General Purpose Trajectory Analyser (GPTA) User Manual

Associate Professor Paolo Raiteri

August 19, 2021

Abstract

The General Purpose Trajectory Analyser (GPTA) is a program that I developed over the years to perform routine day-to-day tasks required for my research as a computational materials scientist. In particular, my main research tool is Molecular Dynamics simulations and I use GPTA on a daily basis to prepare the input structure and analyse the output trajectories. The tasks that GPTA is able to perform can be broadly divided into two categories; manipulation of the coordinates of individual structures and post processing of (large) trajectory files. Although GPTA has been rewritten multiple times, there are some legacy routines and there are some inconsistencies in style and appearance of the routines. It is provided here in the hope that it may be useful to others.

Contents

Disclaimer of Warranty

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

Limitation of Liability

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Contacts

If you want to report a bug please email include a detailed description of the problem you encounter to `gpta.help@gmail.com`. Please also include the command you used and a small coordinates file to reproduce to error.

Although you are not required to do so, the author would consider it a courtesy if you submit to them any changes you regard as being worthwhile adding to the code. The goal would be to keep the development of GPTA more or less centralized.

1 Obtaining and Installing the Program

GPTA can be downloaded from github (https://github.com/praiteri/GPTA). It is composed of a combination of FORTRAN and C/C++ files, hence it requires compatible C and FORTRAN compilers.

In the GPTA root directory there is Makefile, which should work as is on a standard Linux installation with the *gfortran* and *gcc* compilers. However, it should be relatively easy to modify it and adapt it for other operating systems and compilers.

In the simplest of cases, GPTA can be compiled by simply running

```
make serial
```

in the GPTA folder. The executable will be placed in a folder called `bin`, which will be created if required. In order to compile GPTA with the MPI parallelisation enabled one can simply run

```
make mpi
```

this will create a new folder for the MPI object files, `obj_mpi`, and an executable named `gpta_mpi.x`. Other actions available through the Makefile are

```
make clean     # deletes the serial object filess
make clean-mpi # deletes the MPI object files
make clean-all # deletes all object files
make distclean # deletes all object files and executables
```

The compilation of GPTA can be tailored by using command line variables, which are also used to link GPTA to external libraries, such as PLUMED and openMM.

```
make omp=n     # disables openMP parallisation
make dbg=y     # compiles GPTA in debug mode
make plumed=y  # enables the interface with PLUMED
make openmm=y  # enables the interface with openMM
make cuda=y    # enables the interface with openMM for GPU
```

1.1 GPTA doesn't work for my problem, what should I do?

For any moderately experienced programmer it should be easy enough to modify the code to implement a new type of property analysis or pre-processing tool. To make this task a bit easier, there is a sample routine `test.F90` where there is an alternative, albeit slower, implementation of the radial pair distribution function, where some of the core processing routines that are available in GPTA.

2 Main structure of the code

Before getting into the details of what GPTA can (and cannot) do, it is worth having a brief look at the core parts of the code, namely the calculation of the neighbours' list, the use of periodic boundary conditions and parallelisation.

2.1 Periodic Boundary Conditions

Although the most common, and tested, scenario where GPTA is used it to pre/post process coordinates from 3D periodic MD simulation, it should also work for non-periodic (cluster) calculations. In order to efficiently calculate the distances with periodic boundary conditions GPTA uses the standard approach to transform the distances in fractional coordinates using the metric matrix and its inverse, `hmat` and `hinv`, respectively

```
dij = pos(:,i) - pos(:,j)
sij = matmul(hinv , dij)
sij = sij - nint(sij)
dij = matmul(hmat , sij)
distance = sqrt(sum(dij*dij))
```

The `matmul` function is in-lined by the compiler when the `-O3` option is used, which makes its use as efficient as explicitly writing the matrix-vector multiplication. Moreover, to further speed up the calculation, for orthorhombic cells the above routine is replaced by one where only the diagonal elements of the metric tensor, `cell` are used

```
dij(1:3) = pos(1:3,i) - pos(1:3,j)
sij(1) = dij(1) / cell(1)
sij(2) = dij(2) / cell(2)
sij(3) = dij(3) / cell(3)
sij = sij - nint(sij)
dij(1) = sij(1) * cell(1)
dij(2) = sij(2) * cell(2)
dij(3) = sij(3) * cell(3)
distance = sum(dij*dij)
```

