YYY Toolkit: Users Manual

Joseph F. Rudzinski, Nicholas J. H. Dunn

May 23, 2017

Contents

1 Installation

The CGMap, CGFF, Tables, and PMatch software uses CMake (`https://cmake.org`) for its build system. In order to build these tools, you should only need to point the CMakeLists.txt file in the root directory of each package to the location of a valid GROMACS 4.x/5.x installation, and to a linear algebra library such as LAPACK or Intel MKL. Then, run the commands:

```
mkdir build
cd build
cmake .. -G "Unix Makefiles" -DCMAKE_C_COMPILER=mpicc -DCMAKE_INSTALL_PREFIX=build
make
```

If all goes well, the code should then compile without errors. The pressure matching code is bundled separately from the other coarse-graining tools, but is built in the same way.

The modified LAMMPS tool does not modify the LAMMPS installation process, so you can follow the directions available at `http://lammps.sandia.gov/doc/Manual.html` to build the modified LAMMPS tool for your system.

Please note that this software has only been tested on Linux systems.

2 CGMap

The CGMap tool maps an all-atom GROMACS trajectory to its coarse-grained representation. It takes as input an all-atom trajectory, CG topology, and mapping topology, and generates a CG trajectory file.

2.1 Basic Usage

Case 1: Generating a CG gro file

```
CGMap-gmx -p {map.top} -f {AA_traj.trr}  -c {CG_traj.trr}
```

Case 2: Mapping a whole AA trajectory

```
CGMap-gmx -p {map.top} -f {AA_traj.trr} -s {CG_topol.tpr} -o {CG_traj.trr}
```

2.2 Mapping Topology File Format

The basic layout of the mapping topology file is quite similar to a simplified GROMACS topology file. Directives are contained in square braces, and semicolons are used to indicate comment lines. Columns within a directive are separated by whitespace. All directives listed below are mandatory, though depending on the system and the chosen mapping, you may have more than one of each. For the in-text example in this section, we will be assembling a mapping topology for a 1-site, center-of-mass model of water.

2.2.1 [moleculetype]

```
[ moleculetype ]
; name   n_atoms
  H2O    3
```

In the ‘moleculetype’ directive, you will give the name or abbreviation of one of the molecule types in your system followed by the number of atoms in the system. This signifies the beginning of the mapping information of that molecule type, and the next two directives must be ‘sitetypes’ and ‘atomtypes’ for that molecule, in that order.

2.2.2 [sitetypes]

```
[ sitetypes ] 1
; name  mapping_type  n_atoms
  1W    com           3
```

In the ’sitetypes’ directive, you will define one or more sites for the current moleculetype by providing names, mapping types, and the number of atoms for each of the sites defined. The number of site types in the molecule must be provided after the ’sitetypes’ directive on the same line. Here, we have defined a single site named ’1W’ and specified that it will be located at the center of mass of three atoms. The currently supported mapping types include:

```
com - Maps the CG site to the center of mass of the indicated atoms
cog - Maps the CG site to the center of geometry of the indicated atoms
usr - Maps the CG site according to user-provided weights for each atom,
      which must be normalized within a site.
```

Note that the choice of mapping type here will impact what information is read from entries in the ’atomtypes’ directive.

2.2.3 [atomtypes]

```
[ atomtypes ]
; index   type   site   mass/weight
   1       O      1W    15.999
   2       H      1W    1.008
   3       H      1W    1.008
```

In the ’atomtypes’ directive, you will specify the atoms in the atomistic model and which site they map to. The index column corresponds to the index of that atom in the corresponding atomistic topology, allowing the mapping program to identify it in the atomistic trajectory. The type column is used for internal labeling and ease of visually reading the mapping topology file. The site column contains the name of a site from the ’sitetypes’ directive, or the word 'none' if the atom is not used to determine the position of a CG site. An example at the end of this section will show the use of the 'none' keyword in a mapping topology file for toluene. The contents of the mass/weight column depend on the choice of mapping type for the site the atom belongs to. For a 'com' mapping, this column will contain the mass of the atom in amu. For a 'usr' mapping type, this column will contain a user-specified mapping weight. Note that the sum of these user-specified weights must sum to one for the atoms in a single site. For a 'cog' mapping type, the mass/weight column is ommitted.

2.2.4 [molecules]

```
[ molecules ]
; name  number
  H2O   216
```

In the 'molecules' directive, you will specify the composition of the system. The name column will contain the name of a molecule contained in a previous 'moleculetype' directive, and the number column will contain the number of that type of molecule in the system.

2.2.5 Multiple molecules in a system

Many systems of interest will contain more than one type of molecule. For such systems, additional 'moleculetypes' directive specify the different molecule types present in the system, which each require their own 'sitetypes' and 'atomtypes' directives. All molecule types must be enumerated under the 'molecules' directive.

Below, we have provided an example of a mapping topology file for a more complex system, composed of 221 3-site heptane molecules and 331 3-site toluene molecules. Note that each 'moleculetype' directive is followed immediately by a 'sitetypes' directive and an 'atomtypes' directive corresponding to mapping information for that molecule. Additionally, the order in which the molecule types are defined in the file corresponds to the order of molecules in the atomistic trajectory - the first 221 molecules are heptane, and the last 331 molecules are toluene. This is a requirement for the calculation to proceed correctly. Finally, note that the last directive in the file is a singel 'molecules' directive that specifies the number of both heptane and toluene molecules within the system, again in the same order as the atomistic trajectory file.

```
[ moleculetype ]
; name    n_atoms
  HEP     23

[ sitetypes ] 3
; name   mapping_type  n_atoms
  CTA    com           7
  CM     com           9
  CTB    com           7

[ atomtypes ]
; index    type    site    mass/weighting
    1       C       CTA     12.011
    2       H       CTA     1.008
    3       H       CTA     1.008
    4       H       CTA     1.008
    5       C       CTA     12.011
    6       H       CTA     1.008
    7       H       CTA     1.008
    8       C       CM      12.011
    9       H       CM      1.008
    10      H       CM      1.008
    11      C       CM      12.011
    12      H       CM      1.008
    13      H       CM      1.008
    14      C       CM      12.011
    15      H       CM      1.008
    16      H       CM      1.008
    17      C       CTB     12.011
    18      H       CTB     1.008
    19      H       CTB     1.008
    20      C       CTB     12.011
    21      H       CTB     1.008
    22      H       CTB     1.008
    23      H       CTB     1.008

[ moleculetype ]
; name    n_atoms
  TOL    15

[ sitetypes ] 3
; name   mapping_type  n_atoms
  CF     com           5
  CBA    com           1
  CBB    com           1

[ atomtypes ]
; index    type    site    mass/weighting
    1       C       CF      12.011
    2       H       CF      1.008
```

```
        3       H       CF      1.008
        4       H       CF      1.008
        5       C       CF      12.011
        6       C       none    12.011
        7       H       none    1.008
        8       C       none    12.011
        9       H       none    1.008
        10      C       CBA     12.011
        11      H       none    1.008
        12      C       CBB     12.001
        13      H       none    1.008
        14      C       none    12.011
        15      H       none    1.008

[ molecules ]
; name number
HEP 221
TOL 331
```

3 CGFF

The CGFF tool derives a CG force field from a CG trajectory using force matching and/or the g-YBG method.

3.1 Input Files

The main input file for the CGFF code (which MUST be named par.txt) contains a list of directives and associated parameters. All other input files are dependent (either directly or indirectly) on the information placed in par.txt. Below is a list of all the directives currently available for specificying input options. Some directives are mandatory, while others are optional (as indicated below). The directive name must be placed inside square brackets, [directive_name], followed by the appropriate options and an end directive line, [End directive_name]. (Examples will be given below).

3.2 List of Directives

1. Mode
2. Temperature
3. Structures
4. Site_Types

5. Inter_Types
6. Pair_Interaction
7. BondStretch
8. Angle
9. Dihedral
10. IntraMolec_NB_Pair
11. Reference_Potential
12. Perturbation_Theory
13. Eigen
14. SVD
15. TPR
16. TOP_EXCL
17. Metric_Tensor
18. Mean_Force_Decomposition
19. Calculation_Mode
20. Preconditioning
21. Solution_Method
22. Calculate_Errors
23. Reference_Options
24. TRIM
25. CHISQD
26. Regularization
27. Rescale_Forces
28. Constrain_Dihedrals
29. Iterative_gYBG

3.3 Mandatory Directives

3.3.1 Mode

Syntax:

```
[Mode] MODE
```

MODE = “GROMACS”: Use gromacs .trr and .tpr files to obtain structural and force (if applicable) information for the calculation. In this mode, the .tpr file will be used to set up the CG topology. So, the interactions in your .top file must match up with those listed in par.txt (otherwise, the program will terminate and return errors).

Historically, a PDB mode was supported. However, due to the compatibility of PDB files with standard GROMACS tools, this mode was dropped. This directive remains as a stub for file formats that may be supported in the future.

