also, this section will only be parsed if the ’HYDRONIUM’ variable is defined and otherwise ignored.

```
--------------------------------------------------------------------
[segment.molecule_type]
#ifdef HYDRONIUM
[molecule_type.start.H3O]

:       HH        2
:       |         |
:       OH        1
:      / \       / \
:   HH   HH    3   4

: number of atoms,  bonds,  angles, dihedrals,  impropers,  starting rc
             4        3       3        0           0         1
: atomic information
: atom type   charge   kernel
  OH          -0.5      1
  HH           0.5      1
  HH           0.5      1
  HH           0.5      1
: bonds
:  atom 1       atom 2      type
   1            2           OH-HH
   1            3           OH-HH
   1            4           OH-HH
: angles
:  atom 1       atom 2       atom 3     type
   2            1            3          HH-OH-HH
   2            1            4          HH-OH-HH
   3            1            4          HH-OH-HH
: COC
4
1 2 3 4
[molecule_type.end]
#endif
----------------------------------------------------------------------
```

The next molecule template is for a water molecule defined by the variable “H2O”. For the water molecule, the charges used will depend on the choice of MS-EVB model used. This is an example of how a single template can be defined with multiple definitions. Depending on which variable is defined, only one definition will be used enabling one to write a single ”evb.par_4amber” file and easily switch between models with small changes to the ”in.evb” file. In this example, the two possible choices are the MS-EVB2 and MS-EVB3 models defined using the variables “EVB2” and “EVB3”, respectively. In the “in.evb” file, the “EVB3” variable was defined, so the contents of the “EVB2” sections will be ignored. If one instead wanted to run a simulation with the MS-EVB2 model, one would only need to change the definition in the ”in.evb” file (#define EVB2). In this simulation, since a water molecule is not used to initiate a state search, no definition for a COC is necessary. Note also the zero as the last entry on the first line of integers indicating this molecule does not serve as a reactive center.

```
----------------------------------------------------------------------
#ifdef WAT
[molecule_type.start.H2O]

: HW---OW---HW      2---1---3

: number of atoms,  bonds,  angles, dihedrals,  impropers,  starting rc
              3       2       1        0          0          0
: atomic information
: atom type   charge   kernel
```

```
  #ifdef EVB3
    OW          -0.82     1
    HW           0.41     1
    HW           0.41     1
  #endif

  #ifdef EVB2
    OW          -0.834    1
    HW           0.417    1
    HW           0.417    1
  #endif

: bonds
:  atom 1       atom 2      type
   1            2           OW-HW
   1            3           OW-HW
: angles
:  atom 1       atom 2       atom 3     type
   2            1            3          HW-OW-HW
[molecule_type.end]
#endif
---------------------------------------------------------------------
```

2: Definitions for changing molecular topologies during chemical reaction

In this section, the rules for the state search algorithm that will be used to modify the molecular topology over the course of a simulation. The start and end of this section is denoted by the following statements.

```
[segment.reaction]
...
[segment.end]
```

The definition for each chemical reaction begins and ends with the following statements.

```
[reaction.start.###]
...
[reaction.end]
```

, where the "###" is a reaction name. In this example, the "H3O_H2O" variable is used to activate only a handful of reactions althrough many more may be present in the "evb.par_4amber" file. The beginning of the section indicates the molecules involved in the chemical reaction, the type of reaction, and the number of possible pathways this reaction can proceed. For the case of the hydrated proton, there is only one reaction to define:

$$\text{Reaction Index 1 :} \quad \begin{aligned} &H_3O^+ + H_2O \rightarrow H_2O + H_3O^+ \\ &A_{\text{old}} + B_{\text{old}} \rightarrow A_{\text{new}} + B_{\text{new}}. \end{aligned} \tag{1}$$

```
--------------------------------------------------------------------
#ifdef H3O_H2O

[reaction.start.H3O_H2O]

: For reaction pair 1 :  hydronium + water -> water + hydronium
0             : Does atom move from mol. B to mol. A?  0-forward,  1-backward
H3O   H2O     : change of molecule A (Hydronium->Water)
H2O   H3O     : change of molecule B (Water->Hydronium)
3             : number of possible pathways

--------------------------------------------------------------------
```

The first line is an integer indicating the reaction type. The following options are possible, but some may not yet be fully supported.

- **0**: Atom(s) transfer from molecule A to molecule B
- **1**: Atom(s) transfer from molecule B to molecule A
- **2**: Molecule decomposes into two molecules (in development).
- **3**: Two molecules synthesize a new molecule (in development).
- **4**: Multiple template reaction with N = 2 (in development).

On the reactant side of the chemical equation, the hydronium cation is defined to be molecule A and the water molecule is molecule B, as shown in Eq. (1). In this case, since the direction of atoms transferring is from A to B, the reaction type is zero. The next two lines define the change in identity for both molecules. The hydronium cation transforms into a water molecule after donating a proton and the original water molecule is changed into a hydronium cation. For atom transfer reactions (forward or backward), the convention is taken that molecule B at the end of a reaction is the new reactive center. This means that molecule B on the product side of the reaction should be the hydronium cation.

For this reaction, since the hydronium cation has the ability to donate any of the three protons, there are three different paths possible for transferring a proton from the hydronium cation to a nearby water molecule. Detailed information regarding how the topology changes for each of these three pathways is defined next.

```
------------------------------------------------------------
...
: Pathway 1
: A description of reaction contains three parts:
1    3    3   : number of moving atoms, unchanged in A, unchanged in B
: moving atoms:
2    2        : atom 2 of reactant ==> atom 2 in product (HH->HW)
: mol A: ( first type in the pair )
1    1        : atom 1 in reactant ==> atom 1 in product (OH->OW)
3    2        : atom 3 in reactant ==> atom 2 in product (HH->HW)
4    3        : atom 4 in reactant ==> atom 3 in product (HH->HW)
: mol B: ( second type in the pair )
1    1        : atom 1 in reactant ==> atom 1 in product (OW->OH)
2    3        : atom 2 in reactant ==> atom 3 in product (HW->HH)
3    4        : atom 3 in reactant ==> atom 4 in product (HW->HH)
...
------------------------------------------------------------
```

The definition for each reactive pathway starts with a line containing three integers. The first number indicates how many atoms will be transfered between molecules. The next two numbers indicate how many atoms remain unchanged in molecules A and B, respectively. For the case of hydronium and water, a single hydrogen atom will be transfered between the molecules. The hydronium cation has three hydrogen atoms that could potentially be transfer and the corresponding atom ids are 2, 3, and 4 as defined previously in the hydronium cation template. We first consider the case when the first hydrogen atom with atom index equal to 2 is transferred to a water molecule to become part of the new hydronium cation. The remaining three atoms of the original hydronium (1, 3, 4) form the new water molecule (1, 2, 3). The three atoms of the original water molecule (1, 2, 3) form the new hydronium (1, 3, 4).

The reactive pathways for the other two hydrogens of the hydronium cation are similarly defined.

```
------------------------------------------------------------
...
: Pathway 2
1    3    3   : number of moving atoms, unchanged in A, unchanged in B
: moving atoms:
3    2        : atom 2 of reactant ==> atom 2 in product (HH->HW)
: mol A: ( first type in the pair )
1    1        : atom 1 in reactant ==> atom 1 in product (OH->OW)
2    2        : atom 3 in reactant ==> atom 2 in product (HH->HW)
4    3        : atom 4 in reactant ==> atom 3 in product (HH->HW)
: mol B: ( second type in the pair )
1    1        : atom 1 in reactant ==> atom 1 in product (OW->OH)
```

```
2    3        : atom 2 in reactant ==> atom 3 in product (HW->HH)
3    4        : atom 3 in reactant ==> atom 4 in product (HW->HH)

: Pathway 3
: A description of reaction contains three parts:
1    3    3   : number of moving atoms, unchanged in A, unchanged in B
: moving atoms:
4    2        : atom 2 of reactant ==> atom 2 in product (HH->HW)
: mol A: ( first type in the pair )
1    1        : atom 1 in reactant ==> atom 1 in product (OH->OW)
2    2        : atom 3 in reactant ==> atom 2 in product (HH->HW)
3    3        : atom 4 in reactant ==> atom 3 in product (HH->HW)
: mol B: ( second type in the pair )
1    1        : atom 1 in reactant ==> atom 1 in product (OW->OH)
2    3        : atom 2 in reactant ==> atom 3 in product (HW->HH)
3    4        : atom 3 in reactant ==> atom 4 in product (HW->HH)

[reaction.end]
#endif     : #ifdef H3O_H2O

[segment.end]
---------------------------------------------------------------------
```

One important thing to note hear is that the transferred proton in all reaction pathways is always set to have an atom index of 2. As we will see below, this proves to be very convienent.

3: Definitions for state search

In this section, we define the rules that make up the state search algorithm for the current simulation using the reaction pathways specified in the previous section. The beginning and end of this section are denoted by the following statements. This section starts with a single integer specifying whether states are refined during the search and if extra off-diagonal couplings activated, i.e. whether the MS-EVB2 or MS-EVB3 state-search algorithm will be used. *Work in development will enable the user to have a finer control over details such as these for the state search algorithm.*

```
---------------------------------------------------------------------
[segment.state_search]
: refinement of MS-EVB states
#ifdef EVB3
  0
#endif

#ifdef EVB2
  1
```

```
#endif

...
[segment.end]
------------------------------------------------------------------
```

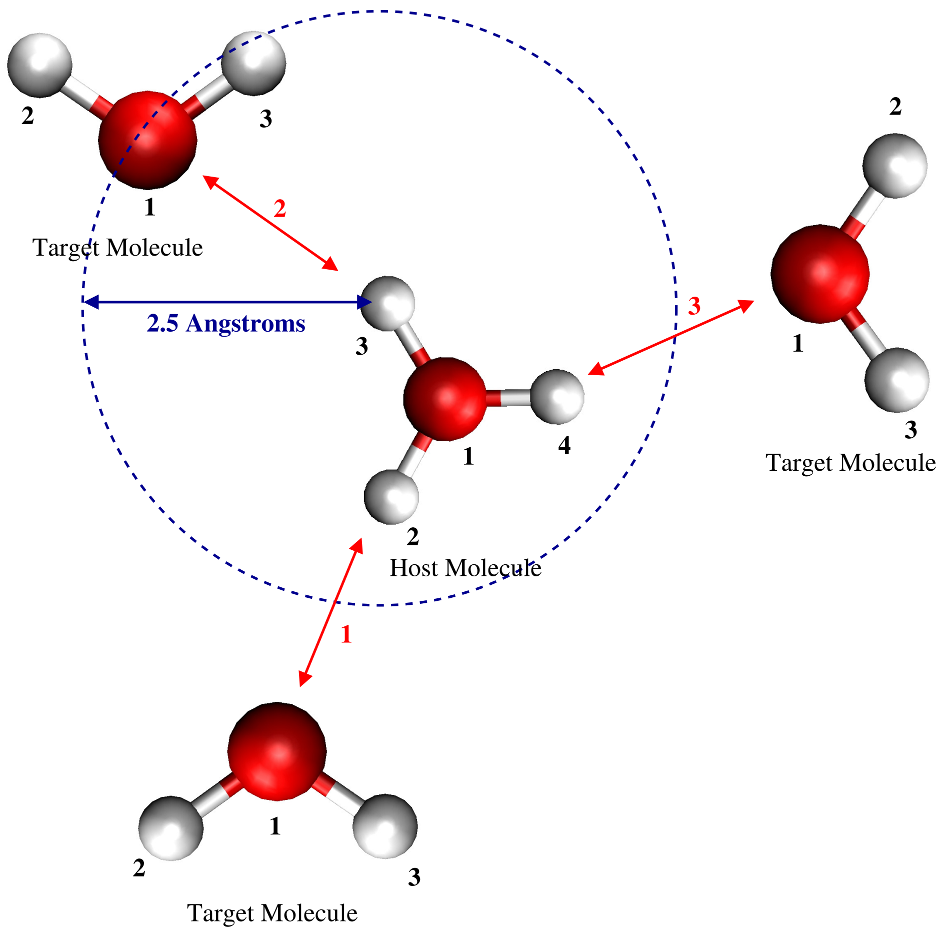

Figure 3: An example of the initial search in the state search algorithm starting from a hydronium cation. There is an oxygen atom on a target molecule that is within the cutoff distance from the third hydrogen on the host molecule. The red arrows indicate which set of rules (reaction path) are used to change the topology as defined in the "[state_search.start.H3O]" section.

The next sections, one for molecule that serves as the starting point of a simulation, will outline the rules used to define the state search. The start and end of a section for each molecule are denoted by the following statements.

```
------------------------------------------------------------------
[state_search.start.###]
```

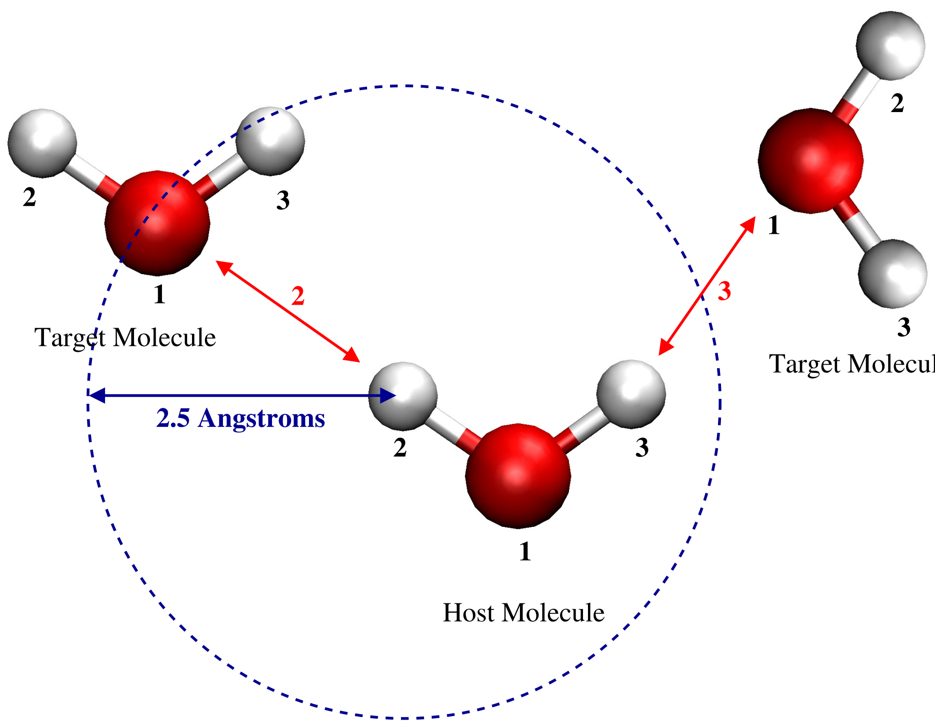

Figure 4: An example of an intermediate step in the state search for which a hydronium cation has already donated a proton to a neighboring water molecule. The atom that was previously part of the original hydronium cation is not used to search for additional MS-EVB states. There is an oxygen atom on a target molecule that is within the cutoff distance from the second hydrogen on the host molecule. The red arrows indicate which set of rules (reaction path) are used to change the topology as defined in the "[state_search.start.H2O]" section.

```
...
[state_search.end]
```

, where the "###" is a molecule type that has been defined to initiate a chemical reaction: "H3O" or "H2O" in this case. *An important point to note is that with the current implementation in RAPTOR, rules are also given for the "H2O" molecule to continue along a reaction pathway. This is because the molecule types are not maintained for each unique reaction path generated by the state search and any newly formed hydronium cation is still labeld as having a molecule type of a water molecule, although the bonding topology is updated to form a hydronium cation. This is one reason why the transferring proton was always listed as the second atom in the new hydronium cation. Work currently underway to generalize and extend the state search algorithms will remove this inconsistency.*

```
#ifdef HYDRONIUM
[state_search.start.H3O]
```

```
: host; target; client; shell limit; distance limit; rxn. pair; rxn. path
  : H->OW

  #ifdef H3O_H2O
2   H2O     1     3       2.5         H3O_H2O             1
3   H2O     1     3       2.5         H3O_H2O             2
4   H2O     1     3       2.5         H3O_H2O             3
  #endif

[state_search.end]
#endif    : #ifdef HYDRONIUM

#ifdef WAT
[state_search.start.H2O]
: host; target; client; shell limit; distance limit; rxn. pair; rxn. path
: H->OW

#ifdef H3O_H2O
2   H2O     1     3       2.5         H3O_H2O             2
3   H2O     1     3       2.5         H3O_H2O             3
#endif

[state_search.end]
#endif   :#ifdef WAT

[segment.end]
---------------------------------------------------------------------
```

This first section for hydronium cation says that once we find ourselves standing on a hydronium cation during the state search, there are three reactions that can be attempted from the list of possible reactions defined in the previous section. If over the course of a state search, we find ourselves sitting on a water molecule, then there are two reactions to be attempted. *Again, a water molecule only shows up in this section for the current implementation of RAPTOR because the molecule types are not separately maintained for each reaction pathway.* For each reaction, the entries are defined below.

