

# XMULTI Manual.

**Christian J. Burnham, University College Dublin, 2019**

christianjburnham@gmail.com

Introduction	2
Compiling and running XMULTI	2
Support	2
Note on conventions used in this document.	3
src directory	3
input_files/mannitol directory	4
Geometry file format.	5
Structure input file format for rigid body coordinates.	5
Structure input file format for XYZ input coordinates.	7
Structure input files: cell vector format.	8
Multipole file format	9
Model file format	10
Control file format	12
XMULTI output files	15
A simple test of XMULTI'S multipole interactions.	17
Appendix I. Euler angles in XMULTI	18

## Introduction

XMULTI is primarily a code for calculating the energies and forces of a molecular crystal given a structure and a defined potential energy surface.

It uses triclinic periodic boundary conditions, with an Ewald sum for long-range electrostatic interactions.

The electrostatics are calculated using multipole-multipole interactions, going up to octopole-octopole interactions.

The multipole interactions are calculated in spherical harmonics, which is faster than the standard Cartesian approach.

The real space sum uses a supercell approach in which an arbitrarily long cut-off can be used, i.e. the cutoff sphere is not confined to fit inside one unit cell.

XMULTI calculates energies, forces, COM forces in XYZ or fractional coordinates, torques, angular derivatives and derivatives with respect to the cell vectors.

XMULTI can perform gradient optimizations in the full set of coordinates, i.e. including relaxation of the cell-vectors.

XMULTI can be run under an external pressure.

XMULTI can also be run with periodic boundary conditions turned off.

## Compiling and running XMULTI

- 1) Navigate to the xmulti/src directory
- 2) Type: make. This should produce an executable, xmulti.f in xmulti/bin
- 3) Make a new directory somewhere where you want to run the code and then navigate to this directory, e.g.: cd; mkdir test; cd test
- 4) If this is your first time running XMULTI, try copying over the example input files from the input\_files directory: cp ../xmulti/input\_files/mannitol/\* ./
- 5) Run the executable with the command: ../xmulti/bin/xmulti

Running ./xmulti -v will return the current version number.

## Support

If you have any questions on the running of XMULTI, or suggestions for improvements, then you can contact the author at christianjburnham@gmail.com.

I have very limited resources, and it is not really part of my current job to provide support for this code, so please be reasonable in your requests! And users who are nice to me are more likely to have their emails answered in a timely manner.

Writing a user manual is hard work, and the author, alas, did not have infinite time in which to write it. Undoubtedly there will be parts readers wish could be explained better, or in need of further explanation. Please contact the author with suggestions. (Remembering the 'be nice' rule.)

## Note on conventions used in this document.

This document specifies the format of input files, and contains lines like:

```
[a (real), b (string), c (real)]
```

which should be read as: three values need to be entered. The first and third values are real numbers, and the second value is a string, e.g.:

```
2.3d0 hello 2.71828d0
```

would be a valid input.

The [ ] brackets denote user-defined input.

The notation:

```
<BUY>, [n (integer)], <APPLES> | <BANANAS>, {<TODAY> | <TOMORROW>}
```

means this line requires the BUY keyword, followed by a user-defined integer, followed by the string-literal APPLES *or* BANANAS.

The { } braces denote that the {<TODAY>|<TOMORROW>} part is optional, so the following would be valid inputs.

```
BUY 3 APPLES TODAY
BUY -4 BANANAS
BUY 2 APPLES TOMORROW
BUY 6 APPLES
```

Note, the commas in the input format descriptions are not to be entered in the actual input.

XMULTI uses subroutines to split text on white spaces, and so the input files don't have to follow a very rigid FORTRAN style format. However, it still has some problems with parsing tabs, so these should be avoided.