3.3.2 Temperature

Syntax:

```
[Temperature] T
```

T is the temperature at which the structures were generated as a floating point number. T is used to calculate each $b_\zeta(x) = r_m k_B T(\frac{d}{dx}\bar{g}_\zeta(x) - L_\zeta(x))$ from structural information. The units of T determine the units of phi_ζ and b_ζ in the output (units of Kelvin correspond to standard Gromacs units).

3.3.3 Structures

Syntax:

```
[Structures] N_Structure_Files
filename1
filename2
...
...  x N_Structure_Files
...
[End Structures]
```

Each filename under the Structures directive should contain a list of filenames that store the necessary structure and force information (if applicable) for the calculation. For example, in Gromacs mode the structure file contains one or more .trr filenames. For most calculations, one structure file will be most convenient (we usually name this file inp.txt for consistency). Here is the syntax of the structure file:

```
[Struct_Files] N_files
filename1    w1
filename2    w2
...
...  x N_Files
...
[End Struct_Files]
```

The w_i after each filename is a weighting term that will be applied to the information in that file in the FF calculation. The weights of all structure files given should sum to 1.0 (If they don't a warning/error message will be printed). If MODE = Gromacs (as described above), then each filename corresponds to a .trr file. The .tpr filename which is also required is not explicitly specified. Instead it is infered from the .trr filename in the structure file (i.e., the .trr and .tpr filenames must be the same). There is a way to overwrite these default .tpr filenames (see the TPR directive below).

3.3.4 Site_Types

Syntax:

```
[Site_Types] N_Site_Types
Type1
Type2
...
...  x N_Site_Types
...
[End Site_Types]
```

Typei corresponds to the name of a CG site type. These must match up with the names in the .tpr file (if in Gromacs Mode) or .pdb file (if in PDB Mode).

3.3.5 Inter_Types

Syntax:

```
[Inter_Types] N_Inter_Types
inter_name1   inter_type1   basis1   basis1_options
inter_name2   inter_type2   basis2   basis2_options
...
...  x N_Inter_Types
...
[End Inter_Types]
```

inter_name: the name of a CG interaction, specified by the user. This name will appear in the naming of relevant output files.

inter_type : the type of each interaction, defined by the scalar order parameter which characterizes the interaction.

- **Pair_Interaction** a nonbonded pair interaction
- **BondStretch** an intramolecular bond
- **Angle** an intramolecular angle
- **Dihedral** an intramolecular dihedral
- **IntraMolec_NB_Pair** an intramolecular pair interaction

basis_type and **basis_options**: Each basis has different options which must be specified. The options should follow the basis name as shown above, be in the order shown below, and contain only blank spaces between them. There are specific options available for inter_type: "Pair_Interaction", "BondStretch", "Angle", "Dihedral", or "IntraMolec_NB_Pair".

- **delta** represents the interaction on a grid using constant values at each gridpoint and can be applied to all inter_types.
  - **dr** the grid spacing as a floating point number
  - **r_min** the minimum grid value as a floating point number
  - **r_max** the maximum grid value as a floating point number
  - **n_smooth** smooth the b_ζ's by a running average with this many points.
- **linear** represents the interaction on a grid using linear interpolation between gridpoints and can be applied to all inter_types.
  - **dr** the grid spacing as a floating point number
  - **r_min** the minimum grid value as a floating point number
  - **r_max** the maximum grid value as a floating point number
  - **n_smooth** smooth the b_ζ's by a running average with this many points.
- **Bspline** represents the interaction on a grid using k^{th} order Bspline interpolation between gridpoints and can be applied to all inter_types.
  - **dr** the grid spacing as a floating point number
  - **r_min** the minimum grid value as a floating point number
  - **r_max** the maximum grid value as a floating point number
  - **k** the order of the spline (k=4 ⇒ cubic Bspline), (e) n_smooth - smooth the b_ζ's by a running average with this many points.
  - **n_smooth** smooth the b_ζ's by a running average with this many points.

- **power** represents the interaction as a sum of Lennard-Jones-type functions. Only valid for Pair_Interaction types
  - **r_max** the cutoff distance for nonbonded pair interaction as a floating point number
  - **N_powers** the number of $\frac{1}{r^n}$ functions to use
  - **m**
  - **...** - the specific power for each function, from smallest to largest
  - **n**
- **harmonic** represents the interaction as a harmonic function with two paramters. Only valid for BondStretch and Angle types.
- **RB** (Ryckaert-Bellman): represents the interaction as a Ryckaert-Bellman cosine series. Only valid for Dihedral types.
- **TOY** represents the interaction as a different cosine series. Only valid for Dihedral types.

3.4 Dependent Directives

These directives are not mandatory in general, but become mandatory based on options specified in other directives.

3.4.1 Pair_Interaction

Syntax:

```
[Pair_Interaction] N_Pair_Types
inter_name1   type1a   type1b
inter_name2   type2a   type2b
...
...  x N_Pair_Types
...
[End Pair_Interaction]
```

This directive is mandatory when there are Pair_Interaction types specified under the Inter_Types directive. Each inter_name corresponds to one of the interaction names under the Inter_Types directive. Each type corresponds to one of the CG site types under the Site_Types directive. A single interaction type may be listed more than once in order to combine several pairs of site types into a degenerate interaction type.

3.4.2 BondStretch

Syntax:

```
[BondStretch] N_Bond_Types
inter_name1   type1a   type1b
inter_name2   type2a   type2b
...
...  x N_Bond_Types
...
[End BondStretch]
```

This directive is the same as the Pair_Interaction directive, except for BondStretch types. The pair of site types must define a bond type in the .tpr file if in Gromacs mode. Additionally, each bond type in the .tpr file must be represented here. (Note: This implies that advanced options are needed for including BondStretch reference potentials. See [Reference_Potential] and [TOP_EXCL])

3.4.3 Angle

Syntax:

```
[Angle] N_Angle_Types
inter_name1   type1a   type1b   type1c
inter_name2   type2a   type2b   type2c
...
...  x N_Angle_Types
...
[End Angle]
```

This directive is the same as the BondStretch directive, except for Angle types.

```
\subsubsection{Dihedral}
\vb
\noindent Syntax:
\vb
[Dihedral] N_Dihedral_Types
inter_name1   type1a   type1b   type1c   type1d
inter_name2   type2a   type2b   type2c   type2d
...
...  x N_Dihedral_Types
...
[End Dihedral]
```

This directive is the same as the BondStretch directive, except for Dihedral types.

3.4.4 IntraMolec_NB_Pair

Syntax:

```
[IntraMolec_NB_Pair] N_Types
inter_name1   type1a   type1b
inter_name2   type2a   type2b
...
...  x N_Types
...
[End IntraMolec_NB_Pair]
```

3.5 Optional Directives

These directives are completely optional. In a few instances they may affect the functionality of one another, but are never needed based on the directives already discussed.

3.5.1 Reference_Potential

Syntax:

```
[Reference_Potential]  ref.txt
```

ref.txt is the path to a file which contains the information about reference contributions which will be subtracted out during the calculation. This method is a bit out dated. We have found that the accuracy of this method is lacking, especially for bonded interactions with many large forces being subtracted from the mean force on each site. Instead, we now implement reference forces by using the -mdrerun option in gromacs to calculate a .trr file with reference contributions on each site. We then read in this trajectory and evaluate the reference contribution to the **b** vector before subtracting from the total contribution to **b**. Options for these calculations can be found under [Reference_Options].

3.5.2 Calculation_Mode

Specifies what part(s) of the calculation to run.

Syntax:

```
[Calculation_Mode]
CalcMODE
[End Calculation_Mode]
```

CalcMODE the calculation mode to perform. The options are case sensitive.

- **FULL** (default) Runs the entire calculation from beginning to end. Save state files will be printed out so additional calculations may be done without re-calculating the correlation functions.
- **FIRST_HALF** Reads input, calculates correlation functions, prints out save state files for later calculations.
- **SECOND_HALF** Reads input to set up system structure, reads in save state files, trims the necessary arrays, and then performs the matrix inversion according to the input options. Note that the matrix inversion is serial, so you should never submit a job with CalcMODE = "SECOND_HALF" on multiple processors.
- **TEST_INP** Reads input, calculates the number of force field parameters and then estimates the amount of memory you will need for the first half, second half, and full calculation. The memory estimation takes into account the optional input parameters in par.txt as well as the number of parameters. This memory estimation should be updated based on the new regularization and constraint algorithms.

3.5.3 TPR

Syntax:

```
[TPR] N_TPR_Files
filename1
filename2
...
...  x N_TPR_Files
...
[End TPR]
```

Without this directive, the program expects a .tpr file for each .trr file listed in the structure files ([Structures]). These .tpr files must have the same name and path as the .trr files. This directive overwrites the default .tpr file paths with those specified here. The number and order of the .tpr files must match up exactly with the .trr files. The program will give an error message if you input the wrong number of files, but there is nothing to check that the order is correct, so be careful!