Although these methods for calculating distances with periodic boundary conditions are very efficient, they are also known to fail for small and skewed cells. Hence, it is possible to force GPTA to use search for the shortest distances between two atoms by looking at all possible distances between one atom and the 27 images of its neighbour and taking the shortest one, which indeed is the definition of "minimum image convention".

```
dij(1:3) = pos(1:3,i) - pos(1:3,j)
distance = sum(dij*dij)
lsafe_pbc_vec = dij
do i=-1,1
  do j=-1,1
    do k=-1,1
      if (i==0 .and. j==0 .and. k==0) cycle
      sij = dij + i*hmat(:,1) + j*hmat(:,2) + k*hmat(:,3)
```

```
          rtmp = sum(sij*sij)
          if (rtmp < distance) then
            distance = rtmp
            lsafe_pbc_vec = sij
          end if
        enddo
      enddo
    enddo
    distance = sqrt(distance)
```

This approach would however be extremely slow and unnecessary for large systems. Its use can however enforced using the *--define safedist* command and it could be useful for debugging purposes and very small systems.

2.2 Neigbours' List

GPTA has three algorithms to compute the neighbours' list, and it attempts to select the optimal one depending on the system size and cutoff distance for the list

- The standard "Verlet" method with a double look over the atoms
- The the "linked cell" algorithm, with ghost cells added around the system to avoid applying PBC while computing the distances.
- The "linked cell" algorithm, where the atoms' positions are projected onto the vector connecting the cells centres to reduce the number of distance calculations. This method requires sorting the projections and it is generally slower for short cutoff distances.

Although, the "linked cell method" is significantly faster than the " Verlet" method, it is possible to force GPTA to use it by using the *--define verlet* command. Information about the chosen algorithm is provided in the screen output.

```
...
-------------------------------------------------------------------------------------------------------------------------
Neighbours' list setup
...Algorithm.............................................: Linked Cell Sorting
......Number of cells....................................: 100 100 292
...Double list...........................................:       FALSE
...Cutoff radius.........................................:      3.0000
...Maximum number of neighbours..........................:          16
-------------------------------------------------------------------------------------------------------------------------
...
```

2.3 Timing

At the end of each run GPTA provides some information about the time spent in the calculations. It also provides a breakdown of the time spent in each action, plus the time spent in the calculations of the neighbours' list. The number in square brackets indicates the number of times the neighbours' list has been calculated.

```
...
-------------------------------------------------------------------------------------------------------------
Frames processing summary
...Number of frames read.............................:          20
...Number of frames processed........................:          11
-------------------------------------------------------------------------------------------------------------
Timing summary (seconds)
...Initialisation time...............................:      0.0189
...Total reading time................................:      0.0561
...Total time in actions.............................:      0.7550
......Total time in --test...........................:      0.7550
...Neighbours list time [n=11].......................:      0.2575
Total elapsed time...................................:      1.0900
-------------------------------------------------------------------------------------------------------------
```

2.4 Parallelisation

GPTA supports two levels of parallelisation, with openMP and MPI. The former is turned on by default and it is used mainly to speed up the calculation of the neighbours' list, which is the bottleneck for most calculations. MPI is instead used to distribute the processing of the trajectory frames across the available CPUs. In this case, the master CPU reads all the frames from the trajectory files and distributes them to the other CPUs for processing. If less than 5 CPUs are used the "reading" node also processes one frame itself, otherwise it is just used to distribute the frames to the "working" nodes. Hence, this type of

parallelisation if beneficial only if the number of frames to be processed is large compared to the number of CPUs used and if the processing time is shorter compared to the reading time, which may not be the case for large system where the coordinates are written in ASCII format rather than binary. At the moment only the actions that compute properties support MPI parallelisation, and no output of modified coordinates is allowed. There are a number of actions that work in parallel but, as mentioned above, this provides a significant speed up only if the reading time is significantly shorter than the processing time. A few actions, such as the radial pair distribution function, have a small level of openMP parallelisation. However, as mentioned before, in the majority of cases the calculation of the interatomic distances, *i.e.* the neighbours' list, is the rate limiting step. Here below you can see a couple of graphs of the time required to compute the neighbours's list for different cutoff radii and different number of CPUs on a 32 cores node with a total of 64 hyperthreads. The test were done for a 1,000 frames trajectory written in DCD format with 4,180 water molecules.