1. **host:** This is the atom index in the current molecule that can be used in the distance criteria to determine if a reaction can be attempted.

2. **target:** This is the molecule type that we are searching for.

3. **client:** This is the atom index in the molecule type being searched for that will be used as the second atom to check the distance criterion for including a state.

4. **shell limit:** This is the maximum number of steps along a reaction path to attempt. *Note that this is commonly referred to as a reactive solvation shell, but should not be confused with a solvation shell. It is not unrealistic for a molecule three steps along a reaction pathway to contribute to the first or second solvation shell of a molecule. The expression "reactive solvation shell" can begin to introduce additional confusion when considering synthesis and decomposition type reactions.*

5. **distance limit:** This is the maximum distance, in internal units as specified in the LAMMPS input script, for which a reaction will be considered. A state is only included if the host and client atoms have a separation within this tolerance.

6. **rxn. pair:** This is the type of reaction that occurs when the distance criterion is satisfied.

7. **rxn. path:** Of the possible pathways for the selected reaction, this integer indicates that one that will be used to change the topology for this reaction.

The first line in the "H2O" section says that if we find ourselves on a water molecule, we will then search for the next step in a reaction pathway starting from either the second or third atom in the water molecule, ie. the hydrogen atoms. From this hydrogen atom (host), we are looking for the oxygen atom (client) of another water molecule (target) that is within 2.5 Å (typical internal unit of length). We only look for next step in a reaction pathway if the number of steps in the pathway has not exceeded the shell limit. Of the three pathways defined for the "H3O_H2O" reaction, the second and third options are used to change the chemical bonding topology.

5: Off-Diagonal Couplings

This section defines the parameters for the off-diagonal couplings for each reaction defined in a simulation. The section begins and ends with the following statements.

```
[segment.off_diagonal]
...
[segment.end]
```

The format for this section is variable and depends on the specific form for the off-diagonal coupling. When in doubt, one can examine the data_offdiag() function in the "EVB_offdiag_###.cpp" files for exactly what info is required. A complete list of currently supported off-diagonal couplings is given below with a brief description.

1. **PT**: The analytic function as defined for the MS-EVB3 model.[?]

2. **VIJ**: This sets the off-diagonal to a constant.

3. **PT_FR_Table**: This off-diagonal is for reactions of a hydronium cation with water and defines the geometric factor as a tabulated function of the oxygen-oxygen distance.

4. **HYDROXIDE_FR_TABLE**: This is for reaction of a hydroxide ion with water and defines the geometric factor as a tabulated function of the oxygen-oxygen distance.

5. **DA_TABLE**: This is a generic off-diagonal coupling that defines the geometric factor as the tabulated function of the separation between two atoms.

6. **DA_GAUSSIAN**: This is off-diagonal coupling defines the geometric factor as a Gaussian function of the separation between two atoms.

```
------------------------------------------------------------
#ifdef H3O_H2O

[off_diagonal.start.H3O_H2O]

PT  : Use the PT coupling

 : Geometry part
 : atom index
 1  1              : evb_index of DONOR atom (should be in molecule A)
 2  1              : evb_index of ACCEPT atom (should be in molecule B)
 2  2              : evb_index of TRANSFER atom (should be in molecule B)

    : A_Rq Type : A(R,q) = f(R) * g(q)
 1 : 1-symmetric; 2-asymmetric
 ...
------------------------------------------------------------
```

In this example, the ”PT” keyword is used corresponding to the functional form used in the MS-EVB3 paper. The complete list of off-diagonaly couplings is given below. The format of the remaining input for this repulsive interaction depends on which definition was used. The next three lines give the atom indices used to evaluate the three-body expression for the off-diagonal coupling between MS-EVB states. The last line defines whether a symmetric or asymmetric definition for the reaction coordinate is to be used. For the case of the reaction between water and hydronium, a symmetric reaction coordinate is selected. The parameters for the interactions are given next depending on whether the MS-EVB2 or MS-EVB3 model is used for the simulations.

```
------------------------------------------------------------
...
  #ifdef EVB3
  : parameters
   1.8302895      : g
   0.2327260      : P
  9.562153       : k
  2.94           : D_oo
  6.0179066      : b
  3.100          : R0_oo
  10.8831327     : P'
  10.0380922     : a
  1.8136426      : r0_oo

: Potential part
  -23.1871874        : vij_const, in kcal/mol
    1                : if contains Vij_ex part

 : exchanged charge
  H2O  H3O : Types of molecule_A (Water) and molecule_B (Hydronium)

  -0.0895456
   0.0252683
   0.0252683
  -0.0895456
   0.0780180
   0.0252683
   0.0252683
 #endif

 #ifdef EVB2
 : parameters
  1.85           : gamma
  0.27           : P
  11.5           : k
  2.875          : D_oo
  4.50           : beta
  3.14           : R0_oo
  10.0           : P'
  12.0           : alpha
  1.88           : r0_oo

: Potential part
  -32.419        : vij_const, in kcal/mol
    1                : if contains Vij_ex part

 : exchanged charge
  H2O  H3O : Types of molecule_A (Water) and molecule_B (Hydronium)
```

```
   -0.116725
    0.04669
    0.04669
   -0.116725
    0.04669
    0.04669
    0.04669
  #endif
[off_diagonal.end]
[segment.end]
-----------------------------------------------------------------
```

The first section corresponds to the model parameters in the expression for the geometric factor. The off-diagonal constant is given next in units of energy consistent with the Lammps simulation. The next integer indicates whether the manner in which an electrostatic component is included in the calculation of the off-diagonal coupling. If zero, no electrostatic interaction will be calculated for the off-diagonal coupling. This is common for protonatable amino acid models. If the value of one is given, then electrostatics will be calculated using the method defined in the Lammps input script, typically a k-space method. If the value of two is given, then a Debye screened Coulomb sum is used to evaluate the electrostatic interactions as opposed to method specified in the Lammps input script. This will typically replace the computationally expensive k-space call (especially PPPM) with a cheaper alternative. If selected, then the Debye screening factor should be included on the same line. Alternative values of 3 and 4 can be used to activate short-ranged approximations using a Wolf formulation[?] or CGIS-derived expression,[?] respectively.

```
2  0.5       : if contains Vij_ex part, Debye screening for off-diagonal only
```

If an electrostatic method is used for the off-diagonal coupling, then the charges on atoms in the transition-type complex will be given next. The general form for the off-diagonal coupling is defined as follows.

$$V_{ij} = (V_{ij}^{c} + V_{ij}^{ex})A(R,q) \tag{2}$$

The geometric factor is written as a product of two functions: one complicated and the other a simple Gaussian.

$$A(R,q) = f(R)g(q) \tag{3}$$

$$\tag{4}$$

The more complicated of the two functions, $f(R)$, can be decomposed into three separate contributions: a Gaussian times the sum of a Tanh and decaying expoential.

$$\begin{aligned}
f(R) &= f_G(f_T + f_E) && (5)\\
f_G &= 1 + Pe^{-k(R-D_{OO})^2} && (6)\\
f_T &= \frac{1}{2}(1 - \tanh[\beta(R - R^0_{OO})]) && (7)\\
f_E &= P'e^{-\alpha(R-r^0_{OO})} && (8)\\
g(q) &= e^{-\gamma q^2} && (9)
\end{aligned}$$

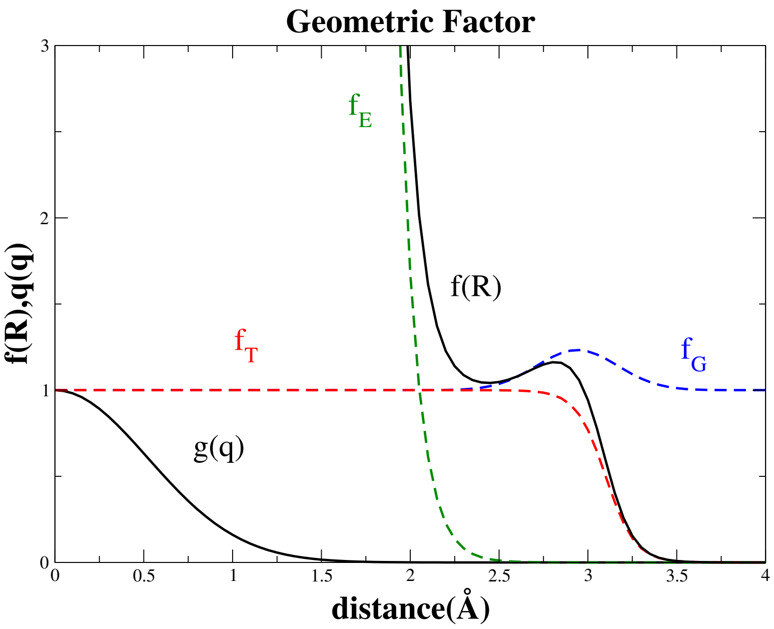

Figure 5: Contributions to the off-diagonal geometric factor for the MS-EVB3 model.

6: Short-Ranged Repulsive Interactions

This section defines the short-ranged repulsion a reactive species and the surrounding molecules. The section begins and ends with the following statements.

```
[segment.repulsive]
...
[segment.end]
```

For each reactive species that initiate a state search, a set of repulsive interactions are expected. In the example discussed here, only repulsive interactions for the hydronium cation need to be defined. For each repulsive interaction, the name for the interaction type is given follwed by the molecule and atom IDs involved. For the case of the hydronium cation, the name of the interaction to be used is ”Hydronium” and this corresponds to the functional forms used in the MS-EVB3 model.

```
-----------------------------------------------------------------
#ifdef HYDRONIUM
[repulsive.start.H3O]

Hydronium : H3O
-----------------------------------------------------------------
```

The complete list of definitions for the repulsive interactions are listed below.

1. Hydronium:
2. Hydroxide:
3. VII
4. Hydronium_Table: (Not yet in released version)
5. Hydroxide_Table: (Not yet in released version)

The format of the remaining input for this repulsive interaction depends on which definition was used. For the Hydronium interaction, the following can be used to simulate the hydrated proton. Again, the choice of using the MS-EVB2 or MS-EVB3 models determines which set of parameters will be used.

```
-----------------------------------------------------------------
: The parameters
H3O   : EVB_Type of H3O
OW    : Atom_Type of OW
```

```
  #ifdef EVB3
11.2600138    : B
 1.1          : b
 2.12         : b'
 2.4          : d_OO
 4.5715736    : C
 2.1          : c
 1.0          : d_OH

: cutoff for OO
2.85   3.05

: cutoff for OH
2.3    2.5
  #endif

  #ifdef EVB2
14.014       : B
2.20         : b
0.000000     : b'
2.40         : d_OO
7.387        : C
3.60         : c
0.98         : d_OH

: cutoff for OO
2.85   3.05

: cutoff for OH
2.50   3.00
  #endif

[repulsive.end]
#endif   :#ifdef HYDRONIUM

[segment.end]
----------------------------------------------------------------------
```

The functional forms for the repulsive interactions are defined in the papers discussing the MS-EVB3 model.? The repulsive interactions between oxygen atoms of hydronium and water is defined to be

$$V^{rep}_{OO_k} = Be^{-b(R_{OO_k} - d^0_{OO})} \sum_{j=1}^{3} e^{-b' \mathbf{q}^2_{H_j O_k}}, \tag{10}$$

where the proton transfer reaction coordinate is defined to be

$$\mathbf{q}_{H_j O_k} = \frac{1}{2}(\mathbf{r}_O + \mathbf{r}_{O_k}) - \mathbf{r}_{H_j} \tag{11}$$

and the sum over j is over the three hydronium hydrogens. A similar interaction is defined for the repulsion between the hydronium hydrogens and water oxygens.

$$V^{rep}_{HO_k} = Ce^{-c(R_{HO_k} - d^0_{OH})} \tag{12}$$

Both of these repulsive interactions are scaled using a switching function, so that they are zero beyond a specified distance. The piecewise continuous switching function is defined as follows.

$$S(r) = \begin{cases} 1 & r < r_s \\ 1 - \frac{(r-r_s)^2(3r_c - r_s - 2r)}{(r_c - r_s)^3} & r_s < r < r_c \\ 0 & r_c < r \end{cases}$$

This switching function is only piecewise continuous and does not have continuous first derivatives at the boundaries.

5.4.5 evb.top

This file contains molecular topology information specific to MS-EVB simulations. Each line of this file contains information for each atom in the system as two integers. The first integer is the kernel ID for the molecule that the atom belongs to. The second integer is the atom ID within the molecule. Both of these integers are defined with respect to the definitions in the ”in.evb” and ”evb.param” files.

```
-----------------------------------------------------------------
        1          1          1
        2          1          2
        3          1          3
        4          1          1
        5          1          2
        6          1          3
...
      766          2          1
      767          2          2
      768          2          3
      769          2          4
-----------------------------------------------------------------
```

5.4.6 lmp.data

This file was already discussed for regular Lammps simulations and does not need to be changed for MS-EVB simulations.

5.4.7 fes.cfg

This file is used as input when a biasing potential is to be used. The name of this file is defined by the first argument of the “fix_umbrella” directive.

```
fix   1 all evb in.evb evb.out lmp.top
fix   2 all nve
fix   4 all umbrella fes.cfg fes.out
```

Here, we give an example input file for the biasing potential where the center of excess charge (CEC) is tagged to a particular atom within a spherical potential. Notice that the file is divided into three sections.

```
--------------------------------------------------------------------
# Segment 1: Group definition: Group A + Group B

# Format of Group
# First line is the [Type]: 0-coord, 1-CEC, 2-atom, 3-CM, 4-CG
# The next lines:
#   for [coord], coordinates of x, y, and z
#   for [CEC], id of EVB_complex
#   for [atom], tag of atom
#   for [CM], # of atoms, then tag_list of atoms
#   for [CG], # of atoms, then tag_list of atoms

# group 1
1 # Group Type: CEC
0 # complex index

# group 2
2 # Group Type: atom
1 # atom index


# Segment 2: Potential setting

# is spherical
1 # yes

# dimensions
1  1  1
```

```
# k in x, y, z-components
10.0


# reference distance
2.0


# Segment 3: Output
1 # output every such number of step(s), 0 means obey LAMMPS output setting
---------------------------------------------------------------------
```

The first section defines the groups between which the biasing force acts upon. There are five possible group types that can be defined.

0. xyz coordinates
1. CEC of EVB complex
2. Atom Tag
3. Center of Mass (CM) of a group of atoms
4. Center of Gravity (CG) of a group of atoms

For each group type, we then either give a set of xyz coordinates or list of indices to indicate which complex, atom, or group of atoms we want to define the group to be. In this example, the group A is defined to be the CEC coordinate of the first EVB complex. Group B is defined to be the second atom. If instead, we wanted to select the first atom, we would have chosen the atom tag to be zero.

The second section sets up parameters for the biasing potential we would like to apply to the two groups.

The third section specifies details for the printing of related output.

5.5 Output Files

Only those output files unique to MS-EVB simulations with the RAPTOR code will be discussed here.