3.5.4 TPR_EXCL

This directive allows the user to pass in a tpr file to be used for pair exclusions only. This is useful for passing in a bond reference force without force-matching the corresponding interaction.

Syntax:

```
[TPR_EXCL]
filename.tpr
[End TPR_EXCL]
```

filename.tpr denotes the name of the tpr file which will be used for calculating pair exclusions. Without this directive, the usual tpr files will be used instead.

3.5.5 Solution_Method

This directive specifies the solution method for solving the system of linear equations.

Syntax:

```
[Solution_Method]
SOLN_METH
[End Solution_Method]
```

SOLN_METH specifies the solution method to be used. The input is case sensitive

- **SVD** (default) Solves the matrix inversion using single value decomposition.
- **CHOLESKY** Solves the matrix inversion using CHOLESKY decomposition (for symmetric positive-definite matrices). This method utilizes packed storage of the symmetric matrix in order to save memory.
- **UU** Solves the matrix problem using LU-type decomposition while also utilizing the packed storage form of the matrix.
- **LU** Solves the matrix problem using general LU decomposition. Consequently, simple preconditioning can be performed on the matrix (see [Preconditioning]).

3.5.6 Error_Estimates

If this directive is present, error estimates will be calculated for the CG force field obtained from the method chosen in [Solution_Method]. The inverse condition number will be approximated from the corresponding decomposition of **G** (Note: You can calculate the condition number more precisely with [SVD] or [Eigen]). Iterative refinement will also be used to estimate the forward and backward errors.

Syntax:

```
[Error_Estimates]
flag_Equil
[End Error_Estimates]
```

flag_Equil "YES" or "NO". The matrix will be equilibrated (i.e., preconditioned) by LAPACK if flag_Equil is YES.

3.5.7 SVD

This directive provides additional information about the single value decomposition calculation.

Syntax:

```
[SVD]
rcond
flag_printSV
flag_printevecs
flag_solve
[End SVD]
```

rcond a floating point value in decimal form. Each inverse singular value below this number will be set to zero before calculating the solution vector. This is only applicable when flag_solve is "YES".

flag_printSV 'YES" or "NO". The singular values will be explicitly calculated and printed out if flag_printSV is "YES".

flag_printevecs "YES" or "NO". The eigenvectors corresponding to each singular value will be explicitly calculated and printed out if flag_printevecs is "YES".

flag_solve "YES" or "NO". If flag_solve is "YES", the matrix problem is solved using singular value decomposition. This flag is equivalent to specifying the solution method "SVD" in [Solution_Method] (and specifying this directive will overwrite whatever option is specified under [Solution_Method]).

3.5.8 Preconditioning

This directive applies to calculations with SOLN _METH = LU. Additionally, if [Regularization], [Constrain_Dihedrals], or [SVD] is specified, then the option under [SOLN _METH] is overwritten such that this directive is applicable. Generally, RPC_type = LPC_type = "dimless" with flag_normb = flag_normphi = "NO" yields the lowest condition number.

Syntax:

```
[Preconditioning]
RPC_type
flag_normb
LPC_type
flag_normphi
[End Preconditioning]
```

RPC_type specifies the right preconditioning method

- **NO** Do not precondition **G**.
- **dimless** Precondition such that **G** is dimensionless. For this to work, "dimless" must be chosen for both RPC_type and LPC_type. To use the rcond option under [SVD], this option must be used.
- **colnorm** divide each column of **G** by the norm of that column
- **MTvar** divide each column of **G** by the sum of the elements in the corresponding column of $var(\mathbf{G})$

flag_normb "YES" or "NO". "YES" rescales the preconditioning terms such that b retains the same norm before and after preconditioning.

LPC_type specifies the left preconditioning method

- **NO** Do not precondition **G**.
- **dimless** dimless: Precondition such that **G** is dimensionless. For this to work, "dimless" must be chosen for both RPC_type and LPC_type. To use the rcond option under [SVD], this option must be used.
- **rowmax** divide each row of **G** by the absolute value of the max of the elements in the row
- **bvar** divide each column of **G** by the sum of the elements $var(b)$

flag_normphi "YES" or "NO". "YES" rescales the preconditioning terms such that ϕ retains the same norm before and after preconditioning.

3.5.9 Iterative_Inversion

This directive applies a matrix perturbation technique to iteratively calculate the solution to the force-matching equations.

Syntax:

```
[Iterative_Inversion]
N_PT
spacing
flag_MMOTF_SEP
flag_eigen
N_eigen
[End Iterative_Inversion]
```

N_PT the number of perturbation steps to perform.

spacing the number of perturbation steps between printing of the solution.

flag_MMOTF_SEP "YES" or "NO". If "YES", the solution at each (printed) step will be multiplied by the full metric tensor, yielding an estimate of the structural correlation functions determined by the present force field.

flag_eigen "YES" or "NO". If "YES", the eigenvalues and eigenvectors of the current approximation to the metric tensor will be calculated for each (printed) step.

N_eigen determines the number of eigenvectors to print out from each side of the spectrum (i.e., N_eigen = 10 will print the ten eigenvectors corresponding to the lowest eigenvalues and the ten eigenvectors corresponding to the highest eigenvalues).

3.5.10 Eigen

This directive allows the user to calculate and print the eigenvalues/eigenvectors of the metric tensor.

Syntax:

```
[Eigen]
N_Eigen
flag_fnbnMn
spacing_top
spacing_bottom
flag_Gbar
flag_norm
[End Eigen]
```

N_Eigen determines the number of eigenvectors to print out from each side of the spectrum (See N_eigen under [Iterative_Inversion]).

flag_fnbnMn "YES" or "NO". If "YES", the eigenvector representation of the forces, b's, and the metric tensor will be calculated for various sets of eigenvectors (according to the options below).

spacing_top A floating point number that determines the grouping of eigenvectors from the "top" (highest eigenvalues) for calculating the representation of the forces, b's, and the metric tensor.

spacing_bottom A floating point number that determines the grouping of eigenvectors from the "bottom".

flag_Gbar "YES" or "NO". If "YES", the eigenvalues/eigenvectors will be calculated for $\bar{\mathbf{G}}$.

flag_norm "YES" or "NO". This flag controls whether or not the metric tensor is normalized by $\frac{1}{r_{\zeta'}(x)^2 r_{\zeta'}(x')^2}$ (see [Metric_Tensor]) before calculating the eigenvalues/eigenvectors.

3.5.11 Metric_Tensor

This directive is concerned with the calculation and print options of the metric tensor and related quantities.

Syntax:

```
[Metric_Tensor]
flag_print
flag_norm
flag_Mcnt
[End Metric_Tensor]
```

flag_print "YES" or "NO". This flag controls whether or not the metric tensor blocks are printed out for analysis purposes. (see Rudzinski, Noid JPCB 2012).

flag_norm "YES" or "NO". This flag controls whether or not the metric tensor blocks (printed out for analysis purposes) are normalized by $\frac{1}{r_{\zeta}(x)^2 r_{\zeta'}(x')^2}$ (see Rudzinski, Noid JPCB 2012).

flag_Mcnt "YES" or "NO". This flag determines whether or not $\mathbf{P}_{\zeta\zeta'}(x, x')$ is calculated or not. Note that flag_Mcnt = "NO" will probably decrease the amount of memory needed for the calculation (See CalcMODE = TEST_INP).

3.5.12 Mean_Force_Decomposition

If this directive is provided in par.txt, the mean force will be decomposed into 2- and 3-body contributions for each interaction type in addition to the normal force field output.

Syntax:

```
[Mean_Force_Decomposition]
[End Mean_Force_Decomposition]
```

3.5.13 Reference_Options

This directive has options for calculating or reading in a reference b vector, bref.

Syntax:

```
[Reference_Options]
flag_calcbref
flag_readbref
flag_splitfiles
flag_breftrr
breftrr_fnm
[End Reference_Options]
```

flag_calcbref 'YES" or "NO". If "YES", the triple loop will be skipped and the calculation will be terminated once bref is calculated and printed to bref.dat

flag_readbref "YES" or "NO". If "YES", the calculation will proceed as normal, but the values in bref.dat will be subtracted from the calculated b's. Note that this will replace subtraction of the normal bref if a reference potential is supplied. Also, if flag_calcbref = "YES", this option does nothing.

flag_splitfiles "YES" or "NO". If "YES", each processor will handle a separate file (instead of splitting up the frames in each file. This is more efficient than the normal way, but requires that you split up your trajectory file by hand. Note that for nproc should divide nfiles. This option was motivated by the situation where you have an extremely large trajectory file and the interactions you are parameterizing are contained only in a subset of the molecules in the system (with reference potentials for the other molecules). In this case, one could precompute bref with flag_calcbref = "YES" and flag_splitfiles = "YES" for maximum efficiency.

flag_breftrr "YES" or "NO". If "YES", reference forces will be taken from a trr file named breftrr_fnm.

breftrr_fnm The file containing reference forces for flag_breftrr. This file should be generated with mdrun -rerun with the reference topology.