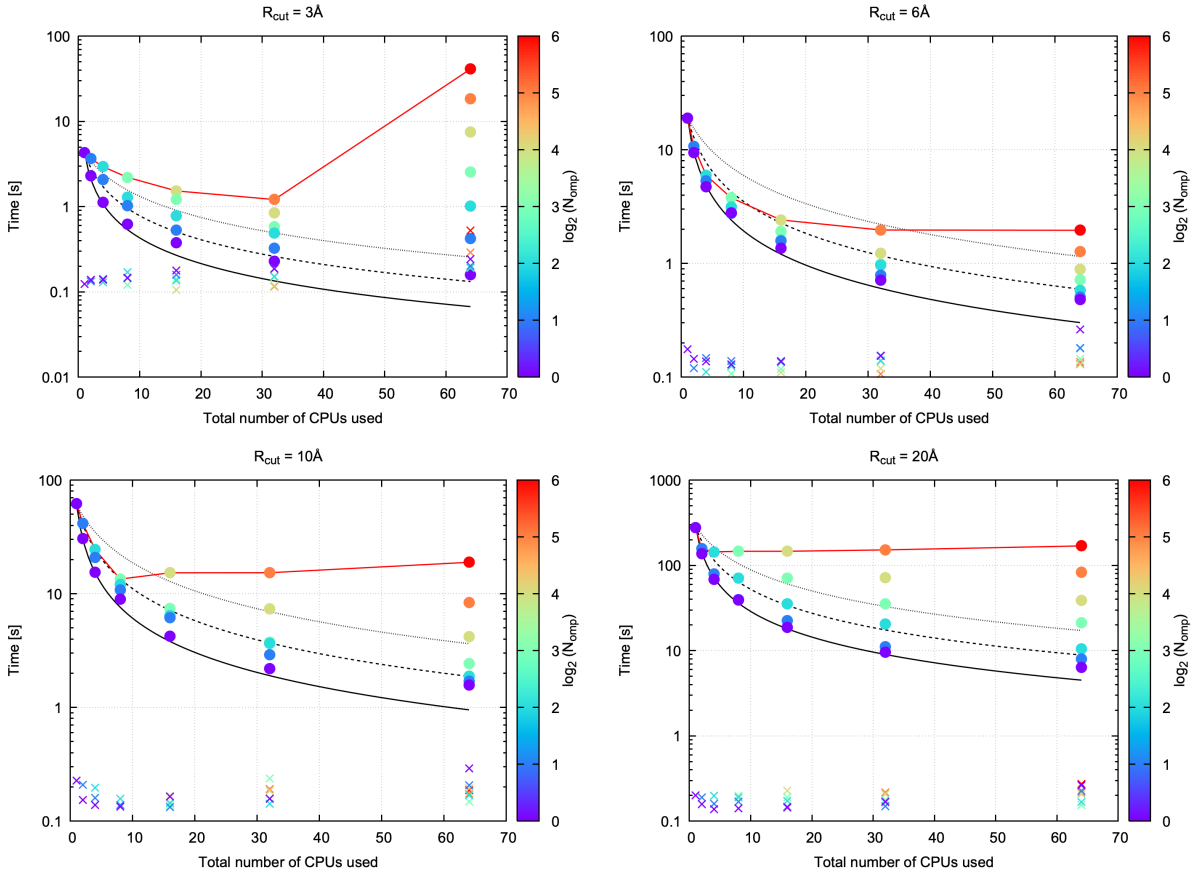

Figure 1: Execution time for the calculation of the neighbours' list with different cutoff radii using up to 64 processes and a different combination of openMP threads and MPI processes (circles). The crosses represent the time spent by the code to read the trajectory file. The symbols are coloured based on the number of openMP threads used. In order to make the color scale more readable we used $\log_2(N_{threads})$, *i.e.* the dots for 64 openMP threads are coloured in red while for 1 openMP thread are coloured in purple. The red line connects the points obtained using only the openMP parallelisation. The black lines indicate ideal scaling (solid), 50% efficiency (dashed) and 25% efficiency (dotted).

It is clear how the MPI parallelisation is more effective and, since each MPI processes work on a separate frame independently, it should scale almost linearly with the number or processes. On the other hand, there is little gain in using more than 8 openMP threads for the calculation of the neighbours' list, and it can actually be detrimental to the performances to use 32 or 64 threads. Hence, by default GPTA would use 8 openMP threads when run in serial.

Although scaling and performances are affected by the size of the system, MPI would always provide better performances for the calculation of properties. openMP is however the only option when manipulating the system coordinates or processing a small number of frames.

3 Commands Description

In the following sections we shall provide a description of all the available commands, which have been grouped according to whether they define global parameters for the simulations (setup commands), or they can be used to modify the input coordinates or to compute some system properties. Finally there will be a section for the commands that require external packages to be compiles separately, *i.e.* openMM and PLUMED.