## src directory

**XMULTI.F:** The main code, which performs crystal structure prediction for empirical models.

**XMULTI\_COMMON:** Array declarations for arrays used by XMULTI.F and other codes.

**ALLOCATE.F:** Allocates the arrays defined in XMULTI\_COMMON, using input file SIZE.DAT

**NR\_MOD.F:** Various numerical subroutines.

**PARSE\_TEXT.F:** Various routines for text parsing

**MAKEFILE:** The XMULTI makefile.

## input\_files/mannitol directory

**SIZE.DAT:** Array size dimensions.

**CONTROL.DAT:** XMULTI main control file.

**INPUT.MANNITOL:** Input struture file. Specifies the starting molecular arrangement for this run.

**MOLECULAR\_GEOMETRY.MANNITOL:** Specifies the geometry of each molecule/conformer.

**MODEL.MANNITOL:** Mostly for specifying the non-electrostatic part of the potential energy surface. Also used to input the atomic masses, and the short-range damping coefficient.

**MULTIPOLES.MANNITOL:** Specifies the atomic multipoles for every molecule/conformer.

**PROB.MANNITOL** (optional): Used in the basin hopping monte-carlo algorithm to determine the probability of flipping from one conformer to another conformer.

To illustrate the different input structure formats available this directory also contains the following files.

**INPUT\_RIGID\_XYZ.MANNITOL:** XMULTI structure input file. Specifies the starting molecular arrangement for this run in RIGID XYZ format (COM X,Y,Z, Euler angles for each molecule).

**INPUT\_XYZ.MANNITOL:** Specifies the same coordinates as the above, but in XYZ format. (X,Y,Z for each atomic site.)

**INPUT\_RIGID\_FRAC.MANNITOL:** Specifies the same coordinates as the above, but in RIGID FRAC format. (COM: F1,F2,F3, Euler angles for each molecule.)

**INPUT\_XYZ\_PBC\_OFF.MANNITOL:** Specifies the same coordinates as above, but in XYZ format with periodic boundary conditions off.

**INPUT\_RIGID\_XYZ\_PBC\_OFF.MANNITOL:** Specifies the same coordinates as above, but in RIGID XYZ format with periodic boundary conditions off.

To use these alternative input files, either rename them to 'input.mannitol' the name specified in the CONTROL file,  
or rename the input file name in the CONTROL file.

## Geometry file format.

Contains the molecular geometry defining each rigid molecule in xyz coordinates.

Format, for each separate molecule/conformer

### First line:

```
<molecule> [MOL_NAME (string)], {<conformer> [CONFORMER_ID (integer)]},  
<NATOMS> (integer), {<energy> [ENERGY (real)]}
```

where MOL\_NAME is the (unique) name of the molecule, NATOMS is an integer giving the number of atoms for this molecule.

<conformer> is an option required for cases where we want to specify multiple conformers of one molecule, where CONFORMER\_ID is the index of the conformer. Conformer indices must be numbered such that they fill 1..N\_CONFORMERS without gaps, where N\_CONFORMERS is the total number of conformers for that molecule.

<energy> is an option for specifying the gas-phase energy of that conformer/molecule (default ENERGY = 0.0d0), and is specified in whatever are the internal energy units (either kJ/mol or kcal/mol) as set in the CONTROL file.

### Next NSITES lines:

```
[SITE_NAME (string), X,Y,Z (real)]
```

where SITE\_NAME is the name of the site, e.g. 'O' or 'H', and X,Y,Z are real numbers giving the XYZ coordinates of that site in Angstroms. The SITE\_NAME values do not have to be distinct, and one molecule can have multiple sites with the same SITE\_NAME. The identification of the site is determined by its order in the list.

## Structure input file format for rigid body coordinates.

Specifies the initial coordinates.

### First line:

```
[NMOL] (integer)
```

where NMOL is the number of molecules in the structure.

**Second line** (for periodic calculations only, otherwise left blank):

<XYZ> | <ANGLE>, [C1, C2, C3, C4, C5, C6 (real)]

where [<XYZ>|<ANGLE>] determines whether the cell-vectors are in Cartesian XYZ format or in standard crystallographic [ANGLE] format. See section **cell vector format** for details.