5.5.1 evb.out

This file contains detailed information regarding the MS-EVB calculation. The frequency of which data is written to this file is controlled by the statements in the ”in.evb” file. This file can get pretty large The number of reactive complexes in the MS-EVB simulation is found on the ”COMPLEX_COUNT” line. For this reactive complex, the molecule ID of the reactive species is shown along with the processor rank to which the molecule is local. Next, the energy decomposition for sub-systems is given along with the total potential energy. This total potential energy is the same as that printed using the ”pe” option for the ”thermo_output” command. The last information for the environment sub-system is list of how the total environmental potential energy is partitioned among the various contributions.

```
--------------------------------------------------------------------
TIMESTEP 0
COMPLEX_COUNT 1
REACTION_CENTER_LOCATION [ center_id | molecule_id | location_rank ]
1   194   2
ENERGY_SUMMARY
ENE_ENVIRONMENT       -2262.785532
ENE_COMPLEX            -412.864330
ENE_TOTAL             -2675.649862
DECOMPOSED_ENV_ENERGY
ENVRIONMENT [vdw|coul|bond|angle|dihedral|improper|kspace]
   577.121 10213.339   254.705   142.169     0.000     0.000 -13450.120
--------------------------------------------------------------------
```

The next section for this reactive complex gives detailed information on each of the MS-EVB states included in the calculation. In this example, a total of 25 states was found by the state search algorithm. The number of states in each reactive shell is also given. Each column of the state search table is defined below.

```
--------------------------------------------------------------------
LOOP_ALL_COMPLEX
START_OF_COMPLEX 1
STATE_SEARCH
COMPLEX 1: 25 state(s) = 1 + 3 + 7 + 14
STATES [ id | parent | shell | mol_A | mol_B | react | path | extra_cpl ]
```

```
                 0      -1       0      -1     194      -1      -1      -1
                 1       0       1     194      32       1       1       0
                 2       0       1     194      35       1       2       0
                 3       0       1     194     115       1       3       0
                 4       1       2      32      45       1       2       0
                 5       1       2      32     248       1       3       0
                 6       2       2      35       1       1       2       0
                 7       2       2      35      83       1       3       0
                 8       2       2      35     235       1       3       1
                 9       3       2     115     193       1       2       0
                10       3       2     115      67       1       3       0
                11       4       3      45      36       1       2       0
                12       4       3      45     157       1       3       0
...
-----------------------------------------------------------------
```

0. id: Running index for state
1. parent: The molecule from which the current state was derived. This number corresponds to the number given in the first column.
2. shell: This number indicates how many steps along a reactive path were taken before the current state was found.
3. mol_A: ID for molecule used to initiate a state search.
4. mol_B: ID for molecule that was found in a state search.
5. react: Index for type of chemical reaction.
6. path: Index for pathway for the chemical reaction.
7. extra_cpl: Number of extra couplings derived from this state.

Given the information in this table, the reactive path used to generate any state may be constructed. For example, state 11 is in the third reaction shell and was derived from state 4. In state 4, the molecule labeled 32 was used to initiate a state search which found the molecule labeled 45 which was subsequently used to initiate a state search which lead to states 11 and 12 being found.

The next section in this output file is the energy decomposition for the nonzero elements of the MS-EVB Hamiltonian matrix. First, information for the diagonal elements is written similar to the decomposition used for the environment energy with the addition of the contribution from the short-ranged repulsive interactions.

```
-------------------------------------------------------------------
DECOMPOSED_COMPLEX_ENERGY
DIAGONAL [id|total|vdw|coul|bond|angle|dihedral|improper|kspace|repulsive]
0  -380.67   68.54  699.27   23.19   13.83   0.0  0.0 -1217.09   31.56
1  -363.02   72.51  689.22   47.53   14.10   0.0  0.0 -1214.42   28.02
2  -347.33   74.89  692.79   55.22   17.91   0.0  0.0 -1209.58   21.41
3  -331.51   75.80  693.41   73.91   18.51   0.0  0.0 -1213.47   20.30
4  -274.25   75.50  679.73  139.58   15.90   0.0  0.0 -1205.95   20.99
...
-------------------------------------------------------------------
```

For the off-diagonal couplings, the three contributions to the energy are given: the geometric factor, $A(R,q)$, the off-diagonal constant, V^c_{ij}, and the total unscaled energy, V_{ij}. The total off-diagonal energy is the unscaled energy multipled by the geometric factor. The contribution from the electrostatic interactions can be calculated by subtracting the off-diagonal constant from the unscaled energy. If any off-diagonal couplings were found in the state search, then the total off-diagonal energy for them are also given along with the state IDs involved in the coupling.

```
-------------------------------------------------------------------
OFF-DIAGONAL [ id | energy |  A_Rq  | Vij_const |   Vij   ]
         1   -25.624288     0.994503   -23.187187   -25.765933
         2   -23.655135     0.941026   -23.187187   -25.137592
         3   -23.641459     0.937413   -23.187187   -25.219885
         4   -23.422207     0.830797   -23.187187   -28.192464
         5   -25.423939     0.934829   -23.187187   -27.196361
...
EXTRA-COUPLING 1 [ I | J | energy ]
                 7    8   -0.086801
-------------------------------------------------------------------
```

The final section for the reactive complex contains information regarding the ground state solution to the MS-EVB Hamiltonian and the coordinates for the CEC.

```
-------------------------------------------------------------------
DIAGONALIZATION
EIGEN_VECTOR
0.7766 0.4858 0.2977 0.2329 0.0836 0.0927 0.0422 0.0133 0.0021 0.0186 ...
NEXT_PIVOT_STATE 0
CEC_COORDINATE
4.352837 6.579997 -5.625293
END_OF_COMPLEX 1
-------------------------------------------------------------------
```

If a multiple reactive complex simulation was run using the SCI-MS-EVB methodology, then corresponding sections for each of the reactive complexes would be written to file.

5.5.2 fes.out

5.5.3 in.evb-raptor.out

This file shows output from the step-by-step processing of the in.evb file. This filename will be generated by appending ”-raptor.out” to the end of the configuration file name, in this case ”in.evb.” If the code were to crash sometime during the processing of the ”in.evb” file, hopefully the contents of this file will be useful in identifying the location of the error in the file and the possible cause for the issue (e.g. undefined macro). In some instances, where possible, warning messages will be printed to this file to indicate possible issues. One example for this warning is the use of an undefined macro for a potential type (bond, angle, etc...), which would cause the potential type to be zero. The calculation will proceed as normal even if warning messages are printed.

5.6 Special Issues for Complex Systems

In the previous sections, the input file format has been discussed using the example of an excess proton in pure water. This example does not illustrate a number of details of the input format that are important for more complicated systems—e.g., a solvated protein with reactive centers. In this section, these details are described using a rather complicated example system (human carbonic anhydrase II). The order of this section roughly follows that of section 4.2. Note that there are no additional concerns for lmp.in.

Concerning the example system: Human carbonic anhydrase II (HCA II) is a 261-amino acid protein which incorporates a Zn ion with a strongly bound water or hydroxide ligand. Here, the protein is defined to be initially in the water-bound state; including the Zn and bound H_2O, there are 4090 atoms in the protein. Two reactive centers are defined: the ZnH_2O group with its three His ligands (His94, His96, and His119), and His64 (which has reactive sites at both N atoms). To model this system, it is necessary to consider seven different reactive “molecules”: $(His)_3ZnH_2O$, $(His)_3ZnOH$, doubly protonated His64, neutral His64 protonated at N^{δ}, neutral His64 protonated at N^{ϵ}, H_3O, and H_2O. This example uses the CHARMM36 force field.

5.6.1 lmp.in and lmp.data

The formats of these files are the same as described before. However, a few points deserve emphasis:

1. It is imperative that either lmp.in or lmp.data include the necessary parameters (mass, LJ, charge, bond, angle, dihedral, etc.) for all possible EVB states, even if those interactions are

not utilized in lmp.data.

2. The interactions in lmp.data should be defined appropriately for the state that will be the pivot state for the first time step of the simulation. All parameters should be specified, not merely the parameters for the non-reactive part of the system.

3. For bonded parameters corresponding to the reactive part of the system, the order in which the atomic indices are specified should match the order used in the EVB molecule definitions. Although failure to do so will not cause an error in all cases, best practices dictate that the orders should be the same. The reason for this is described below.

4. Each EVB complex must have its own molecule number in the "Atoms" section of lmp.data. For example, if the system contains 1 protein, including a reactive histidine, and 10,000 water molecules, the non-reactive part of the protein could be set to molecule number 1, the water molecules to 2–10001, and the reactive histidine to 10002.

5.6.2 in.evb

This section is essentially the same as described before, except that more parameters must now be defined (including dihedral and improper parameters). To determine which parameters must be included in in.evb, it is first necessary to introduce the concept of interaction "ownership" by individual atoms, as well as the rules for determining which atoms to include in the EVB molecule definition, and whether to set the value of "evb_bond" (sometimes also called "kernel") to 0 or 1.

An atom is said to *own* an interaction when it is the first atom listed in the definition of a bond potential, or the second atom listed in the definition of an angle, dihedral, or improper potential. For example, consider the following input (from evb.par_4charmm):

```
[molecule_type.start.HIP]

: number of atoms,   bonds,   angles,  dihedrals,   impropers,   starting rc
             18         16         27          49           4            1

: atomic information
: atom type   charge    evb_bond
     NH1      -0.47     0          : 1
     H         0.31     0          : 2
     CT1       0.07     1          : 3
     HB1       0.09     0          : 4
     CT2A     -0.05     1          : 5
...

: bonds
:  atom 1  atom 2  type
    3       1        CT1-NH1
...

: angles
:  atom 1  atoms 2  atom 3  type
    1       3        4        NH1-CT1-HB1
...

: dihedrals
:  atom 1  atom 2  atom 3  atom 4  type
    5       3       1       2        CT2A-CT1-NH1-H
...
```

As defined, all the bonded potentials listed are owned by atom 3 (CT1). Reversing the order in which the atoms are listed would not change the parameters of the potentials, but would cause the bond and dihedral interactions to be owned by atom 1 (NH1). This change is potentially quite important.

The following rules can be used to determine which atoms must be included in the molecule definition, and the appropriate value of evb_bond for each:

1. Do the atom's type or parameters (bonded or non-bonded) change when a reaction occurs? If so, it must be included in the EVB molecule definition. It may also be helpful to include additional atoms in the EVB molecule definition so as to make the EVB molecule neutral, to include every atom in a protein residue, etc.

2. Does the atom's type change when a reaction occurs, does it appear/disappear when a reaction occurs, or does it own any bonded interactions which change when a reaction occurs? If so, evb_bond should be set to 1 for that atom. Otherwise, evb_bond can be set to 0.

Atomic information should be specified for all atoms in the EVB molecule. However, bonded potentials need only be defined for interactions which change (i.e., are modified, gained, or lost) when a reaction occurs. All such interactions must be owned by an atom for which evb_bond = 1, or RAPTOR will not modify the interaction appropriately. As a matter of best practice, it is typically best to list all interactions which include an atom for which the atom type or any bonded potential changes. This makes the EVB input more explicit and easier to modify. (Note that this may require setting evb_bond to 1 for more atoms than necessitated by the rules above. For instance, in the example, the atom type for atom 5 (CT2A) changes when a reaction occurs, dictating that evb_bond be set to 1 for it. Meanwhile, it might be possible to set evb_bond to 0 for atom 3 if none of its interactions with atom 5 changed. However, the 5-3-1-2 dihedral interaction cannot be owned by atom 5, making it necessary to set evb_bond to 1 for atom 3 (or atom 1) if this parameter is to be included.)

Besides these concerns about which interactions need to be defined in in.evb, the setup of in.evb is relatively straightforward. The only other important point to mention is that the number of the type assigned to a given atom or bonded interaction **must** be the same as the number assigned to that atom or interaction in lmp.data; it is not acceptable to substitute a different type number, even if the associated parameters are identical. This is because RAPTOR relies on correspondence between the type indices in in.evb and lmp.data in order to change the potential when a reaction occurs.

Below, the "Declare Modules" and "Declare LAMMPS Types" sections for HCA II are shown (in abbreviated form) as an example. Note that hydronium oxygen and hydrogen are defined as OY and HY in this section, unlike the previous example.

```
::::::::::::::::::::
:: Declare Modules
::::::::::::::::::::

::: WATER MODEL
#define EVB3

#define WAT
settype H2O 1
settype H3O 2

#define WZN
settype ZNW 3
settype ZNX 4

#define HIS
settype HIP 5
settype HIE 6
settype HID 7

::: Reaction Types
#define WAT_WAT
#define ZNW_WAT
#define HYD_ZNX
#define HIPd_WAT
#define HIPe_WAT
#define HYD_HIE
#define HYD_HID

::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

```
::::::::::::::::::::::::
:: Declare LAMMPS Types
::::::::::::::::::::::::

: --------- WATER ---------
#ifdef WAT

: atom type

#define OW   42
#define HW   43
#define OY   46
#define HY   47

: bond type

#define OW-HW  76
#define OY-HY  79

: angle type

#define HW-OW-HW   174
#define HY-OY-HY   177

#endif

: ---------- WZN ----------
#ifdef WZN

: atom type

#define zNH1   5
#define zH     6
#define zCT1   7
#define zHB1   8
#define zCT2   9
#define zHA2   10
#define zNR1   12
#define zCPH1  13
#define zCPH2  14
#define zHR1   15
#define zNR2   16
```

```
#define zHR3   17
#define zC     3
#define zO     4
#define Zn     39
#define OZ     40
#define HZ     41
#define OD     44
#define HD     45

: bond type

#define zNR2-zCPH2  19
...
#define OD-HD       78

: angle type

#define zCPH2-zNR2-zCPH1  33
...
#define Zn-OD-HD          176

: dihedral type

#define zCPH1-zNR2-zCPH2-zNR1   64
...
#define zNR2-Zn-OD-HD           418

: improper type

#define zCPH2-zNR2-zNR1-zHR1   7
...
#define zHR3-zNR2-zCPH1-zCPH1  8

#endif

: ---------- HIS ----------
#ifdef HIS

: atom type

#define NH1   5
#define H     6
```

```
#define CT1   7
#define HB1   8
#define CT2A  33
#define HA2   10
#define CPH1  13
#define HR1   15
#define NR3   48
#define CPH2  14
#define HR2   49
#define C     3
#define O     4
#define CT2   9
#define NR2   16
#define NR1   12
#define HR3   17

: bond type

#define CT1-NH1    6
...
#define CPH1-HR3   21

: angle type

#define NH1-CT1-HB1     9
...
#define NR2-CPH1-HR3    36

: dihedral type

#define CT2A-CT1-NH1-H       234
...
#define CPH2-NR2-CPH1-HR3    67

: improper type
#define NR3-CPH1-CPH2-H     27
...
#define HR3-CPH1-NR2-CPH1   9

#endif

::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

```
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

#include "evb.par_4charmm"

::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

5.6.3 evb.par_4charmm

1: Definitions for the EVB kernels

This section must include atomic information for all atoms in each EVB molecule, as well as information on bonded potentials for all interactions that are gained, lost, or modified by a reaction. For clarity and ease of modification, it is also useful to include all potentials including any atom for which evb_bond = 1. (It may be that not every such potential can be defined using only atoms from the EVB molecule; in this case, it is fine to ignore the potential if the user is certain that it will not be changed or lost following a reaction.) To determine how to set evb_bond, see the rules in the previous section. In brief, if a given atom changes its type or can disappear following a reaction, or if it is the first atom listed in any bond potential or the second atom listed in any angle, dihedral, or improper potential, it is necessary to set evb_bond = 1 for that atom. In some cases, it may be possible to avoid needing to set evb_bond = 1 for an atom by reversing the order of the indices used to define a potential. Remember, however, that the order of the atoms in this file should match the order in lmp.data.

Below, the molecule definition for doubly protonated His is shown as an example. For this molecule, the atoms which may disappear or change their atom types following a reaction are atoms 5 (CT2A), 9 (HR1), 11 (NR3), 12 (H), 13 (NR3), 14 (H), and 16 (HR2); evb_bond is therefore set to 1 for these atoms. In addition, evb_bond is also set to 1 for atoms 3 (CT1), 8 (CPH1), 10 (CPH1), and 15 (CPH2) because otherwise it is impossible to define all the interactions for the first set of molecules such that an atom with evb_bond = 1 owns the interaction. (For instance, the 10-8-9 angle can only be owned by atom 8.)