First of all it is important to be aware of some general rules about the input syntax. GPTA takes all the instructions from the command line, and the "actions" are typically executed sequentially, *i.e.* in the order in which they appear on the command line. Action commands are preceded by a double dash "--", while settings and for each action are preceded by a "+". Actions are generally self-contained objects and they can be repeated multiple times on the command line, *e.g.* to compute the same property for two different groups of atoms. Although every effort has been made to make all the actions that change the atoms' properties and positions compatible with each other it is always safer to perform such action at the time, maybe with a batch script. Unless otherwise stated, each argument of an action flags should be provided as a comma separated list, this will be more clear by looking at the examples provided below.

In the description of the flags allowed for the available actions we will provide the default value of the flag within square brackets. If no default value is set we will provide a short description of the expected value in brackets, *e.g. +rcut [3.0]* or *+f [filename]*

3.1 Setup commands

In this section we descriribe the commands that are executed only once at the beginning of the job, regardless of where they appear on the command line. These commands allow for

- selecting the number of openMP threads to use
- redirecting the screen output to a file
- overriding the definition of a few parameters that are used

3.1.1 Number of openMP Threads

This command sets the number of OMP threads to be used in the calculation of the neighbours' list. By default all available threads are used, up to a maximum of 8; unless the MPI version of the code is used, in which case only one openMP thread is used.

Command: *--nt [number of threads]*

Examples:
```
gpta.x --nt 4 --i coord.pdb
mpirun -np 8 gpta_mpi.x --nt 4 --i coord.pdb
```

3.1.2 Screen Output

This command redirects the standard output to a file, useful for batch processes in script files.

Command: *--log [logfile]*

Examples:
```
gpta.x --i coord.pdb --log gpta.log
```

3.1.3 Parameters Definition

This command can be used to defined the value of some of the program variables.

Command: *--define [variable] [value]*
This is the list of variables that currently can be set:

seed	seed to initialise the random number generator
rscale	scaling factor to define the maximum bond length
safedist	distances wil be computed with a loop over the 27 neighbour cells
verlet	forces the Verlet algorithm to be used in the calulation of the distances between atoms
rcov	covalent radius of an atom for the topology calculation
bond X,Y=2	maximum bond length between X and Y for the topology calculation

Examples:

```
gpta.x --define seed 123
gpta.x --define rscale 1.24
gpta.x --define safedist
gpta.x --define verlet
gpta.x --define rcov Ca=1.9 rcov Na=0.0
gpta.x --define bond X4,Y4=2. --i coord.pdb --top
```

3.2 Input, output and frame selection

In this section we discuss the commands that can be used to read and write coordinates files, and how it is possible to select which frames to process.

3.2.1 Input Coordinates

GPTA can read the atomic coordinates in different file formats. The coordinates' format is normally inferred from the file extension

.xyz → XYZ file

.pdb → PDB file

.gin → GULP input file

.cif → Crystallographic Information File (CIF) file

.dcd → CHARMM DCD file

.dcd2 → CHARMM DCD file

.gau → Gaussian output file

.lmp → LAMMPS coordinates file

.lmptrj → LAMMPS custom trajectory file

.gro → GROMACS GRO file

.xtc → GROMACS XTC file

.trr → GROMACS XTC file

but it can be overridden by appending the canonical file extension (listed above) to the action flag, like in the last example below. For the DCD, XTC and XTC binary formats that do not contain the atoms' label it is mandatory to read first an XYZ or a PDB file. Multiple files can be specified sequentially or the command can be repeated multiple times. Currenlty the flags must follow the file names, and they are globally applied even if more than one *--i* command are used.

There are two routines that can read DCD files, by default a faster implementation is used, which however may fail if the trajectory file is corrupted. An alternative routine can be used, where careful checks on the integrity of the files are made, can be selected with the type "dcd2". This routine was taken from the libdcdfort library written by James W. Barnett (https://github.com/wesbarnett/libdcdfort).

Currently, GPTA reads only one structure per file in the "cif" and "gin" formats.