3.5.14 TRIM

This directive has options for trimming the basis vectors that are rarely sampled. FE $\approx$ 0.005 - 0.05 works well in practice, depending on the system. The details of this trimming are explained in detail in the SI section of Rudzinski, Noid JPCB (2014).

Syntax:

```
[TRIM]
FE
[End TRIM]
```

FE the sampling cutoff for trimming rarely sampled vectors, as a decimal % of the sampling expected per bin in a uniformly sampled interaction

3.5.15 CHISQD

This directive allows for the calculation of χ^2 for an arbitrary force vector. You can input a force vector contained in the file named force_nm. This vector should be consistent with the dimensions of the problem from the normal calculation and the save state input files.

Syntax:

```
[CHISQD]
force_fnm
[End CHISQD]
```

force_fnm The location of a file containing a complete force vector

3.5.16 Regularization

This directive has options for regularizing the system of linear equations that are solved during the force field calculation.

Syntax:

```
[Regularization]
type
Nmax
tau_alpha
tau_beta
[End Regularization]
```

type the method of regularization

> **BAYES** regularizes all the force field coefficients using the Bayesian inference method (see Liu,...,Voth JCP 2008)
>
> **UNCERT** regularizes the force field coefficients corresponding to nonbonded and intramolecular nb coefficients in a simple way by estimating the uncertainty in the columns of the metric tensor (see Rudzinski, Noid JPCB 2014)

Nmax the number of maximum iterations for optimizing the regularization parameters

tau_alpha the tolerance for convergence for the regularization matrix parameters as a floating point number. This option is only relevant for "BAYES".

tau_beta is the tolerance for convergence for the precision parameter as a floating point number. This option is relevant for both regularization methods.

3.5.17 Constrain_Dih

This directive allows one to constrain the coefficients of each dihedral interaction to sum to 0. This requires that the dihedral force integrates to zero, which is necessary for the dihedral potential to be periodic. This directive is only useful for dihedral force functions represented with a tabulated basis and which sample all grids betwen -180 and 180 during the reference simulation.

Syntax:

```
[Constrain_Dih]
Nmax
tau
L
dL
[End Constrain_Dih]
```

Nmax specifies the maximum number of iterations in the constraint algorithm.

tau specifies the constraint parameter τ as a floating point number

L specifies the tolerance as a floating point number

dL specifies the change in tolerance during each iteration as a floating point number

3.5.18 Iterative_gYBG

This directive has options for performing the iter-gYBG method as described in Ruzinski, Noid JPCB 2014.

Syntax:

```
[Iterative_gYBG]
flag_AAM2
AAM2_fnm
flag_bsolnerr
[End Iterative_gYBG]
```

flag_AAM2 “YES” or “NO”. “YES” allows the user to provide a file with the 2-body contributions to the metric tensor from the AA trajectory. This is necessary for the modified iter-gYBG method proposed by Ruzinski, Noid JPCB 2014. “NO” turns off this directive.

AAM2_fnm the name of the file with the AA 2-body contributions when flag_AAM2 = “YES”.

flag_bsolnerr “YES” or “NO”.“YES” prints out the error in the projected force functions from consecutive iterations. With this option, the force files from the previous iterations must be provided.

3.6 Example par.txt and inp.txt files

As an illustrative example, we present the par.txt and inp.txt files from two different calculations. The first of these is for a calculation to recover the potentials in a simple Lennard-Jones fluid. This system has a single type of nonbonded pair interaction between LJ sites, and no intramolecular interactions. Note that in both the par.txt and inp.txt files, lines beginning with a '!' character are comment lines and are not read by the cgff tool.

Lennard-Jones par.txt

```
!# input file
[Mode] GROMACS

[Temperature] 298

[Structures]    1
inp.txt
[End Structures]

[Site_Types]    1
LJ
[End Site_Types]

[Inter_Types]  1
!inter_name inter_type        basis   dr     R_min  R_max  n_smooth
 LJLJ       Pair_Interaction  delta   0.004  0.000  0.900  5
[End Inter_Types]

[Pair_Interaction] 1
! inter_name   type1  type2
  LJLJ          LJ     LJ
[End Pair_Interaction]

[TPR] 1
../LJ.tpr
[End TPR]
```

Lennard-Jones inp.txt

```
! comment line
[Struct_Files]   1
../LJ.trr    1.0
[End Struct_Files]
```

The next example is for an extended ensemble calculation, over three alkane trajectories: butane, heptane, and decane. These files correspond to the calculation performed in the code release paper: [citation].

Extended Ensemble Alkanes par.txt

```
!# input file
[Mode] GROMACS

[Temperature]  298.0

[Structures]    1
inp.txt
[End Structures]

[Site_Types]    2
CT
CM
[End Site_Types]

[Inter_Types] 8
!inter_name     inter_type        basis     dr        R_min  R_max k   n_smooth
 CTCT           Pair_Interaction  Bspline   0.02      0.000  1.400 4   0
 CMCM           Pair_Interaction  Bspline   0.02      0.000  1.400 4   0
 CTCM           Pair_Interaction  Bspline   0.02      0.000  1.400 4   0
!inter_name     inter_type        basis      dr    R_min  R_max n_smooth
 Bond_CT-CM     BondStretch       linear     0.001 0.000  0.400 0
 Bond_CT-CT     BondStretch       linear     0.001 0.000  0.400 0
 Bond_CM-CM     BondStretch       linear     0.001 0.000  0.500 0
 Angle_CT-CM-CT Angle             linear     0.500 0.000  180.0 0
 Angle_CT-CM-CM Angle             linear     0.500 0.000  180.0 0
[End Inter_Types]

[Pair_Interaction] 3
! inter_name   type1  type2
  CTCT           CT     CT
  CMCM           CM     CM
  CTCM           CT     CM
[End Pair_Interaction]

[BondStretch]         3
!inter_name    site1          site2
 Bond_CT-CM          CT             CM
 Bond_CT-CT          CT             CT
 Bond_CM-CM          CM             CM
[End BondStretch]

[Angle]        2
```

```
!site1     site2     site3     type
 Angle_CT-CM-CT    CT          CM        CT
 Angle_CT-CM-CM    CT          CM        CM
[End Angle]

[TPR] 3
map_butane/cgbutane.tpr
map_heptane/cgheptane.tpr
map_decane/cgdecane.tpr
[End TPR]
```

Extended Ensemble Alkanes inp.txt

```
! comment line
[Struct_Files]   3
map_butane/cgbutane.trr    0.33
map_heptane/cgheptane.trr  0.33
map_decane/cgdecane.trr    0.33
[End Struct_Files]
```

4 Tables

The tables tool creates GROMACS table files for non-bonded and bonded interactions. The bonded interactions include bond-stretches, bond-angles, and bond-dihedrals. The input consists of a file that contains a tabulated force. The independent variable (e.g. distance or angle) must be evenly spaced. The program will quit if this condition is not met. Additionally, for angle and dihedral forces the end points must be consistent with GROMACS requirements which are specified below in the appropriate section. The tabulated forces should already be smooth and any unphysical data should be removed before using this program. Additional parameters are required for certain interaction types as indicated below. The type of interaction is specified at the command line.

4.1 Non-bonded Interactions

For non-bonded interactions, the maximum distance for writing the table is required as input. The force is linearly extrapolated into the hard-core region with a slope determined from the first two points in the input file. It is assumed that the force already converges to zero at the cutoff. The force is saved as zero for points not listed but within the maximum distance specified on the command line.

4.2 Bonded Interactions

4.2.1 Bond Stretch

The minimum and maximum distance for the bond stretch must be specified at the command line. These should include any possible bond length that may be sampled in the simulation. The force curve will be extrapolated in both directions from the minimum and maximum distances in the input file until the minimum and maximum distance from the command line is reached. The extrapolation is linear with a slope that is specified at the command line. The user should make wise choices. The intent is that outside the range given in the input file, the potential becomes harmonic and should push the bond back to distances given in the input file. The minimum and maximum distances specified at the command line should included any possible bond lengths sampled during a simulation. GROMACS will complain if this is not satisfied.

4.2.2 Bond Angle

Only the slope for linearly extrapolating the input force is require at the command line. GROMACS requires that the force begin and end at 0.0 degrees and 180.0 degrees, respectively. The program will complain if the angle variable cannot be extended in either direction with the grid spacing and match these end-point conditions. It should be noted that GROMACS expects a force taken as the derivative in degrees of the specified potential, so the input should be provided appropriately. The program will extrapolate linearly with the specified slope from the minimum and maximum input angles to 0.0 and 180.0 degrees. As with the bond stretch, this becomes a harmonic potential at the extremes to push the angle back to the values specified in the input file.