**Third line:**

<RIGID FRAC> | <RIGID XYZ>

which determines the format of the input coordinates. Then, depending on the format:

**If RIGID FRAC:**

Coordinate file consists of NMOL lines, where each line is of the format:

<MOLECULE> [MOL\_NAME] {<CONFORMER> [conformer\_id]} <COORD>  
[F1,F2,F3,PHI,THETA,PSI (real)],

where MOL\_NAME is the name of the molecule (which must correspond to a molecule name in MODEL\_FILE and GEOMETRY\_FILE).

F1,F2,F3 (real) are the fractional coordinates of the center of mass of that molecule.

PHI,THETA,PSI are the Euler angles (in radians) of that molecule.

The conformer part is optional, but if you have a geometry file specifying different molecular conformers, then you will want to also specify the conformer id of that molecule. (It defaults to conformer\_id = 1)

**If RIGID XYZ:** Coordinate file consists of NMOL lines, where each line is of the format:

<MOLECULE> [MOL\_NAME] {<CONFORMER> [conformer\_id]} <COORD>  
[X1,X2,X3,PHI,THETA,PSI (real)],

where MOL\_NAME is the name of the molecule (which must correspond to a molecule name in MODEL\_FILE and GEOMETRY\_FILE)

X1,X2,X3 (real) are the Cartesian coordinates of the center of mass of that molecule.

PHI,THETA,PSI are the Euler angles (in radians) of that molecule.

The conformer part is optional, but if you have a geometry file specifying different molecular conformers, then you will want to also specify the conformer id of that molecule. (It defaults to conformer\_id = 1)

## Structure input file format for XYZ input coordinates.

Xmulti is written for rigid molecules only, but it is often the case that the coordinates to hand are in XYZ format, in which xyz coordinates of every atomic site are specified, in which case XMULTI provides a convenient XYZ input method whereby it attempts to convert input XYZ coordinates into internal rigid body equivalents.

Note, the following method works best when the xyz atomic coordinates are very close to the rigid body geometries defined in the GEOMETRY\_FILE (related by a rotation and a translation of the COM).

### First line:

[NMOL] (integer)

where NMOL is the number of molecules in the structure.

### Second line (for periodic calculations only, otherwise left blank):

<XYZ> | <ANGLE>, [C1, C2, C3, C4, C5, C6 (real)]

where [<XYZ>|<ANGLE>] determines whether the cell-vectors are in Cartesian XYZ format or in standard crystallographic [ANGLE] format. See section **cell vector format** for details.

### Third line:

<XYZ>

Specifies XYZ format.

### Then, NMOL blocks, where each block has the format:

<molecule> [MOL\_NAME (string)], {<CONFORMER> [conformer\_id]},  
{<ANCHORS> anchor1, anchor2, anchor3, (integer)}

where MOL\_NAME must correspond to a molecule name in MODEL\_FILE and GEOMETRY\_FILE.

ANCHORS specifies a list of three integers (default values 1 2 3) denoting the sites which are used to determine the space-fixed axis of the input structure, so that XMULTI can find the rotation angles from the body-fixed axes.

*Background:* The system used is:

$$\hat{\mathbf{x}} = \frac{\mathbf{r}_{21}}{r_{21}},$$

$$\hat{\mathbf{y}} = \frac{\mathbf{r}_{21} \times \mathbf{r}_{31}}{|\mathbf{r}_{21} \times \mathbf{r}_{31}|},$$

$$\hat{\mathbf{z}} = \hat{\mathbf{x}} \times \hat{\mathbf{y}},$$

where  $\mathbf{r}_{ji} = \mathbf{r}_j - \mathbf{r}_i$ ,  $r_{ji} = \sqrt{\mathbf{r}_{ji} \cdot \mathbf{r}_{ji}}$ , and  $\hat{\mathbf{x}}$ ,  $\hat{\mathbf{y}}$  and  $\hat{\mathbf{z}}$  are the orthonormal space fixed axes. (The same process is used on the body-fixed coordinates to determine the body fixed axes, from which XMULTI determines the rotation matrix.)