```
[molecule_type.start.HIP]

: number of atoms,   bonds,   angles,  dihedrals,   impropers,   starting rc
             18        16        27           49           4           1

: atomic information
: atom type   charge    evb_bond
     NH1      -0.47     0          : 1
     H         0.31     0          : 2
     CT1       0.07     1          : 3
     HB1       0.09     0          : 4
     CT2A     -0.05     1         : 5
     HA2       0.09     0         : 6
     HA2       0.09     0         : 7
     CPH1      0.19     1         : 8
     HR1       0.13     1         : 9
     CPH1      0.19     1         : 10
     NR3      -0.51     1         : 11
     H         0.44     1         : 12
     NR3      -0.51     1         : 13
     H         0.44     1         : 14
     CPH2      0.32     1         : 15
     HR2       0.18     1         : 16
     C         0.51     0         : 17
     O        -0.51     0         : 18

: bonds
:  atom 1  atom 2  type
    3       1        CT1-NH1
    3       4        CT1-HB1
    3       5        CT1-CT2A
    3       17       CT1-C
    5       6        CT2A-HA2
    5       7        CT2A-HA2
    5       10       CT2A-CPH1
    8       9        CPH1-HR1
    8       10       CPH1-CPH1
    8       11       CPH1-NR3
    10      13       CPH1-NR3
    11      12       NR3-H
    11      15       NR3-CPH2
    13      14       NR3-H
    13      15       NR3-CPH2
    15      16       CPH2-HR2

: angles
:  atom 1  atoms 2  atom 3  type
    1       3        4        NH1-CT1-HB1
    1       3        5        NH1-CT1-CT2A
    1       3        17       NH1-CT1-C
    4       3        5        HB1-CT1-CT2A
    4       3        17       HB1-CT1-C
    5       3        17       CT2A-CT1-C
    3       5       6       CT1-CT2A-HA2
    3       5       7       CT1-CT2A-HA2
    3       5       10      CT1-CT2A-CPH1
    6       5       7       HA2-CT2A-HA2
    6       5       10      HA2-CT2A-CPH1
    7       5       10      HA2-CT2A-CPH1
    5       10      8       CT2A-CPH1-CPH1
    5       10      13      CT2A-CPH1-NR3
    8       10      13      CPH1-CPH1-NR3
    10      13      14      CPH1-NR3-H
    10      13      15      CPH1-NR3-CPH2
    15      13      14      CPH2-NR3-H
    11      15      16      NR3-CPH2-HR2
    11      15      13      NR3-CPH2-NR3
    13      15      16      NR3-CPH2-HR2
    8       11      12      CPH1-NR3-H
    8       11      15      CPH1-NR3-CPH2
    15      11      12      CPH2-NR3-H
    10      8       9       CPH1-CPH1-HR1
    11      8       9       NR3-CPH1-HR1
    10      8       11      CPH1-CPH1-NR3

: dihedrals
:  atom 1  atom 2  atom 3  atom 4  type
    5       3       1       2         CT2A-CT1-NH1-H
    5       3       17      18        CT2A-CT1-C-O
    1       3       17      18        NH1-CT1-C-O
    1       3       5       6         NH1-CT1-CT2A-HA2
    1       3       5       7         NH1-CT1-CT2A-HA2
    1       3       5       10        NH1-CT1-CT2A-CPH1a
    1       3       5       10        NH1-CT1-CT2A-CPH1b
    1       3       5       10        NH1-CT1-CT2A-CPH1c
    4       3       1       2         HB1-CT1-NH1-H
    4       3       17      18        HB1-CT1-C-O
    4       3       5       6         HB1-CT1-CT2A-HA2
    4       3       5       7         HB1-CT1-CT2A-HA2
    4       3       5       10        HB1-CT1-CT2A-CPH1
    17      3       1       2         C-CT1-NH1-H
    17      3       5       6         C-CT1-CT2A-HA2
    17      3       5       7         C-CT1-CT2A-HA2
    17      3       5       10        C-CT1-CT2A-CPH1a
    17      3       5       10        C-CT1-CT2A-CPH1b
    17      3       5       10       C-CT1-CT2A-CPH1c
    3       5       10      8        CT1-CT2A-CPH1-CPH1a
    3       5       10      8        CT1-CT2A-CPH1-CPH1b
    3       5       10      8        CT1-CT2A-CPH1-CPH1c
    3       5       10      13       CT1-CT2A-CPH1-NR3a
    3       5       10      13       CT1-CT2A-CPH1-NR3b
    3       5       10      13       CT1-CT2A-CPH1-NR3c
    6       5       10      8        HA2-CT2A-CPH1-CPH1
    6       5       10      13       HA2-CT2A-CPH1-NR3
    7       5       10      8        HA2-CT2A-CPH1-CPH1
    7       5       10      13       HA2-CT2A-CPH1-NR3
    5       10      8       9        CT2A-CPH1-CPH1-HR1
    5       10      8       11       CT2A-CPH1-CPH1-NR3
    5       10      13      14       CT2A-CPH1-NR3-H
    5       10      13      15       CT2A-CPH1-NR3-CPH2
    8       10      13      14       CPH1-CPH1-NR3-H
    8       10      13      15       CPH1-CPH1-NR3-CPH2
    13      10      8       9        NR3-CPH1-CPH1-HR1
    13      10      8       11       NR3-CPH1-CPH1-NR3
    10      13      15      16       CPH1-NR3-CPH2-HR2
    10      13      15      11       CPH1-NR3-CPH2-NR3
    16      15      13      14       HR2-CPH2-NR3-H
    11      15      13      14       NR3-CPH2-NR3-H
    13      15      11      12       NR3-CPH2-NR3-H
    8       11      15      13       CPH1-NR3-CPH2-NR3
    16      15      11      12       HR2-CPH2-NR3-H
    8       11      15      16       CPH1-NR3-CPH2-HR2
    9       8       11      15       HR1-CPH1-NR3-CPH2
    10      8       11      15       CPH1-CPH1-NR3-CPH2
    9       8       11      12       HR1-CPH1-NR3-H
    10      8       11      12       CPH1-CPH1-NR3-H

: impropers
:  atom 1  atom 2  atom 3  atom 4  type
    13      10      15      14       NR3-CPH1-CPH2-H
    14      10      15      13       H-CPH1-CPH2-NR3
    11      8       15      12       NR3-CPH1-CPH2-H
    12      8       15      11       H-CPH1-CPH2-NR3

: COC information
4 : number of COC atoms
11 12 13 14 : index of COC atoms
```

```
[molecule_type.end]
```

2: Definitions for changing molecular topologies during chemical reaction

There are no particularly unique issues that arise beyond what was covered for the excess proton in water.

3: Definitions for state search

Two points should be mentioned. First, this section must be defined for every molecule_type, even if that molecule cannot be used to initiate a state search. In that case, simply define a blank state search segment (i.e., place the [state_search.end] command immediately after the [state_search.start.XXX] command). Second, if any molecule possesses more than one reactive group (e.g., each H in H_3O or ZnH_2O), rules must be provided for continuing the state search from the initially non-reactive form (e.g., H_2O or ZnOH). The setup is entirely analagous to that provided for H_2O earlier, but an example for ZnH_2O/ZnOH is nevertheless provided below.

```
[state_search.start.ZNW]

  #ifdef ZNW_WAT
  :H->Ow
    54     H2O          1             3                 2.5        ZNW_H2O         1
    55     H2O          1             3                 2.5        ZNW_H2O         2
  #endif

[state_search.end]

[state_search.start.ZNX]

  #ifdef ZNW_WAT
  :H->Ow
    54     H2O          1             3                 2.5        ZNW_H2O         2
  #endif

[state_search.end]
```

4: Off-diagonal couplings

There is one important point to mention concerning the use of the "PT" coupling type when $A(R, q)$ is of the asymmetric type: The reaction coordinate q depends on the order in which the EVB indices of the donor and acceptor atoms are provided. $A(r, q)$ should be the same regardless of whether the transfer is specified as DH + A → D + AH or as D + AH → DH + A. Thus, evb_index for molecule D (the "donor") should always be provided before evb_index for molecule A (the "acceptor"), **even**

if molecule A acts as the donor in the reaction at hand. For proton transfer from a non-water molecule to water, typically the non-water molecule is defined as D and the water molecule is defined as A. An example is provided below for proton transfer from ZnH_2O (ZNW) to H_2O and from H_3O to ZnOH (ZNX).

```
[off_diagonal.start.ZNW_H2O]

  PT  : Type of coupling: Proton Transfer
  1 53 : evb_index of DONOR atom (should be in molecule A)
  2 1  : evb_index of ACCEPTOR atom (should be in molecule B)
  2 2  : evb_index of transferred HYDROGEN atom (should be in molecule B)

  : A_Rq Type : A(R,q) = f(R) * g(q)
  2 : 1-symmetric; 2-asymmetric

  : parameters for ZnH2O
...

[off_diagonal.end]

[off_diagonal.start.H3O_ZNX]
  PT  : Type of coupling: Proton Transfer
  2 53 : evb_index of DONOR atom (should be in molecule A)       Note the first two in
  1 1  : evb_index of ACCEPTOR atom (should be in molecule B)    case so that Vex is t
  2 54 : evb_index of transferred HYDROGEN atom (should be in molecule B)

  : A_Rq Type : A(R,q) = f(R) * g(q)
  2 : 1-symmetric; 2-asymmetric

  : parameters for ZnH2O
...

[off_diagonal.end]
```

As a final point, note that the atom indices used to evaluate the off-diagonal couplings all correspond to the **post-reaction** complex. Imagine, for instance, that the evb_index of the DONOR atom in the above example (the oxygen of the ZnH_2O/ZnOH complex) was 53 in ZNW (ZNH_2O), but 54 in ZNX (ZnOH). In that case, the DONOR atom for the first reaction above (ZNW + H2O $\rightarrow$ ZNX + H3O) should be 1 54, corresponding to the index in the product complex. For the H3O_ZNX reaction, the index would still be 2 53.

5: Short-ranged repulsive interactions

There are no particularly unique issues that arise beyond what was covered for the excess proton in water.

5.6.4 evb.top

The only additional point to mention is that atoms that do not belong to any EVB molecule_type should be specified with kernel ID 0. The atom index for such atoms is of no consequence.

5.7 Supported add-on packages

5.7.1 COLVARS

The current version of RAPTOR supports use of the COLVARS library. The COLVARS package can be installed as follows.

```
cd lammps/lib/colvars
make -f Makefile.g++
cd ../../src/
make yes-USER-COLVARS
make yes-USER-RAPTOR
make -j 4 mac
```

Reinstalling the RAPTOR package after the COLVARS package will cause the RAPTOR-edited source files for the COLVARS package to be added to the src directory. The CEC's are treated by COLVARS as extra particles. If the system being studied has 1000 particles, then the CEC coordinate will be particle 1001 and the CEC_v2 coordinate will be particle 1002. In an SCI simulation with two complexes, the CEC coodinates for the second complex will be 1003 and 1004 and so on for additional complexes. Note, if coupling CECs in an SCI simulation it is important to use a recent version of RAPTOR so that the order of complexes does not change over the course of a simulation. At present however, the order is not preserved between separate simulations and will likely change.

5.7.2 PLUMED

Packages for RAPTOR support of PLUMED versions 1.2 and 2.0 is currently available on the wiki. The PLUMED+LAMMPS installation instructions can be followed. The edited source files in these packages can then be included in the src directory to compile RAPTOR with PLUMED support. The convention for CEC particle IDs is the same as for the COLVARS package.

6 SCI Simulations

While most of the algorithms implemented for simulations with a single reactive species also support simulations with multiple reactive species using the SCI algorithm, there are a few key differences that should be noted and they are discussed here.

6.1 Electrostatics

Full support for Ewald is implemented for SCI simulations with exact energies/forces for diagonal and off-diagonal matrix elements.

Support for PPPM is available in SCI simulations for the diagonal matrix elements only. With SCI and PPPM, the off-diagonal matrix elements currently need to be evaluated with a real-space approximation for the electrostatics, using methods discussed in section 8.7.[?] For the diagonal matrix elements, the k-space energies are exact. The foces can be calculated using the Hellmann-Feynman theorem or via an approximae method using a set of effective charges. The default behavior is to use HF forces for all quantities. This can be changed by passing ”hf” or ”acc” to the ”sci_pppm” keyword for the evb fix.

```
# -- Default behavior
fix 1 all evb in.evb evb.out lmp.top sci_pppm=hf

# -- Approximate PPPM forces; old default behavior
fix 1 all evb in.evb evb.out lmp.top sci_pppm=acc

# -- Approximate PPPM + polarization forces
fix 1 all evb in.evb evb.out lmp.top sci_pppm=polar
```

The PPPM energies and forces calculated using the ”hf” method should match those calculated using Ewald to a few decimal places when both calculations are appropriately converged. The ”polar” option is another approximate method to further reduce the number of FFT operations and overall cost of simulations. This is still a work in progress and not currently supported. For the ”acc” method, the effective charges used are those determined after the ground state eigenvector has been converged for each reactive complex, similar to the ELRI approximations discussed in section 8.7. A typical maximum deviation in any cartesian component of an atomic force is on the order of 1-2 kcal/mol per Å when compared to forces calculated using Ewald (Figure.6).

If one is interested in optimizing for computational performance, PPPM is the superior option compared to Ewald. The ”hf” calculation of forces is more expensive than the ”acc” method, however, the multipro parallelization does a good job of hiding most of the increased computational expense. For the approximate methods, it is the responsibility of the user to ensure that the approximations made in the PPPM implementation do not significantly alter the results of their simulations, which can be checked against calculations using Ewald.

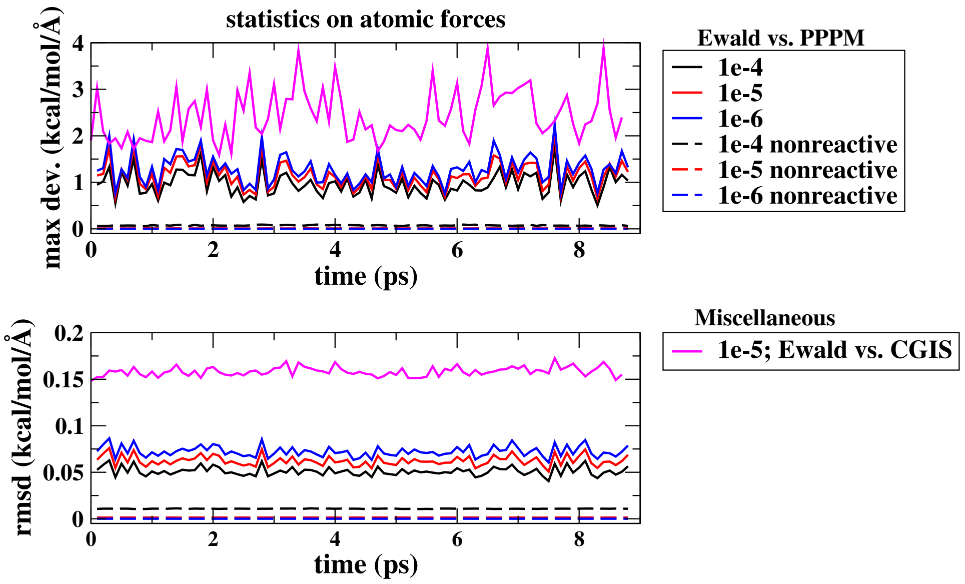

Figure 6: A comparison of a) the maximum deviation between components of atomic forces and b) the root-mean-squared-deviation of all atomic forces calculated with PPPM and Ewald over the course of a 9 ps trajectory of a fuel cell system with 40 reactive protons and a total of 3056 atoms. The solid lines show results for reactive calculations, while the dashed lines give the corresponding errors for nonreactive calculations for reference. Results are shown for three separate values of the relative error in the forces (LAMMPS input parameter). The magenta curve labeled "1e-5; Ewald vs. CGIS" is a point of reference between two methods that both use exact forces for all matrix elements. CGIS was used as the real-space approximation in the SCI PPPM calculations.

The real-space Coulomb interactions calculated in an SCI simulation does not support pre-computing energies and forces stored in a table for terms related to the SCI decomposition. Instead, the energies and forces are computed directly. As such, small differences may arrise when comparing results with non-SCI simulations that may use tabulated energies and forces. Adding the "table 0" option to "pair_modify" in the non-SCI calculation should eliminate these differences.

6.2 SCI convergence

The default convergence criteria is that the energies for all complexes converge to within 1e-6 of the energies from the previous iteration. After a maximum number of iterations have been computed, the last solution obtained is used for the rest of the calculation. The parameters controlling the tolerance and maximum number of iterations are currently hard coded near the top of the EVB_engine_sci.cpp and EVB_engine_sci_mp.cpp source files.

One possible cause for slow SCI convergence identified thus far is the combination of a tight energy criteria and single-precision FFTs for PPPM. For the case of electrode simulations, single-precision FFTs can be problematic to converge energies to within 1e-6. The issue is elliviated with the use of double-precision FFTs. This convergence issue has not yet been observed with SCI simulations of polyelectrolyte membranes.

A second option for convergence criteria is to use the overlap of sequential eigenvectors. With energies, one must compute differences to determine is convergence is satisfied, thus with an energy criteria a minimum of three iterations is necessary. A minimum of two iterations can achieve convergence using eigenvector overlaps, which can lead to a noticeable performance improvement in some cases. The use of eigenvector overlap can be activated by adding the ”sci_overlap” option to the evb fix. The tolerance can be controlled as an argument to the sci_overlap option and the default value is 0.999.