The "xyz" file type supports also the "extended XYZ format" used by the Atomic Simulation Environment (https://wiki.fysik.dtu.dk/ase). For example the simulation box and the atomic charges can be included as

```
800
Lattice="5.44 0.0 0.0 0.0 5.44 0.0 0.0 0.0 5.44" Properties=species:S:1:pos:R:3:charge:R:1
Na        0.00000000      0.00000000      0.00000000   1.0
Cl        1.36000000      1.36000000      1.36000000  -1.0
...
```

– –

Command: *--i [filename1] [filename2] ...*

List of flags: *+cell [simulation cell]*
defines the shape of the simulation cell; 1, 3, 6 or 9 numbers are expected

1 : cubic box

3 : orthorhombic box defined as $|a|$ $|b|$ $|c|$

6 : triclinic box defined as $|a|$ $|b|$ $|c|$ α β γ

9 : triclinic box defined from the cell matrix $\vec{a}$ $\vec{b}$ $\vec{c}$

The cell definition read from the coordinates file, if present, will be overridden.

+nm
assumes the input coordinates are in nm. This flag should not be used for coordinates files that are by definition in nm, *i.e.* `.gro` and `.xtc`, otherwise the Å to nm conversion factor will be applied twice.

+bohr
assumes the input coordinates are in bohr.

Examples:

```
gpta.x --i coord.pdb
gpta.x --i coord.pdb traj.dcd
gpta.x --i coord.pdb --i traj.dcd
gpta.x --i coord.pdb +cell 14.
gpta.x --i coord.pdb +cell 14.0,15.0,16.0
gpta.x --i coord.pdb +cell 14.0,15.0,16,90.0,90.0,120.0
gpta.x --i coord.pdb +cell 14.0,0.0,0.0,,0.0,15.0,0.0,,0.0,0.0,16.0
gpta.x --ipdb coordinates.xyz
```

Screen output: ...

```
-------------------------------------------------------------------------------------------------------------
Opening input coordinates files
...pdb file.....................................: coord.pdb
...dcd file.....................................: traj.dcd
-------------------------------------------------------------------------------------------------------------
Initial cell
...Cell vector A................................:     14.3000         0.0000         0.0000
...Cell vector B................................:      0.0000        14.3000         0.0000
...Cell vector C................................:      0.0000         0.0000        14.3000
...Cell lengths.................................:     14.3000        14.3000        14.3000
...Cell angles..................................:     90.0000        90.0000        90.0000
...Volume.......................................:   2924.2070
-------------------------------------------------------------------------------------------------------------
System composition
...Number of atoms found........................:         298
......Number of unique species Ca...............:           1
......Number of unique species H................:         198
......Number of unique species O................:          99
-------------------------------------------------------------------------
...Total mass (g/mole)..........................:   1824.0600
...Density (g/cm^3).............................:      1.0358
...Total charge.................................:      0.0000
-------------------------------------------------------------------------------------------------------------
Frames processing summary.......................:        3779
...Number of frames read........................:        3779
...Number of frames processed...................:        3779
-------------------------------------------------------------------------------------------------------------
...
```

3.2.2 Custom Selection of Frames

When reading a multi-frame trajectory it is possible to select a subset of configurations for processing. This could be done via a list of comma separated list of individual frames or via a loop-type definition, B:(E):(S). By default the end frame and stride are set to the largest single precision integer and to 1, respectively, and they can be omitted. If only one frame is required it is possible to use the command *--frame N*.

Command: *--frames [indices]*

Examples:
```
gpta.x --i coord.pdb traj.dcd --frames 10
gpta.x --i coord.pdb traj.dcd --frames 10:100
gpta.x --i coord.pdb traj.dcd --frames 10:100:5
gpta.x --i coord.pdb traj.dcd --frames 1,2
gpta.x --i coord.pdb traj.dcd --frame 123
```

Screen output: ...
```
------------------------------------------------------------------------------------------------------------
First frame to process.............................:          10
Last frame to process..............................:         100
Stride for processing frames.......................:           5
------------------------------------------------------------------------------------------------------------
...
```

3.2.3 Skiping Frames

Alternative to the selection of frames with the *--frames* command it is possible to just specify the stride at which the frames have to be processed. The first frame is always processed. This is mostly a legacy command and it will probably be removed in the future.

Command: *--skip [N]*

Examples:
```
gpta.x --i coord.pdb traj.dcd --skip 7
```

Screen output: ...
```
------------------------------------------------------------------------------------------------------------
First frame to process.............................:           1
Last frame to process..............................:  2147483647
Stride for processing frames.......................:           7
------------------------------------------------------------------------------------------------------------
...
```

3.2.4 Process Only The Last Frame

It is often of interest to extract only the last frame of a trajectory, without knowin *a priori* the total number of frames in the input file. This can be achieved with this command, without reading the input file(s) twice.