4.2.3 Bond Dihedral

No special command line parameters are specified for the dihedral interactions. However, the input must have first and last angles of -180.0 and 180.0, respectively. Also, as with angle, everything is specified in terms of degrees.

4.3 Usage

```
tables [infile] [outfile] [type] [basis] (rmin) (rmax)
```

Infile a two-column file containing evenly spaced distance (or angle) and force pairs

Outfile the name of the destination table file

Type Specifies the type of the interaction.

> **nb** nonbonded interactions (requires rmin argument)
>
> **bond** bond stretches (requires rmin, rmax argument)

angle angle bending interactions

dihedral dihedral interactions

Basis The basis type used for force-matching the interaction

delta

linear

Bspline

4.4 Examples

Nonbonded Lennard-Jones interaction with a linear basis:

```
tables LJLJ_force.dat table_LJLJ.xvg nb linear 6.00
```

CT-CT bond stretch with a delta basis:

```
tables CTCT_bond_force.dat table_b0.xvg bond delta 0.2 0.5
```

CT-CM-CT angle bending with a delta basis:

```
tables CTCMCT_angle_force.dat table_a0.xvg angle delta
```

CA-CB-CC-CD dihedral torsion with a Bspline basis:

```
tables CA-CB-CC-CD_dih_force.dat table_d0.xvg dihedral Bspline
```

4.5 Converting to LAMMPS tables

In order to facilitate use of these CG force fields with LAMMPS, a convert_tables.py script is included in the scripts folder for conversion of GROMACS table files to LAMMPS table files. The output assumes the use of 'real' units in the LAMMPS simulation. This usage format for this script is as follows:

```
convert_tables.py [gromacs_tables] [table_type] [lammps_table] [interaction_name]
```

gromacs_table The location of a GROMACS table file to be converted

table_type The type of interaction contained in the table file, selected from the following:

nb specifies a nonbonded interaction table

bond specifies a bond stretch table

angle specifies an angle table

dih specifies a dihedral table

lammps_table The name of the LAMMPS table to be written

interaction_name The name of the interaction within the LAMMPS table file. This identifies the specific interaction to LAMMPS, in the event that you want to have multiple interactions within a single file.

5 PMatch

5.1 Introduction

This document outlines the different filetypes that are used in the current pressure-matching implementation.

5.2 Input file: *.ini

The name of this file is specified at runtime as a command line argument of the form:

```
p_match config.ini
```

For the purposes of this document we will call the input file 'config.ini', but you are free to name it anything you would like. This feature is included so that multiple configuration files can coexist within one directory. Note that you would need to specify different output filenames for each configuration file to avoid overwriting your results. The settings in this file are all specified in the format

```
setting: value
```

Note that the colon is a required part of the formatting. The different settings and acceptable values are listed below. The order in which the settings are listed in the file does not matter.

5.2.1 Universal settings

These settings are required for every pressure matching calculation. Depending on the values chosen for these settings, more settings may be invoked. In cases where this occurs, the extra settings are listed under the value of the original setting that invokes them.

`Pressure_units`

- **md** Pressures are described in kJ per (mol*nm^3), the internal units of pressure in Gromacs
- **bar** Pressures are described in bar, the output units of pressure in Gromacs

`Volume_units`

- **nm3** Volumes are described in nm^3, the internal and output units of volume in Gromacs
- **liter** Volumes are described in liters

`Energy_units`

- **kj_per_mol** Energies are described in kJ/mol, the internal and output units of energy in Gromacs

`Basis_type`

das_andersen Represents the pressure correction as a Taylor series expansion as detailed in MSCG V by Das and Andersen. Invokes the following options:

N_pres_coeff The number of terms in the Taylor series to use

delta Represents the pressure correction on a decoupled grid. Invokes the following options:

vmax The high end of the volume grid

vmin The low end of the volume grid

dv The grid spacing

frac_cutoff The minimum fraction of a uniformly sampled bin to accept as sampled

`Iterative`

0 Performs a calculation with a 1:1 mapping of the AA and CG pressures

1 Performs a calculation with different AA and CG pressure-volume trajectories. Invokes basis-type-specific options in the das_andersen basis type, and the following output options:

CG_Q_output - Location of the correlation matrix Q for the for of the CG data

AA_g_cnt_output - Location of the sampling distribution for the AA data

CG_g_cnt_output - Location of the sampling distribution for the CG data

`Use_reference_Fv`

0 Do not use a reference pressure correction.

1 Use a reference pressure correction that will be added to that calculated for the pressure-volume trajectories given as input. Invokes the following options:

ref_Fv_file Location of the file containing the reference pressure correction. The format of this file depends on the basis type.

`Error_estimate`

0 Solve the set of linear equations using dgesv from LAPACK.

1 Solve the set of linear equations using dgesvx from LAPACK, which provides an estimate of the forward and backward error of the fit, as well as the condition number of the problem.

`N_atoms` - The number of atoms in the atomistic system
`N_sites` - The number of sites in the coarse-grained system
`AA_pressures` - The location of the xvg file containing the atomistic pressures

CG_pressures - The location of the xvg file containing the coarse-grained pressures
AA_volumes - The location of the xvg file containing the atomistic volumes
CG_volumes - The location of the xvg file containing the coarse-grained volumes
log_file - The location of the text file that will be created or overwritten to contain the output log
psi_output - The location of the psi output file that will be created or overwritten to contain the parametrization of the pressure correction (the result of the calculation)
Q_output - The location of the Q output file that will be created or overwritten to contain the correlation matrix from the calculation
g_cnt_output - The location of the g_cnt output file that contains the number of samples per bin for the bins used in the solution

5.3 Example config.ini file

An example config.ini file is provided below for an iterative run of a small heptane/toluene mixture system:

```
#Calculation settings
N_atoms: 4455
N_sites: 891
N_pres_coeff: 2
Pressure_units: bar
Volume_units: nm3
Energy_units: kj_per_mol
Basis_type: das_andersen
vmax: 118
vmin: 110
dv: 0.1
Iterative: 1
frac_cutoff: 0.1
Error_estimate: 1
Use_ref_Fv: 1

#Input Files
AA_pressures: aa_pressures.xvg
CG_pressures: press.dat
AA_volumes: aa_volumes.xvg
CG_volumes: vol.dat

ref_Fv_file: ../iter0/psi.dat

#Output Files
pres_decompose: decompose.xvg
```

```
log_file: log.txt
psi_output: psi.dat
Q_output: Q.dat
CG_Q_output: CG_Q.dat
AA_Q_output: AA_Q.dat
g_cnt_output: g_cnt.dat
AA_g_cnt_output: AA_g_cnt.dat
CG_g_cnt_output: CG_g_cnt.dat
```

5.4 Output file: psi

`basis_type: das_andersen`

This filetype is used for both output of a das_andersen basis calculation, as well as the input to an iterative das_andersen calculation as the ref_Fv file. There are three headings that are expected in this file in any order:

[**AVG_AA_VOLUME**] - Followed by a number on the next line which will be read in as the double-precision average volume of the atomistic system

[**PSI_COEFF**] - Followed by an integer on the same line denoting how many of the next lines to read in as double-precision psi coefficients

[**N_SITES**] - Followed by an integer on the next line which will be read in as the number of sites in the CG system

An example of the contents of this type of file follows:

```
[AVG_AA_VOLUME]
44.367584
[PHI_COEFF] 2
-43.993848
-330.013730
[N_SITES]
250
```

`basis_type: (linear or delta)` This filetype is used for input and output of tabulated basis functions. The output is in two columns separated by a comma, where the first column is the volume and the second column is the pressure correction (and value of psi) at that volume.

5.5 Output file: Q

The correlation matrix Q is printed out as in a text file with n_basis columns and n_basis rows, with the row and column of each entry corresponding directly to the row and column of that entry in Q.

6 Modified LAMMPS

This tool is a modified version of LAMMPS 17-Jun13 that incorporates the F_V pressure correction term from Dunn and Noid JCP 2015 into the MTTK barostat. The purpose of these modifications is to add an extra term to the calculation of the pressure to enable simulation of CG models at the correct density. This software is largely unchanged from the reference version of LAMMPS, so nearly all of the documentation available at `http://lammps.sandia.gov/doc/Manual.html` applies to this tool, with the exception that it is only able to perform simulations that utilize pressure corrections.

Changes have been made to the following source files relative to the vanilla version of LAMMPS:

- fix_nh.cpp
- fix_nh.h
- compute_pressure.cpp
- compute_pressure.h
- compute.h

All of the changes have been signed by the developers initials (NJD). These modifications manifest in the barostat settings through the 'npt' fix. The default npt fix would have a line of the following form in the LAMMPS input script:

```
fix            1 all npt temp 303.0 303.0 100.0 cgiso 0.986 0.986 1000.0
```

The meaning of these settings is documented in the LAMMPS manual linked above.