The conformer part is optional, but if you have a geometry file specifying different molecular conformers, then you will want to also specify the conformer id of that molecule. (It defaults to conformer\_id = 1)

**Next NATOMS\_IN\_MOLECULE lines in block:**

[SITE\_NAME (string), X,Y,Z (real)]

where NATOMS\_IN\_MOLECULE are the number of atoms in that molecule, SITE\_NAME is the name of the atomic site, which must correspond to, and be in the same order, as the sites in MODEL\_FILE and GEOMETRY\_FILE and X,Y,Z are the XYZ coordinate of the site in Angstroms.

## Structure input files: cell vector format.

The cell vector format line is in the form

<XYZ> | <ANGLE>, [C1, C2, C3, C4, C5, C6 (real)],

where <XYZ>|<ANGLE> determines whether the cell-vectors are in Cartesian XYZ format or in standard crystallographic [ANGLE] format.

Input files are often created from outputs from previous runs, in which case the cell vector format line may contain additional values, or strings recording the energy, pressure or other information of that structure. But, this information is ignored when read in.

For calculations without periodic boundary conditions, the entire second line is omitted and to be replaced by a blank line.

**<XYZ> format**

C1,C2,C3,C4,C5,C6 specify  $a_x, b_x, b_y, c_x, c_y, c_z$ , where  $a_x, b_x, b_y, c_x, c_y, c_z$  are Cartesian cell vector components in Angstroms.

*Background:* The three cell vectors of a triclinic cell can be stored in a 3x3 matrix as



$$\mathbf{C} = \begin{bmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \\ a_z & b_z & c_z \end{bmatrix}$$

However, there are only six independent components, as the following argument shows. The x axis can always be chosen to be parallel to the a vector. The y axis can always be chosen such the b vector lies in the x-y plane, which means that, with this choice of axis, the cell vectors can be written in upper triangular form as

$$\mathbf{C}' = \begin{bmatrix} a_x & b_x & c_x \\ 0 & b_y & c_y \\ 0 & 0 & c_z \end{bmatrix}$$

which has just six components.

XMULTI has subroutines allowing it to convert between the <XYZ> and <ANGLE> coordinate systems as convenient.

#### <ANGLE> format

C1,C2,C3,C4,C5,C6 specify  $|a|$ ,  $|b|$ ,  $|c|$ ,  $\alpha$ ,  $\beta$ ,  $\gamma$ , where  $|a|,|b|,|c|$  in Angstroms, and  $\alpha, \beta, \gamma$  are in *degrees*.

## Multipole file format

Contains the multipoles for each model

Format for each separate molecule/conformer type:

#### First line:

<molecule> [MOL\_NAME (string)], <conformer> [CONFORMER\_ID]

where MOL\_NAME is the name of the molecule, CONFORMER is the conformer ID, which corresponds to the conformers given in molecular\_geometry file. If no conformer ID label is supplied, the code will assume that the multipoles belong to the first conformer for that molecule.

#### Next NATOMS\_IN\_MOL lines:

[SITE\_NAME (string), q00 (=CHARGE), q11c (=Dx), q11s (=Dy), q10 (=Dz), q20, q21c, q21s, q22c, q22s, q30, q31c, q31s, q32c, q32s, q33c, q33s (real)]

where SITE\_NAME is the name of the site, which must agree with the site name in molecular\_geometry.dat, and be in the same order.

CHARGE is the charge on that site, in units of  $|e|$ .

$D_x$ ,  $D_y$ ,  $D_z$  are the dipole moments on that site, in units of  $|e| \text{Bohr}$ . Nb. the GDMA code can output dipole moments in order  $D_z, D_y, D_x$ , in which case this order needs to be changed to be consistent with XMULTI.  $q_{20}$ ,  $q_{21c}$ ,  $q_{21s}$ , etc. are the higher order multipoles on that site, in atomic units:  $|e| \text{Bohr}^n$ , where  $n$  is the rank, and following the conventions from the output of Anthony Stone's GDMA code (available at <http://www-stone.ch.cam.ac.uk/pub/gdma/>).