```
fix 1 all evb in.evb evb.out lmp.top sci_overlap
fix 1 all evb in.evb evb.out lmp.top sci_overlap=0.999
```

6.3 I/O

When running with multiple partitions, one will want to be mindful of which partitions are actually involved in the writing of output files. In general, only output from the master partition will be useful. Using the following command line and input script as an example, one can ensure only the master partition will write output. Limiting output to the master partition also improves performance and prevents multiple slave partitions from overwriting the output file generated by the master partition.

The last two options in this command line will turn off the writing of screen and log files for each of the 2048 partitions. In this example, the creation and writing to disk of 8192 separate files is eliminated.

```
mpirun -np 16384 lmp_titan -in lmp.in -p 4096x4 -pscreen none -plog none
```

The writing of a log file can be re-activated using commands in the LAMMPS input script, like below. Also notice the use of ”partition yes 1” commands to limit the writing of output to the master partition only.

```
units real
atom_style full

partition yes 1 log log.lammps.0

pair_style lj/cut/coul/long 9.0
pair_modify     mix arithmetic

bond_style      hybrid harmonic morse
```

```
angle_style     harmonic
dihedral_style  charmm
improper_style  none
kspace_style    pppm  1e-4

multipro_sci    read_data       lmp.data

include         lmp.param

neighbor 2.0 bin

thermo          100
thermo_style    custom step temp pe etotal

fix             evb all evb evb.in evb.out lmp.top
fix             1 all nvt temp 300 300 500

partition yes 1 dump  1 all custom 1000 lmp.lammpstrj.bin id type mol x y z

partition yes 1 restart         1000 run1.restart run2.restart

timestep        1.0
run             1000
```

6.4 GPU acceleration

GPU support is not yet available in SCI simulations, but is currently in the works.

6.5 Intel MIC acceleration

Intel MIC support is not yet available in SCI simulations, but is currently in the works.

7 Electrified Interface Simulations

This section discusses how to simulate polarizable planar electrodes using a computationally efficient image charge method.[?]

8 Improving Performance

This section discusses several routes to improve parallel scaling on high-performance computing resources. When first starting the effort to improve performance of your simulations, the general order of strategies to try is as follows.

1. OpenMP
2. OpenMP + state decomposition: This is only implemented for simulations with a single reactive species and at present is the most successful at scaling simulations out to 8 racks on Mira (IBM BG/Q) (>500K cores).
3. OpenMP + split the electrostatic energy/force calculation across separate partitions.
4. If GPU accelerators are available, LAMMPS and RAPTOR can be compiled with the GPU package in combination with the above algorithms.

The parallelization strategies mentioned thus far do not alter the accuracy of the calculations. To further reduce the computational cost of the electrostatic calculations, approximate algorithms can be used in conjunction with any of the above mentioned parallelization strategies.

For the case of simulations with multiple reactive species via the SCI algorithm, OpenMP and a complex decomposition strategy can be employed to improve the parallel efficiency.

A Hamiltonian screening algorithm is available that uses bonded interactions and geometric factors to construct a computationally cheap Hamiltonian matrix. The ground state eigenvector from this matrix is used to identify those diabatic states with weight larger than a user-specified tolerance. This subset of states is retained and used in the full calculation of energies and forces. Support is currently available for single- and multi-complex simulations using single-partition parallelization algorithms. Support is available for state-decomposition of a single complex using the ”mp_state” keyword. Support for state- and complex-decomposition parallelization algorithms via the ”multi-pro_sci” algorithm is in progress.

8.1 OpenMP

The OpenMP version of RAPTOR can be compiled just as the OpenMP version of LAMMPS. One first installs the USER-OMP package and adds the appropriate flags to the Makefile. It be necessary to update the ”EVB_cracker.h” after installing the USER-OMP package. The following set of commands (or similar) should accomplish this task.

```
cd lammps/src
make yes-USER-OMP
rm EVB_cracker.h USER-RAPTOR/EVB_cracker.h
```

```
make yes-USER-RAPTOR
make makena-OMP
```

The ”make yes-USER-RAPTOR” command will generate a new ”EVB_cracker.h” file that is appropriate for compilation with the USER-OMP package. In the Makefile, one only needs to add the following to the ”CCFLAGS” variable depending on compiler.

```
CCFLAGS = -fopenmp -DLMP_USER_OMP ...        #GNU
CCFLAGS = -openmp -DLMP_USER_OMP ...         #INTEL
CCFLAGS = -qsmp=omp -DLMP_USER_OMP ...       #XLM
```

Adding the appropriate OpenMP flag to the ”LINKSFLAG” variable may also be necessary. One can always check the documentation for instructions on a specific HPC platform. To run the code, one first sets the number of OpenMP threads to be assigned to each MPI rank and then adds the ”-sf” flag to activate OpenMP-optimized algorithms.

```
setenv OMP_NUM_THREADS 4
mpirun -np 8 ./lmp_makena-OMP -sf omp -in lmp.in -out lmp.out
```

On Cray machines that utilize the ”aprun” command to submit jobs, something like the following can be used to submit OpenMP-enabled jobs.

```
setenv OMP_NUM_THREADS 2
aprun -n 16 -N 8 -d 2 ./lmp_garnet -sf omp -in lmp.in -out lmp.out
```

On this machine, each node has 16 cores and this would be used in a 2-node job submission for 32 cores total (16 MPI ranks with 2 OpenMP threads per rank). This command would submit a 2 node job with 8 MPI ranks per node and 2 OpenMP threads per rank.

At larger processor counts, using 4 OpenMP threads per rank tends to perform the best, but this will depend on specifics of the computing platform.

8.2 State Decomposition for Single Reactive Species

The evaluation of a matrix element in the MS-EVB Hamiltonian can be done completely independent of the evaluation of any other matrix element, thus a very useful parallelization strategy is to simultaneously evaluate matrix elements. This parallelization strategy can be activated by compiling the RAPTOR code with the preprocessor flag ”-DSTATE_DECOMP”. In the input file, the command that activates the evb fix is appended with ”mp_state” and an integer indicating the algorithm to be used.[?]

```
fix             evb all evb in.evb evb.out ${input}.top mp_state 3
```

The currently available options 1, 2, 3, 4 are discussed below in more detail.

1. Basic option. States are divvied out as evenly as possible among all partitions and all partitions reduce all forces. Requires reducing N_state * N_atom * 3 size force arrays.

2. Same as 1, except that partition 0 is restricted to only handle the real-space energy/force calculation for the environment. The remaining partitions evaluate energies and forces for all of the different states.

3. Improved communication strategy for option 1. Energies are reduced universally in order to diagonalize the Hamiltonian. Each partition sums its contribution to the ground state forces based on the coefficients from the diagonalized Hamiltonian. Then, the partitions reduce the ground state force only, which only requres reducing a N_atom * 3 size force array leading to an N_state reduction in communication size relative to the default strategy.

4. Improved communication strategy for option 2.

Option 3 is the only one currently enabled for users given that its performance is best.

To submit such a job, one needs to only specify an option on the command line to control the number and size of the partitions. Currently, all partitions are expected to be of the same size and shape. The following would submit a job that utilizes 4 separate partitions, each with 16 MPI ranks.

```
mpirun -np 64 ./lmp.x -in lmp.in -p 4x16
```

For the OpenMP-enabled code, something like the following could be used.

```
env OMP_NUM_THREADS=2 mpirun -np 32 ./lmp.x -sf omp -in lmp.in -p 4x8
env OMP_NUM_THREADS=4 mpirun -np 16 ./lmp.x -sf omp -in lmp.in -p 4x4
env OMP_NUM_THREADS=8 mpirun -np 8  ./lmp.x -sf omp -in lmp.in -p 4x2
```

8.3 Multiple Program Electrostatic Algorithm

In this parallelization strategy, the k-space calculation for electrostatic interactions is evaluated on a separate partition of processors, while the real-space interactions are evaluated on the master partition. This strategy is available for both nonreactive simulations with LAMMPS and reactive simulations with RAPTOR. For LAMMPS, one needs to install the ”REPLICA” package in order to use the verlet/split run_style.

```
make yes-REPLICA
make makena
```

In the LAMMPS input file, the run_style can be specified as follows.

```
run_style  verlet/split
```

Simulations with this parallelization strategy can be run using commands like the following.

```
mpirun -np 16 ./lmp.x -reorder nth 2 -p 8 8 -in lmp.in
mpirun -np 16 ./lmp.x -reorder nth 4 -p 12 4 -in lmp.in
mpirun -np 16 ./lmp.x -reorder nth 8 -p 14 2 -in lmp.in
```

Each of these commands submits a job utilizing 16 MPI ranks, but differ in the relative size of the partitions. The partitions have ratios of 1:1, 3:1 and 7:1 respectivly. The first partition is responsible for real-space interactions, while the second partition, possible smaller than the first, is responsible for FFT-related calculations involving Alltoall communication. The ”-reorder” commands rearrange the MPI ranks so that processors that will share similar local domains of the simulation cell or assigned to cores that are physically close to one another on the computer. More details on the implementation can be found in the original paper.[?]

This strategy can also be used with OpenMP-enabled algorithms. Submitting the same 2-node jobs, but now with 2 OpenMP threads per rank would look like the following.

```
setenv OMP_NUM_THREADS 2
mpirun -np 8 ./lmp.x -reorder nth 2 -p 4 4 -sf omp -in lmp.in
mpirun -np 8 ./lmp.x -reorder nth 4 -p 6 2 -sf omp -in lmp.in
mpirun -np 8 ./lmp.x -reorder nth 8 -p 7 1 -sf omp -in lmp.in
```

The total number of cores being used remains the same, but the number of MPI ranks, which define the partitions, is halved in this case. Ratios of 1:1 and 3:1 tend to perform the best with 3:1 outperforming 1:1 at the highest processor counts. Ratios greater than 3:1 can be attempted, but there will usually be no performance gain or the code will exit with errors relating to initialization of the PPPM FFT grids. This is due to the fact that the partitions in this parallelization strategy differ in size along one axis only, thus with higher ratios, the region of space local to a processor along the special dimension becomes smaller and smaller, which to maintain the specified accuracy, the FFT grid must increase in fineness.

This special axis is taken to be the z-axis by default, which should be appropriate for homogenous systems. In other cases it may be necessary to manually specify the processor grids, so that either the x- or y-axis are integer multiples between partitions. The following commands to specify a job of 16 MPI ranks with a 3:1 ratio would appear at the top of the LAMMPS input script.

```
partition yes 1 processors 2 3 2
partition yes 2 processors 2 1 2
```

To use this strategy in RAPTOR, nothing special is required for compilation. In the LAMMPS input script, the run_style can remain ”verlet”, but an additional keyword ”multipro” needs to be placed before the ”read_data” or ”read_restart” commands in the input script.

```
multipro    read_data      lmp.data
multipro    read_restart   evb.restart
```

8.4 Complex Decomposition for SCI

Just as in the state decomposition strategy discussed above, the energies and forces for different reactive complexes in an SCI simulation can all be evaluated independent of one another at each iteration during each MD step. The use of this parallelization algorithm is activated by adding a keyword ”multipro_sci” in front of the ”read_data” or ”read_restart” keywords in the input script.

```
multipro_sci    read_data      lmp.data
multipro_sci    read_restart   evb.restart
```

Submission of a job with this parallelization strategy is similar to that for any other multi-partition job discussed in this section.

```
mpirun -np 64 ./lmp.x -sf omp -in lmp.in -p 4x16
env OMP_NUM_THREADS=2 mpirun -np 32 ./lmp.x -sf omp -in lmp.in -p 4x8
```

These two examples would both run SCI simulations using 4 separate partitions of processors, each with 16 processing elements. SCI simulations with this complex-decomposition strategy can be run with OpenMP parallelism using the PPPM kspace_style. Ewald can be used, but is not currently supported with OpenMP. There is currently a restriction that all partitions must be the same size and shape. With Multipro does not support extra 1-2 couplings. While they may naturally be filtered away when deleting overlap between reactive complexes it is strongly recommended to turn off extra 1-2 couplings by add the following section to the EVB parameter file.

```
[segment.extension]

[state_search.start]
EXTRA_COUPLINGS  0
REFINE           0
[state_search.end]

[segment.end]
```

The multipro_sci algorithm is not used for state-decomposition of single-complex simulations.

8.5 GPU Acceleration

** GPU support in RAPTOR is limited and performance is poor. This is still a work in progress. Talk to Chris before attempting to venture.**

The current version of RAPTOR can utilize GPU accelerators via the LAMMPS GPU add-on package. To compile RAPTOR with GPU support, the ”-D_RAPTOR_GPU” preprocessor flag is required in the LAMMPS Makefile. To use the GPU package, the gpu library must first be compiled. Instructions for compiling the GPU package on a Mac desktop/laptop (with recent OS) and OpenCL are given below. A similar strategy can be used for compiling on compute clusters with NVIDIA GPUs.

First edit the lammps/lib/gpu/Makefile.lammps file as follows.

```
gpu_SYSINC =
gpu_SYSLIB = -framework OpenCL
gpu_SYSPATH =
```

The following set of commands can then be used to compile LAMMPS with the GPU package.

```
cd lammps/lib/gpu
make -f Makefile.mac_opencl
./ocl_get_devices  # Sanity check; note the device ID for the GPU
cd ../../src
make yes-GPU
rm USER-RAPTOR/EVB_cracker.h EVB_cracker.h
make yes-USER-RAPTOR
make mac  # Makefile used for Mac desktop/laptop with -D_RAPTOR_GPU flag.
```

The defaults for the command-line flags ”-sf gpu” are fine for nonreactive simulations, but not appropriate for simulations with RAPTOR. Instead, the following can be placed at the top of the LAMMPS input script for simulations that will utilize the GPUs.

```
package gpu force 1 1 1.0
newton off on
```

The command ”newton off” should always be used for simulations with GPUs. For the current RAPTOR implementation, neighbor lists must be built on the CPUs, which is why the ”force” option is used as opposed to the ”force/neigh” option. The ”split” option is currently required to be ”1.0”, so that the GPU evaluates all of the pairwise interactions. When running a GPU-RAPTOR simulation, the gpu fix is ”deactivated” so that LAMMPS itself does not perform any GPU operations giving complete control of the GPU to RAPTOR. Thus, support for optimizing the fraction of particles to assign to the GPU is not currently available in RAPTOR. If the device

ID for the GPU accelerator differs from ”1”, then change the device ID accordingly. The correct ID can be discerned from output from the ocl_get_devices script mentioned above. The LAMMPS output can also be check to verify that the correct device is being used. The only other changes to the input file are to append the selected pair_style with ”/gpu”. Currently, there is only support in RAPTOR for the ljcutcoullong pair_style.

```
pair_style lj/cut/coul/long/gpu 9.0
kspace_style pppm/gpu 1e-5
```

Support for PPPM on the GPUs is currently available, but only with the use of a real-space method for the off-diagonals.

GPU support in RAPTOR is currently limited to single-complex simulations using single-partition parallelization algorithms.

8.6 Intel MIC Acceleration

With the USER-MIC package installed, the ”-D_INTEL_MIC” preprocessor flag is required in the LAMMPS Makefile. The following is the relevant section of the Makefile for use on TACC's stampede.

```
CXX =           mpiicpc -vec-report7
CCFLAGS =       -g -O3 -xhost -mkl -openmp -D_INTEL_MIC \
                -DLAMMPS_MEMALIGN=64 -opt-report-phase=offload
DEPFLAGS =      -M
LINK = $(CXX)
LINKFLAGS = -g -O3 -xhost -mkl -openmp
```

The following is an example batch submission script for using all resources on a single-MIC compute node. This example runs 4 MPI x 4 OpenMP processes on the CPUs and 4 MPI x 60 OpenMP processes on the Intel MIC.