The new features associated with the modified barostat enter into the input script as additional arguments at the end of the npt fix line. There are two input modes for F_V that are accepted. For both modes, the units of the input settings must be consistent with the units selected for your LAMMPS simulation.

6.1 Analytic F_V

This input mode for F_V is represented as a set of N coefficients for a Das & Andersen basis function. The full line for the npt fix would look something like this:

```
fix    1 all npt temp 303.0 303.0 100.0 cgiso 0.986 0.986 1000.0 \
       analytic 113606.3 1602 2  119273.87 -80079.04
#      type     v_avg    N    Nd psi1      psi2
```

The line break after the default LAMMPS settings is only added for clarity within this document. The new terms all follow the line break, and are labeled with the corresponding comment line. The analytic keyword specifies the Das & Andersen basis function for F_V and determines the other arguments that are required. The agruments that follow are the average volume of the reference AA system, the number of sites in this CG system, the number of terms in the Das & Andersen basis function to use, and a list of the Das & Andersen psi coefficients.

6.2 Tabulated F_V

The pressure correction can also be input in a tabulated form read in from a file. The fix npt line for this type of F_V input is also modified by additional terms that follow the default input:

```
fix              1 all npt temp 303.0 303.0 100.0 cgiso 0.987 0.987 1000.0 \
                 cubic_spline   delta_Fv.dat
#                interp_type    file

fix              1 all npt temp 303.0 303.0 100.0 cgiso 0.987 0.987 1000.0 \
                 linear_spline  delta_Fv.dat
#                interp_type    file
```

Here, we see two examples of the additional input required for a tabulated F_V: the interpolation type, and the file location. The specific choice of interpolation type determines how the program interpolates for sampled volumes between the supplied grid points. An example delta_Fv.dat is provided below for a small range of volumes. You should make sure that the delta_Fv.dat file you use spans well more than the range of volumes you plan to sample - the simulation will end with an error if the system samples a volume outside of the file's range.

```
# delta_Fv.dat
# vol  press    (make sure the units are correct)
50, -3064.57628
150, -3064.45344
250, -3064.33061
350, -3064.20778
450, -3064.08495
550, -3063.96212
650, -3063.83928
750, -3063.71645
850, -3063.59362
950, -3063.47079
1050, -3063.34796
1150, -3063.22512
1250, -3063.10229
1350, -3062.97946
1450, -3062.85663
1550, -3062.7338
```

```
1650, -3062.61096
1750, -3062.48813
1850, -3062.3653
1950, -3062.24247
2050, -3062.11964
2150, -3061.9968
2250, -3061.87397
2350, -3061.75114
2450, -3061.62831
2550, -3061.50548
2650, -3061.38264
2750, -3061.25981
2850, -3061.13698
2950, -3061.01415
```

7 Tutorials

7.1 Lennard-Jones Recovery Tutorial

In this first tutorial we will demonstrate how to use the XXX toolkit for one of the simplest possible cases: the recovery of the Lennard-Jones nonbonded pair interaction. No actual coarse-graining takes place for this process. Rather, the cgff tool is used to calculate the nonbonded force interaction given a simulated trajectory of Lennard-Jones particles.

This and all following tutorials assume that you are using the default name for all GROMACS and XXX executables, and that you have them in your PATH variable.

To minimize the size of files stored in the XXX toolkit distribution, we have not bundled any actual trajectory files. Rather, we provide the input files so that you can generate the trajectories for yourself. For this tutorial, the simulation is very quick.

Input files for this tutorial can be found in the `tutorials/lj_recovery` folder.

7.1.1 AA Simulation

Files:

- `LJ.mdp`
- `LJ.gro`
- `LJ.top`
- `LJ.ndx`

Providing detailed instructions on the use of GROMACS is beyond the scope of this tutorial. Refer to the documentation provided at http://manual.gromacs.org/documentation/

for the version of GROMACS you are running. This tutorial was developed using GROMACS 4.5.3.

Briefly, this simulation consists of 216 Lennard-Jones particles at constant volume and temperature, parameterized to reproduce the properties of liquid argon. Only a few configurations are needed for recovering this pair potential, so this simulation should run in under a minute on a single processor. Run the simulation with the following commands:

```
$ grompp -c LJ.gro -p LJ.top -f LJ.mdp -n LJ.ndx -o LJ.tpr
$ mdrun -s LJ.tpr -o LJ.trr -e LJ.edr
```

7.1.2 Force Matching

Files:

- `LJ.trr`
- `inp.txt`
- `par.txt`

Now that we have generated the trajectory file, we can proceed with the force matching calculation. This calculation requires two input files, par.txt and inp.txt. The par.txt file specifies the type of force matching calculation we are performing, and inp.txt indicates which trajectory file we are using as input. Refer to the user's' manual for a detailed description of each of these settings.

Briefly, we are force matching a single pair interaction named LJLJ, between LJ particles in the Lennard-Jones trajectory we just generated. This interaction is being fit with a linear basis with a grid spacing of 0.01 nm, from 0.2 nm to 0.9 nm.

You can run the force matching calculation using the following command. Again, this system is small and simple, so the calculation will proceed very quickly on a single processor.

```
$ cgff
```

As indicated in the user manual, running the cgff tool generates a set of output files. For your reference, a set of this output files has been included in the `tutorials/lj_recovery/results` folder. The most immediately interesting of these files is `f_forces.Pair_interaction.total.LJLJ.dat`, which contains the force matching result of the LJLJ interaction.

The full set of results files generated includes:

- `b_forces.Pair_Interaction.total.LJLJ.dat` - the projection of the force field, b, as calculated from the forces
- `f_forces.Pair_Interaction.total.LJLJ.dat` - the calculated force matched force
- `g_cnt.Pair_Interaction.total.LJLJ.dat` - a normalized histogram of sampling along the pair interaction

- g.Pair_Interaction.total.LJLJ.dat - the metric tensor matrix G
- L_coeff.Pair_Interaction.total.LJLJ.dat - the Laplacian contribution that incorporates the appropriate Jacobian factor

A set of save state files, save.*.dat, are also generated by this calculation. These filenames correspond to internal variables in the cgff tool, and are printed right before solving the set of linear equations to determine the force-matched force field. They can be used as input into a second half calculation for re-calculating the force field without looping over the trajectory again. (see user's manual).

7.2 1-Site Methanol Tutorial

In this next tutorial, we will perform some simple coarse-graining and calculate a force field for 1-site methanol. This model maps each methanol molecule to its center of mass, and has a single interaction type: the nonbonded pair interaction between methanol sites. Despite also having a single site type in the coarse-grained representation, this calculation is substantially more computationally demanding than the Lennard-Jones recovery tutorial. This is because this tutorial is actually performing a coarse graining operation, and thus needs a larger amount of sampling. Additionally, the CG methanol pair interaction has a longer range, so a larger box must be simulated to accommodate its range. This simulation therefore contains 968 methanol molecules, run for 100 ns. Files for this tutorial can be found in the tutorials/1-site_methanol folder.

7.2.1 AA Simulation

Files:

- methanol.gro
- methanol.mdp
- methanol.top
- methanol_oplsaa.itp

You will want to run this simulation on HPC resources if at all possible, as generating a smooth CG force curve for this system requires significant simulation. We recommend running the AA simulation step overnight on 32 cores. You can set up and run the simulation with the following commands:

```
$ grompp -f methanol.mdp -c methanol.gro -p methanol.top -o methanol.tpr
$ mpirun mdrun -s methanol.tpr -o methanol.trr -e methanol.edr
```

7.2.2 Mapping

Files:

- `methanol.trr`
- `methanol.tpr`
- `map.methanol.top`
- `cg_methanol.top`
- `cg_methanol.mdp`
- `cg_methanol.ndx`

Now that we have generated a set of atomistic configurations, we must map them to the corresponding coarse-grained representation. The file map.methanol.top contains the information necessary for mapping from the AA to the CG representation. The anatomy of a mapping topology file is discussed in more detail in the user's manual. Briefly, the AA representation contains 6 atoms, and this mapping file specifies that the CG site for each methanol molecule shall be placed at the center of mass of the molecule, for all 968 methanol molecules in the system.

The first step we must take is to make all of the molecules whole across periodic boundary conditions. The cgmap and cgff tools are not very clever with respect to periodic boundaries, so we must translate for them using the command:

```
$ trjconv -f methanol.trr -force -pbc whole -s methanol.tpr
  -o whole_methanol.tpr
```

The XXX toolkit uses the cgmap tool to transform configurations from the AA to the CG representations, and it requires a GROMACS topology file to write to a CG trajectory. This requires a CG configuration as a .gro file as an input. You can generate this gro file using the cgmap command:

```
$ cgmap -f whole_methanol.trr -p map.methanol.top -c cg_methanol.gro
```

The `cg_methanol.top` file is a normal GROMACS .top file for a system of 1-site molecules, except that all interaction functions must be set to 1. This is important later for the cgff tool to determine which interactions are present in your system. The `cg_methanol.mdp` must be a valid .mdp file, but its contents do not matter, as we will not be using the resulting .tpr file for any simulations. With these files, you can map the AA trajectory using the following commands:

```
$ grompp -f cg_methanol.mdp -p cg_methanol.top -n cg_methanol.ndx
  -c cg_methanol.gro -o cg_methanol.tpr

$ cgmap -f whole_methanol.trr -p map.methanol.top -s cg_methanol.tpr
  -o cg_methanol.trr
```

Mapping is a computationally simple operation, so this command takes only a minute or two to complete.