XMULTI can also handle truncated multipole expansions. For instance, if you only want to calculate multipoles up to rank 2, then just enter in the first 9 components  $q_{00}$ ,  $q_{11c}$ ,  $q_{11s}$ ,  $q_{10}$ ,  $q_{20}$ ,  $q_{21c}$ ,  $q_{21s}$ ,  $q_{22c}$ ,  $q_{22s}$ , leaving the rest of the line blank. XMULTI will then calculate interactions for that site only up to rank 2.

And, of course, if you only want to use charges, then just enter the first component,  $q_{00}$ , leaving the rest of the line blank.

*Background:* There are  $2n+1$  multipole components of rank  $n$ . Thus, the rank 0 multipoles (charges) have one component, the rank 1 multipoles (the dipoles) have three components, the rank 2 multipoles (the quadrupoles) have five components, and the rank 3 multipoles (the octopoles) have seven components. Currently, XMULTI only goes up to rank 3 (octopole) moments, but this is usually good enough for most applications given the multipole expansion strongly converges with increasing rank.

It is also worth mentioning that the time of an energy call scales roughly as  $n^3$ , where  $n$  is the maximum multipole rank.

## Model file format

Contains the Lennard-Jones parameters and particle masses.

### First line:

```
<molecule> [MOLECULE_NAME (string)]
```

which specifies the molecule name. Note that no conformer is specified, as the data in this file will be applied to every conformer.

### Next NATOMS lines:

```
[SITE_NAME (string)], <type> [TYPE (integer)], <mass> MASS (float)
```

where NATOMS is the number of atoms for this molecule.

SITE\_NAME is the site name of this atom, e.g. 'O', 'H', etc.

TYPE is an index specifying the interaction type of this site. Multiple sites can share the same interaction types. Even sites across different molecules can share the same type. When it comes to specifying the non-electrostatic pair interactions below, these can be

done for each separate pair of interaction types, e.g. for all particle pairs involving type 3 and type 7.

MASS is the particle mass, in atomic units.

### Following the specification of all the molecules:

{<damp> [DAMP]}

where DAMP is an optional short-range electrostatic damping coefficient in Angstroms. Default: Damp = 0.

*Background:* The damped electrostatic energy between two unit charges is calculated according to

$$U^{cc} = \frac{\text{erf}\left(\frac{1}{\sqrt{2}} \frac{r}{r_d}\right)}{r}$$

where  $r_d$  is the damping width. The above can be compared to  $U^{cc} = 1/r$  without damping, and, in fact,  $U^{cc}$  reduces to  $1/r$  when  $r/r_d \rightarrow \infty$ .

The same damping kernel is used for multipoles of all ranks.

{<nspline> [NSPLINE]}

Sets the number of spline points used in the pairwise splines for the non-electrostatic part of the model. (Default: NSPLINE = 100).

<inter> {P1, P2, P3,... (integer)}

where power1, power2, power 3 specify the default inverse powers for the non-electrostatic pair interactions, e.g. use P1 = 6, P2 = 12 for a standard 6-12 potential.

**Then, each bond is specified by a block of three lines. (Four, if spline-data is included.)**

#### PAIR line:

<PAIR>, [TYPE1 (integer), TYPE2 (integer)]

which specifies that this intermolecular interaction is to take place between all sites with type TYPE1 and all sites with types TYPE2.

#### POWER line:

<POWER>, [P1, P2, P3... (real)]

where  $P_i$  is the power in inverse  $r$  of the  $i$ th term in the interaction

**COEFF line:**

<COEFF>, [C1, C2, C3... (real)]

where  $C_i$  is the coefficient for the  $i$ th term, such that the pair energy between the sites is given by:

$$U^{pair} = \frac{C_1}{r^{P_1}} + \frac{C_2}{r^{P_2}} + \frac{C_3}{r^{P_3}} + \dots$$

where  $r$  is the inter-site distance.