```
#!/bin/bash
#SBATCH -J RAPTOR
#SBATCH -o RAPTOR.o%j
#SBATCH -N 1 -n 4
#SBATCH -p normal-mic
#SBATCH -t 0:30:00

#
# -- Initialization
```

```
WRKDIR=$WORK/LAMMPS/benchmarks/1H_50KWater_Nonreactive/mic/1node
EXE_MIC=${HOME}/bin/lammps/src/lmp_stampede-offload
OFFLOAD_AFFINITY=${HOME}/bin/offload/offload_affinity.slurm
cd $WRKDIR

module switch mvapich2 impi

export MIC_LD_LIBRARY_PATH=$MIC_LD_LIBRARY_PATH:
                /opt/apps/intel/13/composer_xe_2013.2.146/mkl/lib/mic

NUM_NODES=1

NUM_MICS_PER_NODE=1
export OMP_NUM_THREADS=4
NUM_MICS=`expr $NUM_NODES \* $NUM_MICS_PER_NODE`

export MIC_PPN=4
export MIC_OMP_NUM_THREADS=60
export MIC_MY_NSLOTS=`expr $MIC_PPN \* $NUM_MICS`

export MKL_MIC_ENABLE=1
export MIC_ENV_PREFIX=MIC

export MIC_KMP_AFFINITY=noverbose,granularity=fine,compact

NUM_STEPS=50
LMP_ARG="-sf omp -in lmp.mic.in -var NSTEPS $NUM_STEPS"

ibrun ${OFFLOAD_AFFINITY} $EXE_MIC ${LMP_ARG}
```

The ”offload_affinity.slurm” script is necessary to pin MIC processes to CPU processes and can be obtained from Chris. Current support for offloading work to the Intel MIC is limited to the computation of pairwise interactions. There is currently only support for the lj/cut/coul/long pair_style. This is an ongoing development project in collaboration with staff at TACC. Support is currently limited to compute nodes with a single MIC co-accelerator. The code will soon be generalized to run on compute nodes with two MIC co-accelerators. ** Talk to Chris about the current status of the USER-MIC package before attempting to venture.

8.7 Approximate Electrostatic Algorithms

The strategies discussed thus far do not alter the accuracy of the calculation, only the parallelization strategy. To further reduce the computational costs of the k-space electrostatic calculations, several approximations can be utilized. The first of which is reducing the precision of variables used in FFT-related arrays. Switching from double to single precision FFTs requires a few small changes to the Makefile assuming single-precision versions of the appropriate libraries are already present. In older Makefiles, the following would is appropriate.

```
CCFLAGS = -DFFT_FFTW -DFFT_SINGLE -DFFTW_SIZE
USRLIB = -lsfftw
```

In newer Makefile, the following should work.

```
FFT_INC = -DFFT_FFTW -DFFT_SINGLE -DFFTW_SIZE
FFT_LIB = -lsfftw
```

The ”-DFFT_SINGLE” flags activates the use of single-precision FFTs and the ”-DFFTW_SIZE” flag indicates that the library has been named to indicate the precision. If the ”-DFFTW_SIZE” flag is present, then the code will expect to link against libsfftw.*, otherwise, it will expect to link against libfftw.*. Making this change can affect energy conservation and it is the responsibility of the user to ensure that any introduced drift is manageable.

The second option available is to evaluate off-diagonal k-space energies and forces using a short-ranged real-space approximation. This is reasonable given that the exchange charges are typically an order of magnitude smaller than the partial charges on atoms in the environment. The method used to evaluate electrostatic interactions for the off-diagonal terms is controlled in the corresponding sections in the evb parameter file, as already discussed in Section 5.4.4.[?]

```
: Potential part
  -32.419         : vij_const, in kcal/mol
    1                 : if contains Vij_ex part
```

Optional values for the method used to evaluate electrostatics in the off-diagonals are given below.

0. No electrostatics. $V_{EX} = 0$
1. A k-space algorithm is used as defined in the input script.
2. Debye screening. The screening parameter is an additional parameter expected on the same line.
3. A damp-shifted force expression of the Wolf-type. The expressions for energy and force are given in Equations 18 and 19 in the work by Fennell and Gezelter.[?]
4. A damp-shifted force expression derived from analysis of numerical CGIS results. The expression for energy is given by Equations 19-23 in the work by Shi, Liu, and Voth.[?]

Energy conservation should not be too strongly perturbed by the Wolf and CGIS approximations, since the energy and force both smoothly go to zero at the cutoff. It is the responsibility of the user to ensure that the use of these approximations does not significantly alter their results.

A third option is available to additionally approximate the diagonal k-space energies and forces. This approximation is available via the ”ewald/acc” and ”pppm/acc” kspace_styles. The use of these approximations requires the use of a real-space method for off-diagonal electrostatics, for example option 4 to use the CGIS expressions. A brief summary of the algorithm for this approximation follows and more detail can be found in the original paper.

1. Use coarse FFT grid to caclucate k-space energies for diagonal matrix elements.
2. Use real-space approximation for off-diagonal matrix elements.
3. Diagonalize Hamiltonian matrix
4. Use ground state eigenvector to construct set of effective charges for atoms within complex.
5. Use effective charges in single k-space force calculation on original fine grid.

Since the Hamiltonian and forces are not strictly consistent, it is the responsibility of the use to ensure that any introduced drift in energy is manageable and that results are not significantly affected by the approximation.

8.8 Hamiltonian Screening

The Hamiltonian screening algorithm filters out unimportant diabatic states to eliminate the wasteful computation of high energy states that essentially contribute zero to the ground state wavefunction. The algorithm currently employed is based on local, computationally cheap interactions. For the diagonal matrix elements, only the bonded interactions are computed. For the off-diagonal matrix elements, only the geometric factor is computed. The coefficients of the ground state eigenvector from this computationally cheap Hamiltonian are then compared to user-specified tolerances to decide which matrix elements are retained for evaluation of the full Hamiltonian matrix (now of a smaller dimension). The algorithm can be activated by adding the following to your evb.par file as an extended section.

```
[segment.extension]

[state_search.start]
EXTRA_COUPLINGS  0
REFINE           0
[state_search.end]
```

```
#ifdef SCREEN_HAMILTONIAN
[screen_hamiltonian.start]
MINP 1e-5
MINP2 1e-9
[screen_hamiltonian.stop]
#endif

[segment.end]
```

It is also advantageous to turn off extra 1-2 couplings as they rarely survive the Hamiltonian screening. Turning off the extra couplings is also useful for SCI simulations, where they rarely survive the SCI calculation when overlap between complexes is removed. A diabatic state is retained for the full calculation only if its weight (c_j^2) is larger than the MINP tolerance and its coupling ($c_j c_k$) to another diabatic state is larger than MINP2. These tolerances only affects the decision of what diabatic states are filtered away and otherwise have no effect on the calculation of the full interactions. The values shown above have yield good energy conservation for the case of a single proton in bulk water using the MS-EVB3 model. It is the user's responsibility to determine appropriate values for the system/model and to validate that the employed parameters do not introduce artifacts in their simulations.

If used, the "#ifdef SCREEN_HAMILTONIAN" and "#endif" logic when defined in the "in.evb" file is a convenient way to quickly turn this algorithm on/off. It is not required.

With this algorithm, one can ideally run calculations in the complete basis set limit (large shell limit and cutoff in state_search sections) and only those important states would be retained. For the case of the MS-EVB3 proton in bulk water, experience has shown that one can run a simulation with a shell limit of 4 resulting in improved energy conservation, but at only half the cost of a typical shell limit of 3 simulation.

Support for Hamiltonian screening is currently available for single- and multicomplex simulations using both single- and multipartition parallelization algorithms.

9 FitEVB

There is also a 'FitEVB.docx' file that has been prepared with current information that may not be contained here. It is recommended that the latest version of FitEVB be downloaded from svn repository.

```
svn co https://w3.rcc.uchicago.edu/svn/fitevb/trunk FitEVB
```

9.1 Input Files

The main directory that the **FitEVB** will be run should contain several input files to control the running of the code and a **DATA** directory that contains all reference and checkpoint data.

9.1.1 ga.in

When using the genetic algorithm for optimizing model parameters, this input file is used to initialize several control variables. This input file contains the control statements and identifies which parameters will be optimized.

```
NGENOME    160000
NSELECT        50
RMUTATE        10
OUTPUT         10
TOTAL           0
```

The following is a brief explanation for all acceptable keywords.

1. **NGENOME**: The total number of parameter vectors that will be maintained. This is the sum of the number of children and parents. The default value is zero.

2. **NSELECT**: The number of parameter vectors (parents) used to generate new parameter vectors (children). The total number of children is NGENOME minus NSELECT. The default value is zero.

3. **RMUTATE**: The total number of mutatations made at each step. For each mutation, an element (gene) of a parameter vector (child) is selected at random and replaced with a random value within the specified range. The default value is zero.

4. **OUTPUT**: This is the frequency with which a restart file is written to "fit.restart". The default value is 60.

5. **TOTAL**: This is the frequency with which an output file is written to “ga.total” with some general information on how the optimization is proceeding. The default value is 10.

6. **SCREEN**: When given the “OFF” option, this command turns of the writing of output to standard output. The default value is “YES”.

The defaults for most parameters are set to zero, so it is advisable to set all of the keywords listed in the example above.

9.1.2 fit.inp

This file contains a list of each model parameter that is to be optimized in the calculation. This file is divided into sections for each contribution to the energies and forces that model paramters will be optimized. Currently, this would correspond to different sections for fitting model parameters of the off-diagonal couplings and short-range repulsions in an MS-EVB3 model. The example below is used to fit the off-diagonal coupling using the functional form of the MS-EVB3 model.?

```
FIT  ARQ2  H3O_H2O
     -2.0     3.0    : g
     -0.5     2.5    : P
    -10.0    10.0    : k
     -1.0     5.0    : D
    -10.0    10.0    : beta
     -2.0     3.5    : R00
      5.0    15.0    : P1
     -5.0    15.0    : alpha
     -3.0     3.0    : r00
    -65.0   -25.0    : Vij_const
```

The first line of each section contains the ’FIT’ command followed by the functional form to be fit and the name of the interaction given in the checkpoint files generated by the Raptor code. Each line afterwards contains the lower and upper limits to be sampled by the optimization algorithm. The following is a list of the currently implemented functional forms. The list of parameters for each function can be found at the top of the respective compute() function in the **FitEVB** source code.

Off-diagonals

1. **ARQ**: Fit a general non-symmetric off-diagonal coupling using the MS-EVB3 functional form without exchange charges. This is used when fitting amino acid models.

2. **ARQ2**: Fit a general symmetric off-diagonal coupling for the MS-EVB3 functional form without exchange charges. This is used when fitting models for the excess proton.

3. **ARQ3**: Fit a general symmetric off-diagonal coupling for the MS-EVB3 functional form with nonzero exchange charges. This is used when fitting models for the excess proton.
4. **ARQ4**: Fit a general symmetric off-diagonal coupling for the MS-EVB3 functional form with nonzero exchange charges. This is used when fitting models for the hydroxide ion.
5. **DAH**:
6. **DA_Gaussian**: Fit a simple Gaussian off-diagonal as function of separation of two atoms without exchange charges.
7. **Table**:

Diabatic Shifts

1. **REP1**: Fit in form of short-ranged repulsions using the MS-EVB3 functional form when fitting models for the excess proton.
2. **REP2**: Fit in form of short-ranged repulsions for the MS-EVB3 functional form when fitting models for the hydroxide ion.
3. **VII**: Fit as a constant diabiatic shift. This is typically used for assymmetric reactions, such as those involving amino acids.
4. **Morse**: Fit a diabatic correction with a Morse potential (specialized).
5. **Pair**: Fit a diabatic correction with a pairwise LJ potential (specialized).

9.1.3 INDEX

This file contains a list of file names that the **FitEVB** code will use to search for input for each configuration in the training set. The first line contains the number of configurations that will be included. This is followed by a list of filenames with one per line. Below is a simple example.

```
4
1
TWO
333
FouR
```

With this list of filenames, the **FitEVB** code will search for the following files in the **/DATA** directory.

```
CHK_1       REF_1
CHK_TWO     REF_TWO
CHK_333     REF_333
CHK_FouR    REF_FouR
```

9.1.4 ENERGY

This file contains a list of potential energies for each of the configuration names given in the INDEX file. The first line indicates which configuration is to be used as the zero of energy. In the example below, this would be the second configuration.

```
ZERO   TWO
 1    -1904.7238
 TWO  -1905.3051
 333  -1903.3633
 FouR -1904.8856
```

9.1.5 WEIGHT

This file controls the different weighting options for the evaluation of the residual used in the minimization procedure. Different weights can be given for energies/forces, configurations, and even individual atom indices. An example of this file is given below.

```
energy 0.5
point * 1.47736141596464
atom  * 1 0.708813792540141
atom  * 2 2.04051115090347
atom  * 3 2.42977708218165
```

The first line indicates the relative weight used for energies and forces. In the example below, the energies are included with a 50% weight. The statement 'energy 0.0' would indicate that only the forces are used in the contruction of the residual. To turn off any special weighting for the configurations and atoms, the following could be used for the **WEIGHT** file.

```
energy 0.0
point *   1.0
atom  * * 1.0
```

9.1.6 /opt

The **opt** directory contains the coordinates for the atoms in all configurations. The names of the files to be read are controlled by the names given in the first column of the **metadata** file. In this example, the code will expect to find the files **1.xyz**, **2.xyz**, ..., **11.xyz**. Each file contains the coordinates of all atoms in the order specified in the **index** file and is formatted as in the following example.