7.2.3 Force matching

Files:

- inp.txt
- par.txt
- cg_methanol.trr
- cg_methanol.tpr

Now, finally, we can perform some actual force matching. As before, the inp.txt and par.txt files (included in the tutorials/1-site_methanol/coarse_graining folder) specify the trajectory file and the type of force matching calculation we are performing. This par.txt file is similar to that for the Lennard-Jones recovery tutorial because both systems are 1-site per molecule in the CG model. This is another case where you will want to run the calculation on approximately 30 cores overnight. The command to run the calculation is:

```
$ mpirun cgff
```

This will generate an analogous set of files to those seen in the Lennard-Jones recovery tutorial.

7.2.4 Test Simulation

- f_forces.Pair_Interaction.total.MEOMEO.dat
- cg.sim.methanol.top
- cg.methanol.gro
- cg.sim.methanol.mdp
- cg.sim.methanol.ndx

Now that we have calculated the CG pair force via force matching, let's use it in a simulation. First, we must refine the output of the cgff tool for use in a simulation. This refining is done by the tables tool, and takes the form of extending the force back to zero, and out to the cutoff required by GROMACS, as well as changing to the required xvg file format.

```
$ tables f_forces.Pair_Interaction.total.MEOMEO.dat table_MEOMEO.xvg
  Bspline 6.00
```

The tables tool will complain if there are any unsampled bins in `f_forces.Pair_Interaction.total.MEOMEO.dat`. To address these, you may interpolate between points using a method of your choosing, or simply delete the bins up to the missing bin. Since unsampled bins typically occur at very short distances, we recommend simply deleting the bins up to the missing bin.

We must also prepare a tpr file for this simulation. Note that the mdp file specifies the MEO energy group, and the MEO MEO energy group table. This causes GROMACS to look for our `table_MEOMEO.xvg` file to define the MEO MEO pair interaction. Also note that the setup of the pair interaction in the cg.sim.methanol.top file uses function type 1, with coefficients of 0 and 1. This identifies the column in our table file that will be read for the pair interaction. The index file is also necessary for this table file interaction scheme, as it identifies which sites have the MEO type to the edr file. In this case, it is all of them.

```
$ grompp -f cg.sim.methanol.mdp -p cg.sim.methanol.top  -c cg.methanol.gro
  -n cg.sim.methanol.ndx -o cg.sim.methanol.tpr
```

inally, you can run the simulation. On 4 processors, this simulation takes less than an hour.

```
$ mpirun mdrun -s cg.sim.methanol.tpr -o cg.sim.methanol.trr
  -e cg.sim.methanol.edr
```

You can compare the results of this CG simulation to those of the mapped AA simulation. A common property to compare between the two would be the methanol-methanol radial distribution function, which can be accessed with the GROMACS `g_rdf` tool.

7.3 Extended Ensemble and PMatching Tutorial: Heptol Mixtures

This tutorial will take you through the process of force matching an from an extended ensemble of systems, where we will derive a force field that is optimal across a set of related chemical systems (the extended ensemble). For this example, we will perform this calculation for a set of heptane:toluene mixtures of differing compositions, and obtain a single force field that can be used to simulate systems with any ratio of heptane:toluene. The CG representations used here are 3-site models for both molecules.

7.3.1 AA Simulations

Files:

- `X_Y_HEP_TOL.gro`
- `X_Y_HEP_TOL.top`
- `X_Y_HEP_TOL.mdp`
- `heptane.itp`
- `toluene.itp`

In order to perform this extended ensemble force matching calculation, we must first generate the extended ensemble of atomistic simulations for use as input. This tutorial provides input files for atomistic simulations of 5 heptane:toluene systems in the folder: `tutorials/heptol_xn/aa_simulations`

The gro files contain configurations from already equilibrated systems, so you need only grompp and mdrun the files. These systems contain approximately 10,000 atoms, so they will parallelize efficiently onto around 50 cores. These simulations need around 50 ns of simulation time to get enough sampling, and there are 5 such systems, so you will likely want to run them as a job on HPC resources.

The commands you will use are as follows (substituting the composition #'s for X and Y):

```
$ grompp -f X_Y_HEP_TOL.mdp -c X_Y_HEP_TOL.gro -p X_Y_HEP_TOL.top
  -o X_Y_HEP_TOL.tpr

$ mpirun mdrun -s X_Y_HEP_TOL.tpr -o X_Y_HEP_TOL.trr -e X_Y_HEP_TOL.edr
```

7.3.2 Mapping

Files:

- `X_Y_HEP_TOL.trr`
- `X_Y_HEP_TOL.tpr`
- `map.X_Y_HEP_TOL.top`
- `cg.X_Y_HEP_TOL.top`
- `cg.X_Y_HEP_TOL.mdp`
- `cg.X_Y_HEP_TOL.ndx`

As before, you will need to make the trajectories whole with respect to their periodic boundary conditions.

```
$ trjconv -f X_Y_HEP_TOL.trr -force -pbc whole -s X_Y_HEP_TOL.tpr
  -o whole_X_Y_HEP_TOL.tpr
```

You will again use the 'cgmap' tool to generate a CG gro file, and to map each of the 5 atomistic simulations to their CG representations. You can generate this gro file using the cgmap command:

```
$ cgmap -f whole_X_Y_HEP_TOL.trr -p map.X_Y_HEP_TOL.top -c cg.X_Y_HEP_TOL.gro
```

And map the trajectories with these commands:

```
$ grompp -f cg.X_Y_HEP_TOL.mdp -p cg.X_Y_HEP_TOL.top -n cg.X_Y_HEP_TOL.ndx
  -c cg.X_Y_HEP_TOL.gro -o cg.X_Y_HEP_TOL.tpr

$ cgmap -f whole_X_Y_HEP_TOL.trr -p map.X_Y_HEP_TOL.top -s cg.X_Y_HEP_TOL.tpr
  -o cg.X_Y_HEP_TOL.tpr
```

7.3.3 Force Matching

Files:

- inp.txt
- par.txt
- cg.1_4_HEP_TOL.trr
- cg.2_3_HEP_TOL.trr
- cg.1_1_HEP_TOL.trr
- cg.3_2_HEP_TOL.trr
- cg.4_1_HEP_TOL.trr
- corresponding .tpr files

The `inp.txt` and `par.txt` files for the extended ensemble calculation can be found in the folder:
`tutorials/heptol_xn/coarse_graining/extended_ensemble`

Note that both files point to a set of .tpr and .trr files corresponding to those used in the previous mapping step. If you moved these files outside of their respective folders or named them differently, you will need to update the paths in inp.txt and par.txt to reflect this.

The `par.txt` file specifies the interactions that will be force matched in this calculation. In this case, all of these interactions are present in each of our source systems, as they all contain mixtures of heptane and toluene. It is also possible to include interactions here that are present in only some of the extended ensemble systems.

You can run this calculation as before, in the same folder as `inp.txt` and `par.txt`, with:

```
$ mpirun cgff
```

Since these are larger systems and there are 5 of them that the calculation needs to loop through, this calculation is best submitted as a job on an HPC system. It parallelizes very well up to a very large number of processors, so the more resources you can throw at it the better. In our tests, it takes around 48 hours on 64 processors.

This will generate a large set of output files, which are analogous to those generated in the methanol tutorial. Each of the per-interaction files will be labeled with the interaction name specified in par.txt. We will work directly with the `f_force.XXX.total.YYY.dat` force field files in the coming steps.

We have also included `inp.txt` and `par.txt` files for generating independent force fields for each of the individual systems, if you would like to do so for comparison purposes. These files are found in the corresponding folders:
`tutorials/heptol_xn/coarse_graining/x_y_heptol`

7.3.4 Make Table Files

Files:

- `f_forces.Pair_Interaction.total.CMCB.dat`
- `f_forces.Pair_Interaction.total.CMCF.dat`
- `f_forces.Pair_Interaction.total.CMCM.dat`
- `f_forces.Pair_Interaction.total.CTCB.dat`
- `f_forces.Pair_Interaction.total.CTCF.dat`
- `f_forces.Pair_Interaction.total.CTCM.dat`
- `f_forces.Pair_Interaction.total.CTCT.dat`
- `f_forces.Pair_Interaction.total.CBCB.dat`
- `f_forces.Pair_Interaction.total.CFCB.dat`
- `f_forces.Pair_Interaction.total.CFCF.dat`
- `f_forces.Angle.total.Angle_CT-CM-CT.dat`

- f_forces.BondStretch.total.Bond_CB-CB.dat
- f_forces.BondStretch.total.Bond_CF-CB.dat
- f_forces.BondStretch.total.Bond_CT-CM.dat
- make_gmx_table.sh

As is sometimes the case for force matching calculations, there are several forces that need to be processed and converted into table files. Since we are going to continue on to pressure-match these systems as well, we will need to generate table files for both LAMMPS and GROMACS.