**SPLINE line (optional):**

{<SPLINE>, <rmin> [RMIN (real)], <rmax> [RMAX (real)], <r0> [R0 (real)], <u0> [U0 (real)], <dudr0> [DUDR0 (real)]}

If the spline line is used, then a spline fit of  $UPAIR_{ij}$  is attempted between distances RMIN and RMAX. The short-range part of the potential ( $r < RMIN$ ) is also covered by a spline, which goes through the point R0, with energy U0, and gradient DUDR0. These values can be 'guesstimated' to produce a curve that behaves well as  $r \rightarrow 0$  (i.e. doesn't become very negative). The long-range ( $r > RMAX$ ) part of the polynomial is handled by the  $UPAIR_{ij}$  polynomial.

**Control file format**

The main control file.

**KEYWORDS**

**To be specified at the beginning of the file:**

<input\_file>, [INPUT\_FILE (string)]

sets INPUT\_FILE name.

<model\_file>, [MODEL\_FILE (string)]

sets MODEL\_FILE name.

<geometry\_file>, [GEOMETRY\_FILE (string)]

sets GEOMETRY\_FILE name.

{<batch\_file>, [BATCH\_FILE (string)]}

sets BATCH\_FILE name.

{<prob\_file>, [PROB\_FILE (string)]}

sets PROB\_FILE name.

## Then

<cvec output>, <XYZ>|<ANGLE>

sets the output cell vectors to either XYZ or ANGLE format.

<rigid output>, <XYZ>|<FRAC>

sets the rigid-body output to either XYZ or FRAC (fractional coordinates) format.

<input units>, <KJ>|<KCAL>

sets the input energy units to either kJ/mol or kcal/mol

<output units>, <KJ>|<KCAL>

sets the output energy units to either kJ/mol or kcal/mol

<pressure>, [PRESSURE (real)]

sets the external pressure to pressure, in bar.

<mintol>, [MINTOL (real)]

sets the tolerance of the local optimizations to mintol in the input energy-units, where the default value is 0.0d0, and suggested value of 0.1 kJ/mol. Note that a value of 0.1 kJ/mol has been found to substantially speed up the basin-hopping.

{<rcut>, [RCUT (real)]}

sets the cut-off radius to RCUT, in Angstroms. (default, rcut = 8.0d0)

<ewald\_error>, [ERROR (real)]

sets the Ewald error term to ERROR. (recommended, error = 1.d-7)

<findmin>

performs a conjugate gradient energy minimization to find the local minimum.

<stop>

halts the code.

## Directives to print out results

<energy>

calculates and prints the energy of the simulation cell, and decomposes the total energy into its constituent terms.

The electrostatic components are decomposed into contributions from each multipole rank.

*Background:* The total electrostatic energy can be written as

$$U^{elec} = \frac{1}{2} \sum_{n=0} U_n,$$

where

$$U_n = \sum_i^N \langle \mathbf{M}^{i(n)}, \phi^{i(n)} \rangle$$

where  $\mathbf{M}^{i(n)}$  is the rank  $n$  multipole tensor on the  $i$ th particle,  $\phi^{i(n)}$  is the rank  $n$  field tensor on the  $i$ th particle, and  $\langle \mathbf{M}^{i(n)}, \phi^{i(n)} \rangle$  is a tensor inner product between  $\mathbf{M}^{i(n)}$  and  $\phi^{i(n)}$ . The  $U_n$  then are the rank  $n$  contributions to the electrostatic energy, with  $U_0$  being the charge energy,  $U_1$  being the dipole energy, and so on.

<print rigid>

prints the current rigid-body coordinates to output.rigid

<print xyz>

prints the current cartesian coordinates to output.xyz

<print multipoles>

Prints out the molecular multipoles in Gaussian and GDMA formats.