```
C                 -0.00120000    0.00000000    0.00060000
H                 -0.00240000   -0.00240000    1.08970000
C                  1.45030000    0.00510000   -0.53940000
N                 -0.65620000    1.25430000   -0.40670000
H                 -0.41530000    2.05310000    0.15940000
N                  2.46640000   -0.00260000    0.35170000
H                  3.37490000   -0.00260000   -0.09170000
C                  2.41710000   -0.12390000    1.80950000
```

The first character on each line is read by the code, but is not used for anything. The coordinates have units corresponding to those defined in the Lammps input script (Å if ”real” units are used).

9.1.7 /force

The **force** directory contains the reference forces for all atoms in all configurations. The names of the files to be read are controlled by the names given in the first column of the **metafile** file. In this example, the code will expected to find the files **1.force**, **2.force**, ..., **11.force**. Each file contains the forces of all atoms in the order specified in the **index** file and is formatted just like the files containing the coordinates. Again, the first character on each line is ignored by the code and the forces have units corresponding to those defined in the Lammps input script (kcal/mol per Å if ”real” units are used).

9.1.8 ga.chk

This file contains the initial genomes use to start the optimization.

9.2 Running Fitevb

9.3 Output Files

9.4 ToDo List

1. Flexible input to fit any model parameter
2. If ’ga.chk’ file not found, then initialize with random genomes.

10 Additional Small Programs

10.1 restart2data

This tool is out-dated and no longer necessary with recent versions of LAMMPS. The write_data keyword provides the same functionality. This program, part of the standard Lammps code, converts a binary restart file generated during a LAMMPS simulation to an ASCII data file that can be used to start a new simulation. One important point is that the restart2data tool corresponds to specific versions of the Lammps code and must remain consistent. If changes to Lammps lead to changes in how the restart files are written, then you must use an up-to-date restart2data tool to manipulate the generated restart file. If you develop and maintain your own pairstyles, then it will be necessary for you to maintain a separate version of this file to propertly read the restart information that your pair_style writes to file.

10.2 restart2top

This tool is out-dated and no longer necessary with recent versions of LAMMPS. The write_top keyword provides the same functionality. This program, which is a slightly edited version of the restart2data tool, takes the binary restart file generated during a LAMMPS simulation to an ASCII top file that can be used to start a new simulation. Whenever the restart2data tool is updated, this tool must also be updated. An up-to-date version of this tool should always be available for download from main raptor directory on makena or the group wiki.

11 Writing & Maintaining your own add-on packages

12 LAMMPS MS-EVB Tutorial

This tutorial will walk a user through a typical calculation, using the LAMMPS MS-EVB code for the hydrated proton.

12.1 Water

12.2 Common Error Messages

•

13 Detailed Notes Regarding Algorithms

13.1 SCI-MS-EVB

13.2 Biasing Potentials Applied to CEC

This section will discuss the implementation of biasing potentials applied to the center of excess charge (CEC) in MS-EVB simulations. The forces due to the biasing potential will involve derivatives of the ground state MS-EVB eigenvector with respect the particle coordinates.[?,?]

The position of the CEC is defined as follows

$$\mathbf{r}_{CEC} = \sum_{j=1}^{N_{states}} c_{0i}^2 \, \mathbf{r}_{COC,j}, \tag{13}$$

where c_{i0} are the coefficients of the ground state eigenvector and $\mathbf{r}_{COC,j}$ is the position of the center of charge (COC) for the jth MS-EVB state. The forces due to the biasing potential, $V(\mathbf{r}_{CEC})$ can be calculated using the chain rule.

$$\mathbf{F}_j = -\frac{\partial V(\mathbf{r}_{CEC})}{\partial \mathbf{r}_j} = -\frac{\partial V(\mathbf{r}_{CEC})}{\partial \mathbf{r}_{CEC}} \frac{\partial \mathbf{r}_{CEC}}{\partial \mathbf{r}_j} \tag{14}$$

The first derivative can be evaluated straightforwardly when the form of the biasing potential has been defined. Using the definition for the CEC coordinate, the second derivative can be written.

$$\frac{\partial \mathbf{r}_{CEC}}{\partial \mathbf{r}_l} = \frac{\partial}{\partial \mathbf{r}_l} \sum_{j=1}^{N_{states}} c_{0i}^2 \, \mathbf{r}_{COC,j} = \sum_{j=1}^{N_{states}} \left[2 \frac{\partial c_{0i}}{\partial \mathbf{r}_l} \, \mathbf{r}_{COC,j} + c_{0i}^2 \frac{\partial \mathbf{r}_{COC,j}}{\partial \mathbf{r}_l} \right] \tag{15}$$

The derivatives involving the COC coordinate are straightforwardly evaluated using the definition for the COC coodinate.

$$\mathbf{r}_{COC,j} = \frac{1}{\sum_{k=1}^{N_{A,COC}} |q_{k,j}|} \sum_{k=1}^{N_{A,COC}} |q_{k,j}| \mathbf{r}_k, \tag{16}$$

where the sum is over all atoms in the reactive molecule. The derivative involving the ground state eigenvector coefficients can be derived using first order perturbation theory.[?,?]

$$\frac{\partial c_{0i}}{\partial \mathbf{x}_k} = -\sum_{j \neq 0}^{N_S} \sum_{lm}^{N_S} \frac{\mathbf{F}_k^{lm} c_{jl} c_{0m}}{E_0 - E_j} c_{ji} \tag{17}$$

In Raptor, the derivatives of the COC coordinates are evaluated in the **EVB_CEC::decompose_force()** function. The derivatives of the eigenvector coefficients are calculated in the **EVB_CEC::partial_C_N3()**

function. In this function, the dummy arrays are defined in the following sequence.

$$\textbf{array1}[l] = -\sum_{m=0}^{N_S} \mathbf{F}_k^{lm} c_{0m} \tag{18}$$

$$\textbf{array2}[j] = \sum_{l=0}^{N_S} \textbf{array1}[l] c_{jl} = -\sum_{lm}^{N_S} \mathbf{F}_k^{lm} c_{jl} c_{0m} \tag{19}$$

$$\textbf{array1}[j] = \frac{\textbf{array2}[j]}{E_j - E_0} = -\sum_{lm}^{N_S} \frac{\mathbf{F}_k^{lm} c_{jl} c_{0m}}{E_j - E_0} \tag{20}$$

$$\frac{\partial c_{0i}}{\partial \mathbf{r}_k} = \textbf{dcdx}[i] = \sum_{j\neq 0}^{N_S} \textbf{array1}[j] c_{jl} = -\sum_{j\neq 0}^{N_S} \sum_{lm}^{N_S} \frac{\mathbf{F}_k^{lm} c_{jl} c_{0m}}{E_j - E_0} c_{ji} \tag{21}$$

13.3 Proposal for New State Search Algorithm

A proposal to improve the current state search algorithms (MS-EVB2 and MS-EVB3) is discussed here. The issue with the current algorithms is the abrupt addition/removal of MS-EVB states with significant weight whenever a chemical reaction takes place. For example, in Figure 7, the proton transfer between hydronium and water is shown. The molecule with the blue center (hydronium) is the molecule used to initiate the state search. In this example, only molecules within the second solvation shell of hydronium are considered for clarity. When a chemical reaction occurs (proton transfer), the identity of the molecule used to initiate the state search changes which can lead to signifcant changes in which MS-EVB states are included. In the figure, those molecules with green centers were removed or included depending on whether they are found outside or within the second solvation shell of the new molecule used to initiate the state search.

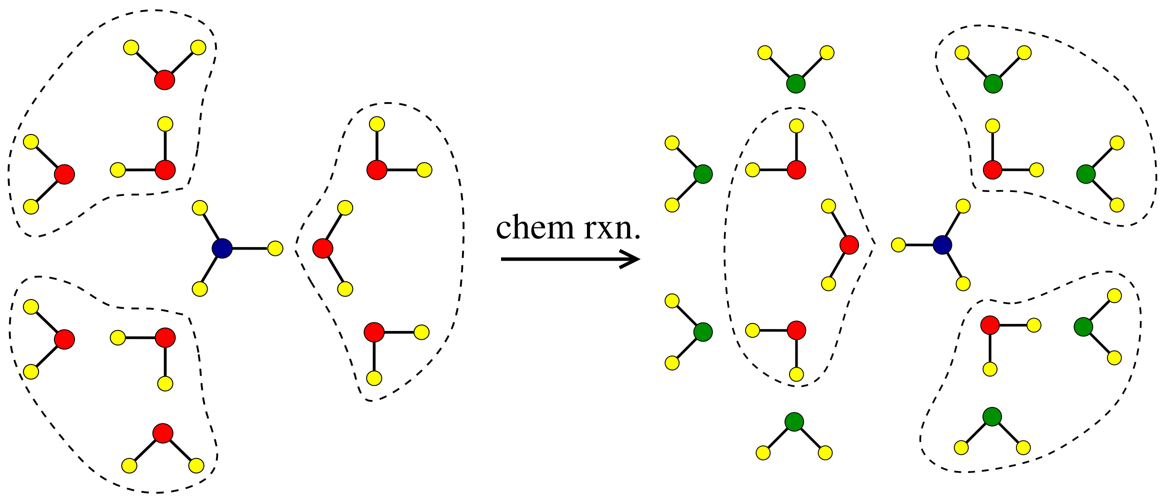

Figure 7: A sketch of the current state search algorithms illustrating the addition/removal of states of significant weight as chemical reactions occur in MS-EVB simulations as discussed in detail in the text. For both reactant and product, those molecules within the dashed circles indicate the corresponding MS-EVB states included. Those molecules with green atom centers are the molecules added/removed which correspond to MS-EVB states with significant weight and thus lead to poor energy conservation properties.

In the current state search algorithms, the outermost solvation shell of the initial molecule is an input parameter. Typically, one only considers molecules within and including the third solvation shell. The idea being that the interaction potentials are sufficiently short-ranged and the local configuration of the outermost shell is not favorable for the solvation of the reactant molecule so that the corresponding MS-EVB weights are negligible. The addition/removal of states with essentially zero weights have a negligible contribution to the forces. However, the removal of MS-EVB states with significant weight can have a large impact on the forces and thus introduces a "kick" to the system. The accumulation of these "kicks" over the course of a simulation manifests itself as a drift in the total energy. One solution would be to simply included more solvation shells in the state search algorithm. While this would remedy the energy conservation issue, one would now have to evaluate an exceeding large number of MS-EVB states at each MD step increasing the computational cost significantly. It is likely that the majority of these extra states would have negligible weight and thus a waste of time and effort to evaluate. The algorithm proposed here is a first step to minimizing the magnitude of the "kicks" to the system and thus improve the energy conservation properties of MS-EVB simulations independent of the system being studied by incorporating only those states which are significant.

The proposed algorithm is outlined below. Please note that we are not making changes to what type of MS-EVB states are included. For example, one of the differences between the MS-EVB2 and MS-EVB3 algorithms is that the MS-EVB3 state search includes both acceptors of a bifurcated H-bond while the MS-EVB2 algorithm only includes the "best" acceptor. It may be necessary for improved algorithms to incorporate additional types of states not considered by these current algorithms, but that is not the focus here. Instead, the focus will be how to best include/remove states which ONLY have insignificant weight and thus minimizes any drift in the total energy over the course of a simulation. The goal of this algorithm would be to already include all MS-EVB states on the right side of Figure 7 before the reaction is attempted. In the following, we refer to the MS-EVB2 and MS-EVB3 algorithms as templates for a state search calculation.

ALGORITHM:

The MD simulation is initialized by running a state search template on the initial reactant molecule including one more solvation shell than specified. For the rest of the simulation, the following steps are taken.

1. All states from the previous MD step are included except those with weights below some tolerance.
2. A state search template is run using the current reactant molecule to include any states not in the current list of states.
3. If the weight of any state (not reactant) is above some tolerance indicating that a reaction may soon occur, then also run a state search template for that molecule and include any states in the current list of states.

This history dependent algorithm would require information written to a restart file for the successful continuation of a trajectory.

With this algorithm, it is the following intentions that:

1. Those states with insignificant weight are not included in the next MD step under the assumption that they will again have insignificant weight in the very next MD step. This is an attempt to keep the number of states down.

2. By keeping all important states from the previous step, we prevent the potential removal of states based soley on the distance criteria.

3. Running a state search template on those molecules with weights above some tolerance mimics the inclusion of an extra solvation shell, but includes only those states which potentially have nonzero weight.

13.4 Ewald

The derivation of the necessary expressions can be found in any standard solid state physics text. The expression for the potential energy is

$$\begin{aligned} V &= \frac{2\pi}{V}\sum_{k\neq 0}^{\infty}\frac{1}{k^2}e^{-k^2/4\alpha^2}\left|\sum_{j=1}^{N} q_j e^{-i\mathbf{k}.\mathbf{r}_j}\right|^2 + \sum_{i<j}\frac{q_i q_j}{r_{ij}}\mathrm{erfc}(\alpha r_{ij}) - \frac{\alpha}{\sqrt{\pi}}\sum_{j=1}^{N} q_j^2 - \qquad (22) \\ & \sum_{\substack{i<j \\ ij\,\in\, molecule}} \frac{q_i q_j}{r_{ij}}\mathrm{erf}(\alpha r_{ij}) + \frac{2\pi}{(2\epsilon_S+1)V}\left|\sum_{j=1}^{N} q_j \mathbf{r}_j\right|^2 - \frac{\pi}{V2\alpha^2}Q_{TOT}^2 \qquad (23) \end{aligned}$$

A factor of $1/4\pi\epsilon_0$ has been factored out of the expression and multiplication by the **qqrd2e** variable in Lammps accounts for this. The first three terms correspond to the k-space, real space, and the self-interaction correction to the k-space term. The fourth term corrects for the intramolecular contributions that were included in the k-space calculation which are normally omitted. The second from last term accounts for dipolar interactions with the surrounding environment. Typically, conducting (or ”tin-foil”) boundary conditions are assumed, $\epsilon_S = \infty$. If vacuum, $\epsilon_S = 0$ boundary conditions are used, then one must be careful to remove discontinous jumps in energy as particles move across the periodic boundaries. The last term is a correction for non-neutral simulation cells.

Some useful expressions for the calculation of the forces follow.

$$\left|\sum_{j=1}^{N} q_j e^{-i\mathbf{k}.\mathbf{r}_j}\right|^2 = \sum_{i,j}^{N} q_i q_j e^{-i\mathbf{k}.\mathbf{r}_{ij}} \qquad (24)$$

$$\begin{aligned} \mathrm{erf}(x) &= \frac{2}{\sqrt{\pi}}\int_0^x dte^{-t^2} \qquad (25) \\ \mathrm{erfc}(x) &= 1 - \mathrm{erf}(x) \qquad (26) \end{aligned}$$

$$\begin{aligned} \frac{d}{dx}\mathrm{erf}(x) &= \frac{d}{dx}\left\{\frac{2}{\sqrt{\pi}}\int_0^x dte^{-t^2}\right\} = \frac{2}{\sqrt{\pi}}e^{-x^2} \qquad (27) \\ \frac{d}{dx}\mathrm{erfc}(x) &= -\frac{2}{\sqrt{\pi}}e^{-x^2} \qquad (28) \end{aligned}$$

The particle-particle contribution to the force on the *m*th atom due to the k-space interactions is

$$\mathbf{F}_m = 2q_m\frac{2\pi}{V}\sum_{j=1}^{N} q_j \sum_{k\neq 0}^{\infty}\frac{1}{k^2}e^{-k^2/4\alpha^2}\sin[\mathbf{k}.(\mathbf{r}_j - \mathbf{r}_m)]\mathbf{k} \qquad (29)$$

The particle-particle contribution to the force on the *j*th atom due to the real space interactions is

$$\mathbf{F}_j = q_j \sum_{i \neq j}^{N} \frac{q_i}{r_{ij}^3} \Big\{ \mathrm{erfc}(\alpha r_{ij}) + \frac{2\alpha r_{ij}}{\sqrt{\pi}} e^{-\alpha^2 r_{ij}^2} \Big\} \mathbf{r}_{ij} \tag{30}$$

The particle-particle contribution to the force on the *j*th atom due to the intramolecular corrections is

$$\mathbf{F}_j = q_j \sum_{i \neq j}^{N} \frac{q_i}{r_{ij}^3} \Big\{ -\mathrm{erf}(\alpha r_{ij}) + \frac{2\alpha r_{ij}}{\sqrt{\pi}} e^{-\alpha^2 r_{ij}^2} \Big\} \mathbf{r}_{ij} \tag{31}$$

13.4.1 LAMMPS

The variable “qqr2de = qqrd2e / dielectric” is the Coulomb constant used for calculating electrostatic energies and forces. The relative dielectric constant is defined to be one unless specified otherwise in the Lammps input script.

$$qqr2e = \frac{1}{4\pi\epsilon_0} = 8.9876 \text{ x } 10^9 J\ m\ C^{-2}, \tag{32}$$

where the vacuum permittivity, ϵ_0, is 8.8542 x $10^{-12} F\ m$. The constant “qqr2de” is then defined to be the electrostatic energy between two point charges separated by one Angstrom.

$$\begin{aligned} qqr2de &= \frac{1}{4\pi\epsilon_0}\frac{q_1 q_2}{r_{12}} = \frac{1}{4\pi\epsilon_0}\frac{(1.0e)(1.0e)}{1\text{Å}} && (33) \\ &= 8.9876 \text{ x } 10^9 J\ m\ C^{-2} \frac{(1.6022 \text{ x } 10^{-19}\ C)^2}{10^{-10}\ m} && (34) \\ &= 2.3072 \text{ x } 10^{-18} J && (35) \end{aligned}$$

This value is then divided by Avogadro’s number and converted to kilocalories.

$$qqr2de = \frac{(2.3072 \text{ x } 10^{-18} J)\ (6.022 \text{ x } 10^{23} mol^{-1})}{4184\ J/kcal} = 332.064\ kcal/mol \tag{36}$$

This is the value defined in the **Update::set_units()** function in Lammps when using “real” units.

The definitions for some other variables and arrays are defined below.

- g_ewald: Screening parameter α used in the equations for the Ewald energies and forces.
- kcount: Total number of wavevectors included in calculation.
- kmax: Largest integer component of wavevector included in calculation.
- kmax3d: Number of wavevectors in sphere with radius defined by kmax.

- kxvecs[k]: x-component of *k*th wavevector stored as integer.
- kyvecs[k]: y-component of *k*th wavevector stored as integer.
- kzvecs[k]: z-component of *k*th wavevector stored as integer.