Command: *--last*

Examples:
```
gpta.x --i coord.pdb --last
```

Screen output: ...
```
-----------------------------------------------------------------------------------------------------
Processing only the last frame
-----------------------------------------------------------------------------------------------------
...
```

3.2.5 Output Coordinates Files

This action writes the (manipulated) manipulated coordinates to a file. Currently these are the available coordinates outpu formats:

.xyz → XYZ file

.pdb → PDB file

.pdb2 → PDB file with connectivity at the end

.gin → GULP input file

.gin2 → GULP input file with fractional coordinates

.dcd → CHARMM DCD file (binary)

.psf → CHARMM PSF input file

.arc → TINKER format

.lmp → LAMMPS input file

.lmptrj → LAMMPS trajectory input file (dump custom)

.xtc → GROMACS xtc trajectory file (binary)

Command: *--o [filename1]*

List of flags: *+append*
forces to coordinates to be appended to the existing output file. No checks are currently made to ensure the the coordinates in the existing file and the new output have the same format

+bohr [check]
forces the output coordinates to be in Bohr.

+f [forcefield]
this flag is available only for the LAMMPS format. defines the file with the forcefield to generate a coordinates file for LAMMPS. It requires *--top* to compute the system's topology.

+ignore [check]
this flag is available only for the LAMMPS format. It can be used to ignore missing terms in the forcefield file. Because missing terms could indicate either an unphysical topology or errors in the force field file, extreme care should be paid when using this flag. If this flag is not used a warning message is printed on the screen and the user has to manually ignore all the missing terms, or stop the program and fix the problem in the coordinates file or in the force field file. This flag requires one argument that specifies which term(s) can be ignored:

- *check*
- *all*
- *bonds*
- *angles*
- *torsions*

– *impropers*

Multiple options can be specified in a comma separated list.

+map [labels] [types]
this flag is available only for the LAMMPS trajectory file. It provides a 1 to 1 map between the atoms' labels in the file and the LAMMPS types given in the forcefield. If it is not present the atom types are assigned a integer value using the atoms' label in the order they appear in the input file.

Examples:

```
gpta.x --i coord.pdb --o coord.xyz
gpta.x --i coord.pdb --o test.pdb +append
gpta.x --i coord.pdb --opdb test +bohr
gpta.x --i coord.pdb --top --o coord.lmp +f forcefield.lmp
gpta.x --i coord.pdb --o coord.lmptrj +map C4,O4,Ca 3,4,5
```

Screen output: ...

```
-------------------------------------------------------------------------------------------------------
Writing coordinates in xyz format.................: coord.xyz
-------------------------------------------------------------------------------------------------------
...
```

3.3 Description of Actions That Modify the System

In this section we describe all the actions that manipulate the atomic coordinates. All these actions can be used only in with the serial executable or one MPI process.

3.3.1 Molecular Connectivity and Topology

This action computes the atoms' connectivity and molecular topology of the system. Although strictly speaking this action does not change the atomic coordinates, it is often a pre-requisite for most of the actions discussed in this and the following sections. By default the topology is determined using a distance criterion for the definition of the covalent bonds. Two atoms are considered bonded if

$$r_{ij} < R_s(r_i^{cov} + r_j^{cov})$$

where r_{ij} is the distance between species i and j, r^{cov} is the covalent radius of the element as defined in the `elementsModule.F90` file and R_s is a scaling factor set to 1.15. The scaling factor and special bond lengths can be specified with the *--define* command.

Alternatively, the molecular composition of the system can be defined by providing the list of atoms' names in each molecules. Each molecule is defined by a comma separated list of atoms, and the atoms must be in the same order as they appear in the coordinates file.

– –

Command: *--top*

List of flags: *+ion [list of species]*
ensures that no covalent bond in created with the listed species.

+def [molecule1] [molecule2] ...
with this flag the topology is not built from the neighbours list but simply using the atoms' labels. Each molecule has to be given as an individual argument, with the atoms belonging to each molecule provided in a comma separated list. All molecules in the input file(s) must have the atoms in the same order as they appear on the command line to be correctly identified.

+update
forces the topology the be updated at every frame. Using this flag drastically reduces the performances of the code.

+rebuild
forces the molecule that are broken across the PBC to be rebuilt.

+check
checks if any molecules are broken across the PBC and rebuilds them

+reorder
asks for the atoms in the system to be reordered and grouped by molecule. The program will also attempt to recognise equivalent molecule's types where the atoms are simply in different order.