### Testing routines for comparing numerical and analytical derivatives.

{[DELTA]} is optional in each case, and gives the displacement used to calculate the numerical derivatives. The default value is 1.d-6. Derivatives are calculated in the most naive way:  $(f(x+dx) - f(x))/dx$ , and so will become numerically unstable close to the minimum.

<test forces>, {[DELTA]}

tests the forces on each atom by comparing analytical and numerical derivatives.

<test rforces>, {[DELTA]}

tests the rigid-body forces

<test fforces>, {[DELTA]}

tests the fractional forces, i.e.  $dU/df_1$ ,  $dU/df_2$ ,  $dU/df_3$ , where  $f_1, f_2, f_3$  are the fractional COM coordinates for that molecule.

<test torques>, {[DELTA]}

tests the torques for each molecule

<test euler>, {[DELTA]}

tests the euler angle derivatives for each molecule

<test cvec>, {[DELTA]}

tests the cell-vector derivatives.

### **Basin hopping directives**

<basin temp>, [TEMP (real)]

sets the temperature of the basin-hopping simulation to temp.

<basin rmin>, [RMIN (real)]

sets the minimum approach distance in the basin-hopping simulation to <rmin>, in Angstroms. (recommend, rmin = 0.6)

<basin delta\_frac>, [DELTA\_FRAC (real)]

sets the basin\_hopping fractional step to delta\_frac (dimensionless).

<basin delta\_cvec>, [DELTA\_CVEC (real)]

sets the basin\_hopping fractional step to delta\_cvec, in Angstroms.

<basin delta\_angle> [DELTA\_ANGLE (real)]

sets the basin\_hopping fractional step to delta\_angle, in radians.

<basin run>

starts a basin-hopping simulation

<basin restart>

restarts a basin-hopping simulation

## **XMULTI output files**

OUTPUT.XYZ: produced by PRINT XYZ command in input file. XYZ format output of current coordinates.

OUTPUT.RIGID: produced by PRINT RIGID command in input file. COM and Euler angles of each molecule. The COM coordinates

can be in either Cartesians, or fractional coordinates, according to directives <rigid output>, <XYZ>|<FRAC> in control file.

### **Basin Hopping output files:**

MONTE\_MIN.XYZ: Contains VMD suitable xyz coordinates of the lowest minima.

MONTE\_MIN.RIGID: Contains rigid-body coordinates of the lowest minima. (Same as monte\_min.xyz, with a different coordinate system.)

MINIMUM.DAT: contains list of the energies of the lowest minima. First column: Energy. Second column: internal pressure (bar). Third column: density (g/cm<sup>3</sup>)

MONTE\_LOG.DAT: Log file of the basin-hopping run.

MONTE\_ENERGY.DAT: Prints out the energy, internal pressure and density on each step of the Monte-Carlo run.

RELAXED.XYZ: Outputs the XYZ coordinates of the last accepted minimum of the Monte-Carlo run. To save space, X-MULTI will only print out an accepted minimum structure if it is different from the last accepted minimum.

RELAXED\_ENERGY.DAT: Outputs the energy of the last accepted minimum of the Monte-Carlo run. There should be a 1:1 correspondence of energies in RELAXED\_ENERGY.DAT to structures in RELAXED.XYZ.

OUTPUT0.RIGID: Outputs the rigid-body coordinates on each step of the Monte-Carlo run.

For periodic systems, MONTE\_LOG.DAT takes the form:

1st column: step number

2nd column: energy of current step

3rd column: energy of the trial step

4th column: lowest energy structure found so far.

5th column: [accept,reject] indicates whether the last monte-carlo step was accepted or rejected.

6th column: internal pressure, in bar.

7th column: density, in g/cm<sup>3</sup>

8th column: acceptance probability of the last Monte Carlo move.

9th column: percentage of Monte Carlo steps accepted

For non-periodic systems, MONTE\_LOG.DAT is the same as the above, but without columns 6 and 7.