- ug[k]: Wavevector-dependent prefactor calculated in the Ewald::coeffs() function.

$$ug[k] = \frac{4\pi}{V\mathbf{k}^2} e^{-\mathbf{k}^2/4\alpha^2} \tag{37}$$

- eg[k]: 3d force contribution from *k*th wavevector calculated in the Ewald::coeffs() function.

$$eg[k] = 2\mathbf{k} ug[k] \tag{38}$$

The Ewald::eik_dot_r() function calculates the partial structure factor from particles local to a processor. The total structure factor is accumulated on all processors using MPI_Allreduce() calls in the Ewald::compute() function.

- sfacrl[n]: The real contribution to the structure factor for the *n*th wavevector summed over all particles local to the processor.

$$sfacrl[n] = \sum_{i=1}^{N_{local}} q_i \cos(\mathbf{k}_n.\mathbf{r}_i) \tag{39}$$

- sfacim[n]: The imaginary contribution to the structure factor for the *n*th wavevector summover over all particles local to the processor.

$$sfacim[n] = \sum_{i=1}^{N_{local}} q_i \sin(\mathbf{k}_n.\mathbf{r}_i) \tag{40}$$

- sfacrl_all[k]: The real contribution to the structure factor summed over all particles for the *k*th wavevector.
- sfacim_all[k]: The imaginary contribution to the structure factor summec over all particles for the *k*th wavevector.
- cs[**k**,i]: The term cos($\mathbf{k}.\mathbf{r}_i$) for each particle stored as an array for each of the xyz components.
- sn[**k**,i]: The term sin($\mathbf{k}.\mathbf{r}_i$) for each particle stored as an array for each of the xyz components.

The contributions from each wavevector are accumulated (?) using standard trigonometric identities in the eik_dot_r() function.

$$\sin(x \pm y) = \sin(x)\cos(y) \pm \cos(x)\sin(y) \quad (41)$$
$$\cos(x \pm y) = \cos(x)\cos(y) \mp \sin(x)\sin(y) \quad (42)$$

The Ewald::compute() function calculates the electrostatic energies and forces. The self-interaction and non-neutral corrections are added to energy in the compute() function before the result is multiplied by the variable **qqrd2e**.

13.4.2 Raptor

For MS-EVB simulations, the partial structure factor for the environment atoms is calculated once and stored at the beginning of an MD step. The electrostatic energies for each of the diagonal elements are then calculated by calculating the partial structure factor for the reactive complex atoms and adding this to the environment contribution. The same is done for the off-diagonal elements and this leads to significant savings as opposed to recalculaing the entire lattice sum for each nonzero element of the MS-EVB Hamiltonian. After the MS-EVB Hamiltonian has been diagonalized, the electrostatic forces due to the reactive complex atoms are then calculating using an effective set of charges. With this algorithm, the electrostatic forces between complex and environment atoms only need to be calculated once.[?]

13.5 PPPM

13.5.1 LAMMPS

The following is a brief description for most of the functions in the Lammps PPPM code.

1. init() : Allocates memory, defines constants, special initialization for TIP4P, defines FFT grid and assigns procs, and pre-compute prefactors.
2. setup() : Calculate local wavevectors and Green's functions. This is called initially and whenever volume changes.
3. compute() : Calculate energy and forces.
4. allocate() : Allocate memory for arrays.
5. deallocate() : Deallocate memory for arrays.
6. set_grid() : Calculates precision and FFT grid spacing.
7. brick2fft() : Maps 3D array to 1D array for FFTs. This involves communication between each processor and its neighbors.
8. fillbrick() : Maps 1D array to 3D array after FFTs.
9. particle_map() : Maps particle coordinates to grid.
10. make_rho() : Accumulates real-space charge onto grid.
11. poisson() : Calculates forward FFT of charge density, energy, and 3 backward FFTs to get electric field.
12. fieldforce() : Maps electric field on grid points to particle coordinates to calculate forces.

13.5.2 Raptor

The following is a brief description of each of the functions in the Raptor PPPM code for MS-EVB simulations.

1. clear_density() : Initialized 3D array to zero.
2. load_env_density() : Initializes 3D array with charge density from environment atoms.
3. particle2map() : Slightly modified version of the original particle_map() function.
4. map2density_one() : Accumulates charge density from single particle onto grid. This would be the equivalent to the original make_rho() function if looped over all local particles.

5. poisson_energy() : Calculates forward FFT and energy for a charge distribution.
6. field2force_one() : Maps electric field on grid points to single particle. This would be equivalent to original fieldforce() function is looped over all local particles.
7. compute_env() : Computes the environment-environment contribution to the energy. This is not used in current version.
8. compute_cplx() : Loads environment and complex charge density and then calculates energy
9. compute_exch() : Computes exchange energy from three separate PPPM evaluations. (In need of optimization)

13.6 Effective Long-Range Interactions Method for Electrostatics

The main culprit for the additional computational burden over nonreactive simulations is the explicit calculation of the electrostatic interactions for each matrix element of the MS-EVB Hamiltonian. In this approach, a global Ewald[?] (or PPPM[?]) summation is performed for each nonzero element of the MS-EVB Hamiltonian matrix. In particular, it is the long-ranged k-space calculation involving the Fourier transform of the charge density that degrades parallel efficiency at high processor counts. It would be highly advantageous to develop an algorithm for MS-EVB simulations that would only require a single long-ranged electrostatics calculation for each MD step, independent of the dimension of the MS-EVB Hamiltonian. This single electrostatic calculation would use a set of effective charges for the atoms in the reactive complex calculated as a linear combination of the charges from each of the bonding topologies. The weights for each bonding topology would be obtained from the ground state solution to a reference ”short-ranged” MS-EVB Hamiltonian. This reference Hamiltonian will omit the long-ranged (k-space) contribution, as illustrated

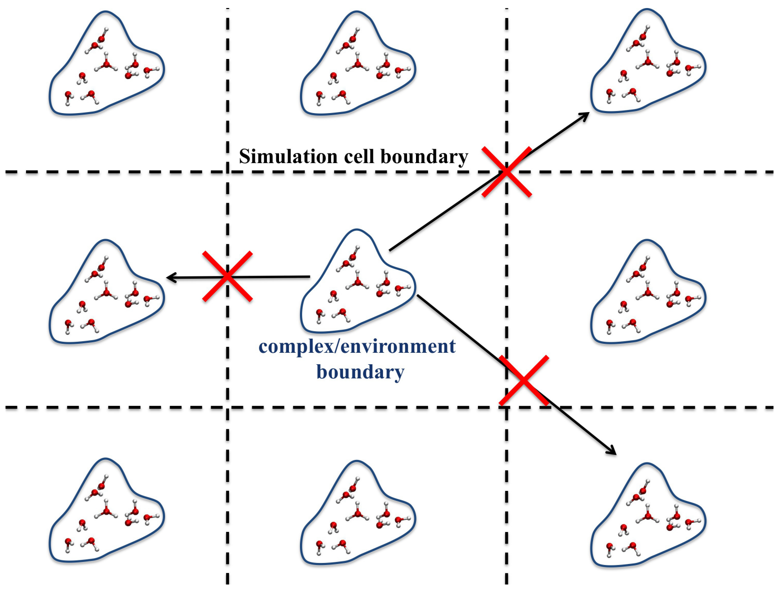

Figure 8: Illustration of the approximation made in the ELRI method whereby the electrostatic interactions with periodic images is neglected in the diagonalization of the MS-EVB Hamiltonian matrix. These long-ranged interactions are included through the use of effective charges for those atoms within the reactive complex.

in Figure 8, from the electrostatic interactions making this effectively a short-ranged Hamiltonian. The approximation made here is that the long-ranged contribution to the electrostatic interactions does not significantly alter the relative weights of the bonding topologies. After the reference ”short-ranged” Hamiltonian has been diagonalized and the effective charges calculated, a proper long-ranged electrostatic method will be used to calculate the electrostatic forces on all atoms, just like in a fixed-topology or nonreactive simulation. This Effective Long-Range Interaction (ELRI) method, also referred to as the FECM method, would significantly reduce the computational cost of a MS-EVB simulation to that of a nonreactive simulation with minimal overhead (hopefully!). After validation for the case of a single excess proton, the next step in the development of this method is its incorporation into the SCI-MS-EVB algorithm, which would significantly speed up the simulation of concentrated systems. This new algorithm, when combined with the MPI/OpenMP and MP programming models, will enable the efficient simulation of reactive events in complicated condensed phase environments consisting of millions of atoms.

The Hamiltonian matrix for this method is the sum of two contributions: short and long-range.

$$\mathbf{H} = \mathbf{H}_{\text{short}} + H_{\text{long}}(\mathbf{c}_0) \tag{43}$$

The long-ranged contribution is treated as a constant that depends on the ground state eigenvector of the short-ranged Hamiltonian. For each step in an MD simulation, a ”normal” MS-EVB calculation is done except that all of the k-space related computations are skipped. The Hamiltonian thus constructed only contains short-ranged contributions, such as bonds, angles, and pairwise interactions. Since k-space calculations, such as Ewald and PPPM, are separated into rapidly convergent real and reciprocal space sums, we choose to include the real space summation in the short-range Hamiltonian. The incorporation of some electrostatic interactions in the short-ranged Hamiltonian should improve the approximation to the ground state eigenvector. The long-range terms that are not evaluated for the MS-EVB Hamiltonian matrix elements are shown below.

$$H_{\text{long}} = \frac{1}{2V\epsilon_0}\sum_{\mathbf{k}\neq 0}^{\infty}\frac{1}{k^2}e^{-k^2/4\alpha^2}\Big|\sum_{j}^{N} q_j e^{-i\mathbf{k}.\mathbf{r}_j}\Big|^2 - \frac{1}{4\pi\epsilon_0}\frac{\alpha}{\sqrt{\pi}}\sum_{j}^{N} q_j^2 \tag{44}$$

$$= \frac{1}{2V\epsilon_0}\sum_{\mathbf{k}\neq 0}^{\infty}\frac{1}{k^2}e^{-k^2/4\alpha^2}\sum_{ij}^{N} q_i q_j e^{-i\mathbf{k}.(\mathbf{r}_j-\mathbf{r}_i)} - \frac{1}{4\pi\epsilon_0}\frac{\alpha}{\sqrt{\pi}}\sum_{j}^{N} q_j^2 \tag{45}$$

The first term is the reciprocal space sum for the periodic charge density and the second term is the self-interaction correction. In the second line, the squared modulus of the kth compoent to the Fourier transformed charged density was expanded. In this expression, the charges, q_j, are the effective charges calculated as the linear combination over all chemical bonding topologies weighted by the coefficients of the ground state eigenvector obtained by diagonalizing the short-ranged Hamiltonian:

$$q_j = \sum_{mn}^{N_S} c_{0m}c_{0n}q_j^{mn}, \tag{46}$$

where N_S is the number of MS-EVB states and q_j^{mn} is the charge on the jth atom for the (m,n)th element of the Hamiltonian matrix which defined the chemical bonding topology.

The forces due to the short-ranged Hamiltonian can be calculated using the Hellman-Feynman theorem as is normally done in an MS-EVB simulation. The forces from the k-space contribution will include the expected terms plus some additional terms due to the implicit dependence of the charges on the ground state eigenvector. These additional $\partial q^{eff}/\partial \mathbf{r}_j$ terms will not necessarily be zero and must be included in the force calculation.

$$\mathbf{F}_j = -\frac{\partial H_{long}(\mathbf{c}_0)}{\partial \mathbf{r}_j} \tag{47}$$

Taking the derivative of the k-space term with respect to an atomic coordinate will yield two terms. One term will be the "normal" force contribution calculated with the Ewald or PPPM methods while the other will contain $\partial q^{eff}/\partial \mathbf{r}_j$ terms. The "normal" term will not be considered any further since it can be calculated using standard algorithms in MD codes.

$$\frac{\partial}{\partial \mathbf{r}_l}\sum_{ij}^{N} q_i q_j e^{-i\mathbf{k}.(\mathbf{r}_j-\mathbf{r}_i)} = \sum_{ij}^{N}\left[\left\{\frac{\partial q_i}{\partial \mathbf{r}_l} q_j + q_i \frac{\partial q_j}{\partial \mathbf{r}_l}\right\} e^{-i\mathbf{k}.(\mathbf{r}_j-\mathbf{r}_i)} + q_i q_j \frac{\partial}{\partial \mathbf{r}_l} e^{-i\mathbf{k}.(\mathbf{r}_j-\mathbf{r}_i)}\right] \tag{48}$$

Similarly, the force contribution from the self-interaction term will now contain derivatives of the effective charges.

$$\frac{\partial}{\partial \mathbf{r}_l}\sum_{j}^{N} q_j^2 = 2\sum_{j}^{N} q_j \frac{\partial q_j}{\partial \mathbf{r}_l} \tag{49}$$

Using the expression above for the effective charges, these derivatives can be expressed in terms of derivatives of the eigenvector coefficients with respect to atomic positions.

$$\frac{\partial q_j}{\partial \mathbf{r}_l} = \frac{\partial}{\partial \mathbf{r}_l}\sum_{mn}^{N_S} c_{0m} c_{0n} q_j^{mn} = 2\sum_{mn}^{N_S} \frac{\partial c_{0m}}{\partial \mathbf{r}_l} c_{0n} q_j^{mn} \tag{50}$$

The charges q_j^{mn} only depend on the chemical bonding topology and not the atomic coordinates. An expression for the derivatives of the coefficients with respect to atomic coordinates can be developed using 1st order perturbation theory. These derivatives actually appear in the calculation of forces due to a biasing potential applied to the CEC. Also, only those atoms contained within the reactive complex have variable charges. The atoms in the environment region have fixed charges independent of the variable bonding topology in the reactive complex. For the self-interaction term, there will be a force on all atoms in the system, but for each contribution, only a sum over the atoms in the reactive complex is needed. For the k-space summation, there will be two types of terms included: complex-complex and complex-environment. The environment-environment terms will be zero and not contribute to the forces.

The current situation is that without these additional force contributions, there is a horrendous drift in the total energy in excess of 1000 kcal/mol per nanosecond. This drift can be minimized by shifting the workload to the realspace calculation and thus minimizing the relative contibution from the k-space forces. At present, the self-interaction contribution to the forces is coded in RAPTOR. Some minor improvement is seen in the drift, but it is still not acceptable. Below is an outline for the implementation of the k-space term to be calculated using the Ewald method.

$$\sum_{ij}^{N}\left\{\frac{\partial q_i}{\partial \mathbf{r}_l} q_j + q_i \frac{\partial q_j}{\partial \mathbf{r}_l}\right\} e^{-i\mathbf{k}.(\mathbf{r}_j-\mathbf{r}_i)} = \sum_{ij}^{N} \frac{\partial q_i}{\partial \mathbf{r}_l} q_j e^{-i\mathbf{k}.(\mathbf{r}_j-\mathbf{r}_i)} + \sum_{ij}^{N} q_i \frac{\partial q_j}{\partial \mathbf{r}_l} e^{-i\mathbf{k}.(\mathbf{r}_j-\mathbf{r}_i)} \tag{51}$$

$$= \left(\sum_{i}^{N} \frac{\partial q_i}{\partial \mathbf{r}_l} e^{i\mathbf{k}.\mathbf{r}_i}\right)\left(\sum_{j}^{N} q_j e^{-i\mathbf{k}.\mathbf{r}_j}\right) + \left(\sum_{i}^{N} q_i e^{i\mathbf{k}.\mathbf{r}_i}\right)\left(\sum_{j}^{N} \frac{\partial q_j}{\partial \mathbf{r}_l} e^{-i\mathbf{k}.\mathbf{r}_j}\right) \tag{52}$$

$$= 2\mathrm{Re}\left[\left(\sum_{i}^{N} \frac{\partial q_i}{\partial \mathbf{r}_l} e^{i\mathbf{k}.\mathbf{r}_i}\right)\left(\sum_{j}^{N} q_j e^{-i\mathbf{k}.\mathbf{r}_j}\right)\right] \tag{53}$$

The two terms involving the charges q_j are just the structure factors which are already calculated in the Ewald::compute() function. The other two terms in parentheses can be calculated in an analogous fashion as the structure factor terms (as discussed in Section 13.4).

13.6.1 Self-Interaction

The energy due to the self-interaction correction is

$$H^{self} = -\frac{1}{4\pi\epsilon_0}\frac{\alpha}{\sqrt{\pi}}\sum_j^N q_j^2, \tag{54}$$

where q_j is the effective charge on the jth atom. The ELRI force on the kth particle due to the self-interaction correction is

$$\mathbf{F}_k^{self} = -\frac{\partial H^{self}}{\partial \mathbf{r}_k} = \frac{1}{4\pi\epsilon_0}\frac{2\alpha}{\sqrt{\pi}}\sum_j^N q_j\frac{\partial q_j}{\partial \mathbf{r}_k}. \tag{55}$$

As stated earlier, only atoms within the reactive complex may have variable charge and so the summation over all atoms can be replaced with a summation over atoms within the reactive complex.

13.6.2 K-space

The energy due to the k-space summation is

$$H^{ksp} = \frac{1}{2V\epsilon_0}\sum_{\mathbf{k}\neq 0}^{\infty}\frac{1}{k^2}e^{-k^2/4\alpha^2}\Big|\sum_j^N q_j e^{-i\mathbf{k}.\mathbf{r}_j}\Big|^2 \tag{56}$$

The ELRI force on the lth particle due to the k-space summation is

$$\begin{aligned}\mathbf{F}_l^{ksp} = -\frac{\partial H^{ksp}}{\partial \mathbf{r}_k} &= \frac{1}{2V\epsilon_0}\sum_{\mathbf{k}\neq 0}^{\infty}\frac{1}{k^2}e^{-k^2/4\alpha^2}\sum_{ij}^N\Big\{\frac{\partial q_i}{\partial \mathbf{r}_l}q_j + q_i\frac{\partial q_j}{\partial \mathbf{r}_l}\Big\}e^{-i\mathbf{k}.(\mathbf{r}_j-\mathbf{r}_i)} &(57)\\ &= -\frac{1}{2V\epsilon_0}\sum_{\mathbf{k}\neq 0}^{\infty}\frac{1}{k^2}e^{-k^2/4\alpha^2}2\mathrm{Re}\Big[\Big(\sum_i^{N_{CPLX}}\frac{\partial q_i}{\partial \mathbf{r}_l}e^{i\mathbf{k}.\mathbf{r}_i}\Big)\Big(\sum_j^N q_j e^{-i\mathbf{k}.\mathbf{r}_j}\Big)\Big] &(58)\end{aligned}$$