Examples:

```
gpta.x --i coord.pdb --top
gpta.x --i coord.pdb --top +ion Na Cl
gpta.x --i coord.pdb --top +def Ca O,H,H
gpta.x --i coord.pdb --top +update
gpta.x --i coord.pdb --top +rebuild
gpta.x --i coord.pdb --top +reorder
```

Screen output: ...

```
-----------------------------------------------------------------------------------------------------------
Topology information
...Number of molecules found.............................:         520
...Number of unique molecules found......................:           2
....Molecule type M1
......Number of molecules................................:           1
......Concentration [M]..................................:      0.1081
......Number of atoms in molecule........................:           1
......Atoms types in molecule............................: Ca
......Number of atoms in molecule........................:           1
......Molecular charge...................................:      0.0000
....Molecule type M2
......Number of molecules................................:         519
......Concentration [M]..................................:     56.0923
......Number of atoms in molecule........................:           3
......Atoms types in molecule............................: O2,H2,H2
......Number of atoms in molecule........................:           3
......Molecular charge...................................:      0.0000
-----------------------------------------------------------------------------------------------------------
...
```

3.3.2 Add particles

This action adds new particles to a frame. It can be used to add solute/solvent atoms or molecules, or to add dummy particles to the system. The new species are added randomly to the cell. This action is a wrapper for four different actions which are called by the following commands, which accept different flags and work in slightly different ways;

- --add solvent
  This command adds a solvent to the input frame either as elements from the periodic table or as molecules read from a file.
- --add solute
  This command adds a solute to the input frame either as elements from the periodic table or as molecules read from a file.
- --add centre
  This command adds a dummy atom to a molecule, *e.g.* the fourth site in TIP4P water.
- --add dummy
  This command adds the geometric centre (or the centre of mass) to a molecule as an extra particle, which can be used for visualisation purposes or for computing other properties related to the position of the centre of mass.

In particular, for the first two commands, after the new particles have been added, the overlapping solvent molecules are removed. Hence, in the case of *--add solvent* the number of molecules added will be less than the number specified in the input, while in the case of *--add solute* some of the molecules that were initially in the system will be removed. No molecules will be removed after the other commands.

This action can also be used to create a system from scratch, *i.e.* when no *--i* command is used.

— —

Command: *--add solvent / -add solute*

Prerequisite: *--top*

List of flags: *+atom [species]*
defines the name of mono-atomic species to add, it doesn't have to be an element from the periodic table..

+mol [filename]
defines the name of the file that contains the coordinates of the molecule to add to the system.

+n [1]
sets number of species to add.

+rmin [1.5]
sets the cutoff distance to remove overlapping solvent molecules

+box [cell]
defines the size and shape of the region of space to be filled by the new molecules. If no cell was present, this will be the new global cell. See the reading coordinates action (*--i*) for how the cell can be specified.

sphere [radius]
define the radius of the sphere to be filled by the new molecules.

+origin [0,0,0]
defines the origin of the box or the centre of the sphere to be filled by the new molecules.

Examples:

```
gpta.x --i coord.pdb --top --add solute +atom Na,Cl +n 2,2 +rmin 4.0
gpta.x --i coord.pdb --top --add solvent +mol benzene.pdb +n 5 +rmin 5.0 --o new.pdb
gpta.x --add solvent +mol h2o.pdb +box 12. +n 57 +rmin 2 --top --o water.pdb
```

Command: *--add centre*

Prerequisite: *--top*

List of flags: *+mol [molecule ID]*
defines the name of the molecule whose centre of mass will be calculated

+i i1,i2...
defines the indices of atoms in the molecules to be used for the calculation of the centre of mass. By default all atoms are used.

Examples:

```
gpta.x --i coord.pdb --top --add centre +mol M1
gpta.x --i benzene.pdb --top --add centre +mol M1 +i 1,3,5,7,9,11
```

Command: *--add dummy*

Prerequisite: *--top*

List of flags: *+mol [molecule ID]*
defines the name of the molecule where the dummy site will be added. The dummy particle is added along the bisector of the HOH bond.

+i i1,i2,i3
defines the indices of the O and H atoms that will be used to compute the bisector. The O atoms is the first index.

+dist [0.1]
defines the distance along the bisector where the dummy site will be added.