We will start with processing the files for use with GROMACS. The script make_gmx_table.sh contains the list of commands needed to perform this processing for the full set of table files. You can run it via:

```
$ sh make_gmx_table.sh
```

This should generate the full set of tables in the folder:
/tutorials/heptol_xn/coarse_graining/table/

Now, we can convert these table files for use in LAMMPS. This conversion makes use of a script in the YYY_toolkit/scripts directory called translate_table.py. It is a simple python script that converts table files from GROMACS format and units to LAMMPS ’real’ units and format. You should have this script on your path. If so, then you can move into the ’tables’ folder and run the command:

```
$ sh ../lammps_table.sh
```

This is a helper script to run the full set of table conversion commands. Inspect this script to see the form of the translate_table.py commands; it is also documented within the python script itself.

7.3.5 Pressure Matching: Rerun

Files:

- cg.1_1_HEP_TOL.trr
- cg.rerun.1_1_HEP_TOL.gro
- cg.rerun.1_1_HEP_TOL.mdp
- cg.rerun.1_1_HEP_TOL.top
- cg.rerun.1_1_HEP_TOL.ndx

At this point, we could go ahead and run a CG simulation of any of our heptane:toluene mixtures. However, if we do so under constant pressure (NPT) conditions, the simulations will blow up. This is because the pressure of the CG simulation is substantially higher than that of the underlying AA system - a common problem with bottom-up coarse-graining methods.

Therefore, we will instead proceed by determining a volume-dependent pressure correction for use with the CG simulation. Here, we will generate a pressure correction for the 1:1 heptane:toluene system. You can follow an analogous procedure for any of the other compositions, as the pressure matching calculations for extended ensemble calculations are performed independently of the other systems in the ensemble.

The first step in this process is to determine an approximate correction based on the pressure of the mapped trajectory with the CG force field. We will do so using the 'mdrun -rerun' command in GROMACS, which allows us to evaluate the forces (and therefore the pressures) over an existing trajectory with a new force field.

Note that the input files (mdp and top in particular) are of the type used for CG simulation, not for mapping. This means that the .mdp file has a fully filled out 'energygrps' entry for all pair interactions in the system, and that the .top file has all intramolecular interactions set to function type 8.

Also note that as of GROMACS 4.5.3, there is a bug with rerunning trajectories when you compile with openMPI that causes nonsense energy outputs. Therefore, we recommend that you perform the rerun with an mdrun executable compiled without openMPI.

Rerun the trajectory from the `tutorials/heptol_xn/pressure_matching/rerun` folder using the following commands:

```
$ cp ../../coarse_graining/extended_ensemble/table/*xvg ./

$ grompp -c cg.rerun.1_1_HEP_TOL.gro -p cg.rerun.1_1_HEP_TOL.top
  -f cg.rerun.1_1_HEP_TOL.mdp -n cg.rerun.1_1_HEP_TOL.ndx
  -o cg.rerun.1_1_HEP_TOL.tpr

$ mdrun-noMPI -rerun ../../coarse_graining/1_1_heptol/cg.1_1_HEP_TOL.trr
  -s cg.rerun.1_1_HEP_TOL.tpr -e rerun.edr -nt 8
```

This rerun should proceed relatively quickly, as there are 1000x fewer frames for which to compute forces compared to the original atomistic simulation, and we are calculating the forces in the CG system, which has fewer particles and therefore fewer pair interactions. Once the rerun is complete, extract the CG pressures and volumes using the GROMACS g_energy utility:

```
$ g_energy -f rerun.edr -o cg_volumes.xvg

$ g_energy -f rerun.edt -o cg_pressures.xvg
```

While you're at it, extract the pressures and volumes from the corresponding atomistic simulation as well:

```
$ g_energy -f ../../aa_simulations/1_1_heptol/1_1_HEP_TOL.edr -o aa_volumes.xvg
```

```
$ g_energy -f ../../aa_simulations/1_1_heptol/1_1_HEP_TOL.edr -o aa_pressures.xvg
```

7.3.6 Pressure Matching: Zeroth Iteration

Files:

- `aa_volumes.xvg`
- `aa_pressures.xvg`
- `cg_volumes.xvg`
- `cg_pressures.xvg`
- `config.ini`
- `cgheptol.lmp`
- `data.cgheptol`

Copy the xvg files you just generated into
`tutorials/heptol_xn/pressure_matching/iteration_zero`,
and use the p_match tool to generate the approximate pressure correction with the command:
`p_match config.ini`

The `config.ini` file contains all the settings for the pressure matching calculation. Here, we are using an analytic basis of the form recommended by Das and Andersen with two terms. This is equivalent to directly correcting the pressure and the compressibility of the CG model. A number of diagnostic files will be generated when running this command, but we are most interested in `psi.dat`, which contains the two coefficients for our pressure correction. Our `psi.dat` file contains the following:

```
[AVG_AA_VOLUME]
113.610979
[PSI_COEFF] 2
101.348739
14.024344
[N_SITES]
1602
```

Your average volume and coefficients should be similar but not identical, as our calculations will differ by the particular AA simulations used as input.

The psi coefficients are given in units of bar*nm^3, and the average volume is given in nm^3. In order to use these with LAMMPS, we must convert them to the corresponding LAMMPS 'real' units, which use atm for pressures and Å^3 for volumes. These conversions are:

$$\text{X nm}^3 = \frac{1000\ \text{Å}^3}{1\ \text{nm}^3} = 1000\ \text{Å}^3 \tag{1}$$

$$\text{X bar nm}^3 * \frac{1\ \text{atm}}{1.01325\ \text{bar}} * \frac{1000\ \text{Å}}{1\ \text{nm}^3} = 986.923\ \text{atm Å}^3 \tag{2}$$

Edit these converted coefficients and the average volume into the `cgheptol.lmp` file in the `test_sim` folder, replacing the angular brace placeholders on line 70 of this file, making sure that there are not remaining braces in the final line.

Note that the `data.cgheptol` file contains the coordinates and specifies the connectivity of the molecules in this system. It was generated using the (unsupported) script `YYY_toolkit/scripts/translate_top.py` to pull information from .gro, .top, and .ndx files.

Assuming you have built LAMMPS for use with openMPI, you can run the simulation via:

```
$ mpirun lmp_linux < cgheptol.lmp &> out.dat
```

This will generate a trajectory file, `outfile.xtc`, and a set of .dat files that print various thermodynamic quantities. We are most interested in `vol.dat` and `press.dat`, which contain the volume and pressure (converted to GROMACS units). We print in GROMACS units so that we can directly use the output of this simulation in the next step of the pressure matching process, which will compare the PV curve estimated by rerunning over the mapped ensemble with this CG simulation.

Once this simulation is done, plot the distribution of volumes in the AA simulation against the distribution of volumes in this CG simulation using your favorite plotting tool. You will see that the pressure correction used here is not quite right, and the average density and compressibility of the CG simulation are therefore different than those of the AA model. This is because of subtle structural differences in the simulated CG ensemble compared to the mapped ensemble.

7.3.7 Pressure Matching: First Iteration

Files:

- `aa_volumes.xvg`
- `aa_pressures.xvg`

- vol.dat
- press.dat
- config.ini
- cgheptol.lmp
- data.cgheptol

To correct for these structural differences, we can apply an iterative method to update the pressure correction for this CG model. To do so, we will compare the pressure-volume (PV) curve of the CG simulation in the last step to the PV curve of the AA simulation, and infer an update to the pressure correction based on this difference.

To perform this calculation, copy the AA pressure and volume files into the folder:
tutorials/heptol_xn/pressure_matching/iteration_one
as well as the press.dat and vol.dat files generated during the previous iteration. You can then run the command:

```
$ p_match config.ini
```

To generate an updated set of psi coefficients based on the difference between the simulated AA and CG PV curves. This calculation will read in the previous psi coefficients from the psi.dat file in the iteration_zero' folder, and generate a new psi.dat file here that incorporates the updates.

You can then proceed to simulate with these coefficients in the same way as before - edit in the average volume and converted coefficients into cgheptol.lmp, then simulate with

```
$ mpirun lmp_linux < cgheptol.lmp &> out.dat
```

When you plot the distribution of volumes this time, it should be closer to reference AA distribution. You can continue iterating over this process by generating new sets of coefficients and simulating with them until the volume distribution is satisfactorily converged. For most systems, this takes fewer than 10 iterations.