Each 'frame' of output.xyz, output.rigid and monte\_min.rigid can also be used as input files.



## A simple test of XMULTI'S multipole interactions.

This section describes a good test of the multipole interactions as implemented by XMULTI. It compares numerical multipoles created by infinitesimally displacing multipoles of a lower rank with the equivalent analytic multipole interactions as hard-coded by XMULTI. Performing these tests allow us to be more or less certain that the code is calculating the higher order multipoles correctly.

i) Make a local directory `./mtest` and two sub directories `./mtest/test1` and `./mtest/test2`

ii) Copy over the XMULTI input files from `../xmulti/input_files/mtest/` into local directory `./mtest`,

Copy over input files from `../xmulti/input_files/mtest/test1` into local directory `./mtest/test1`

Copy over input files `../xmulti/input_files/mtest/test2` into local directory `./mtest/test2`

iii) Inspect the mtest input files. They define 4 molecules, called 'charge\_mol', 'dipole\_mol', 'quad\_mol' and 'oct\_mol'.

**charge\_mol** is just a point charge.

**dipole\_mol** is two charges of opposite sign, displaced by a small amount, which approximates a pure dipole.

**quad\_mol** is two dipoles of opposite sign, displaced by a small amount, which approximates a pure quadrupole.

**oct\_mol** is two quadrupoles of opposite sign displaced by a small amount, which approximates a pure octopole.

Running XMULTI in this directory, should output the multipole moments of each molecule.

iv) Now inspect the `mtest/test1` input files. They are the same as in `mtest`, except for the fact that the structure input file defines a `quad_mol` molecule interacting with a `dipole_mol` molecule, separated by a few Angstroms, with the molecules also rotated. Running XMULTI in this directory should produce an output giving the energy of this two-molecule system.

v) Now inspect the `mtest/test2` input files. In this directory, all the molecules contain only 1 atomic site apiece, with input atomic multipoles given by the molecular multipole expansions in the `mtest` output.

The structure input file however is identical to that of `test1`, and once again it describes a `quad_mol` molecule interacting with a `dipole_mol` molecule, separated by a few Angstroms, with the molecules also rotated.

Running XMULTI in this directory should produce an output giving the energy of the two molecule system, and that energy should be nearly identical to that found in `test1`.

You can also try modifying the input files to e.g. give the interaction between an `oct_mol` molecule and a `quad_mol` molecule, and as long as the same change is made in both directories, the energies should be nearly identical.

This experiment demonstrates that the multipoles as calculated by XMULTI are consistent. That is, the same result is produced by a numerical rank  $n+1$  multipole created by the infinitesimal displacement of two rank  $n$  multipoles as given by an analytic multipole with the same multipole moments.

## Appendix I. Euler angles in XMULTI

XMULTI relates the lab coordinates to the body centred coordinates from

$$\mathbf{r}^l = \mathbf{R}\mathbf{r}^b + \Delta\mathbf{r}^{COM},$$

where  $\Delta\mathbf{r}^{COM}$  is the center of mass separation, and  $\mathbf{R}(\phi, \theta, \psi)$  is the orthogonal rotation matrix given by

$$\mathbf{R}(\phi, \theta, \psi) = \begin{bmatrix} \cos(\phi)\cos(\psi) - \sin(\phi)\cos(\theta)\sin(\psi) & \sin(\phi)\cos(\psi) + \cos(\phi)\cos(\theta)\sin(\psi) & \sin(\theta)\sin(\psi) \\ -\cos(\phi)\sin(\psi) - \sin(\phi)\cos(\theta)\cos(\psi) & -\sin(\phi)\sin(\psi) + \cos(\phi)\cos(\theta)\cos(\psi) & \sin(\theta)\cos(\psi) \\ \sin(\phi)\sin(\theta) & -\cos(\phi)\sin(\theta) & \cos(\theta) \end{bmatrix}$$