Examples:
```
gpta.x --i water.pdb --top --add dummy +i 1,2,3 +dist 0.1577 --o tip4p_ice.pdb
```

Screen output: ...
```
-----------------------------------------------------------------------------------------------------------
Add Atoms
...Size of the box to be filled.....................:     10.0000        10.0000        10.0000
...Angles of the box to be filled...................:     90.0000        90.0000        90.0000
...Origin of the box to be filled...................:      0.0000         0.0000         0.0000
...New moleculed from file..........................: h2o.pdb
...Number of molcules to add........................:          33
...Minimum distance between the molecules...........:      2.0000
-----------------------------------------------------------------------------------------------------------
...
```

3.3.3 Delete Atoms

This action deletes the selected atoms from the input configuration. The atoms to be removed can be selected either by their label (*+s*), their indices (*+i*) or by the type of molecule they belong to (*+mol*). Mixed selections, *e.g.* based on labels and indices, should be possible within one command, but the command can also be repeated.

Command: *--delete*

List of flags: *+s [species]* or *+i [indices]* or *+mol [molecule name]*
selects the species to be deleted using the atoms' names or their indices or the name of the molecules they belong to. The species selection is done by providing a comma separated list of atom names. The index selection can be done by providing a comma separated list of indices or in loop format B:E:(S), where the stride is optional. The *+mol* flag requires *--top* flag.

Examples:
```
gpta.x --i coord.pdb --delete +s O
gpta.x --i coord.pdb --delete +s O,H
gpta.x --i coord.pdb --delete +i 1:30
gpta.x --i coord.pdb --delete +i 1:30:3
gpta.x --i coord.pdb --delete +i 1,34,6789
gpta.x --i coord.pdb --top --delete +mol M2
```

Screen output: ...
```
-----------------------------------------------------------------------------------------------------------
Removing atoms
...Atoms selected by label.........................: Na
...Deleting selected atoms
-----------------------------------------------------------------------------------------------------------
...
-----------------------------------------------------------------------------------------------------------
Topology information
...
....Molecule type M2
...
-----------------------------------------------------------------------------------------------------------
Removing atoms
...Atoms selected by molecule......................: M2
...Deleting selected atoms
-----------------------------------------------------------------------------------------------------------
...
```

3.3.4 Translate the Simulation Centre of Mass

This action can be used to traslate the centre of mass of selected atoms to a chosen position. It is mostly useful to prepare initial configurations and for focussing the visualisation in post-processing.

Command: *--fixcom*

List of flags: *+s [species]* or *+i [indices]* or *+mol [molecule name]*
select the species to be used in the calculation of the centre of mass using the atoms' names or their indices or the name of the molecules they belong to. The species selection is done by providing a comma separated list of atom names. The index selection can be done by providing a comma separated list of indices or in loop format B:E:(S), where the stride is optional. The *+mol* flag requires *--top* flag.

+centre
shift the centre of mass of the selected atoms to the centre of the simulation cell

+pos [position]
shift the centre of mass of the selected atoms to the specified position

Examples:
```
gpta.x --i coord.pdb --fixcom +s Ca +centre
gpta.x --i coord.pdb --fixcom +mol M1 +centre
gpta.x --i coord.pdb --fixcom +s Ca +pos 1.5,2.5,3.5
```

3.3.5 Create Mirror Image of the Simulation Cell

This action creates a mirror image of the simulation cell in one of the cartesian directions.

Command: *--mirror*

Prerequisite: *--top*

List of flags: *+x*
selects the x direction for mirroring the cell.

+y
selects the y direction for mirroring the cell.

+z
selects the z direction for mirroring the cell.

Examples:
```
gpta.x --i coord.pdb --mirror +z --out new.pdb
```

Screen output: ...
```
-------------------------------------------------------------------------------------------------------
...Mirroring the cell
-------------------------------------------------------------------------------------------------------
...
```

3.3.6 Wrap Molecules Inside the Simulation Cell

This action wraps the atoms back into the simulation cell. If the molecular topology has been calculated, the molecules are kept intact.

Command: *--pbc*

Examples:
```
gpta.x --i coord.pdb --pbc
```

Screen output: ...
```
-----------------------------------------------------------------------------------------------------------
...Applying periodic boundary conditions...........: [0:1]
-----------------------------------------------------------------------------------------------------------
...
```

3.3.7 Unwrap Molecules from Trajectory

This action unwraps the atoms in the system to produce a continuous trajectory, useful to compute the self diffusion coefficient. If the molecular topology has been calculated, the molecules are kept intact.

Command: *--unwrap*

Examples:
```
gpta.x --i coord.pdb --unwrap
```

Screen output: ...
```
-----------------------------------------------------------------------------------------------------------
...Unwrapping trajectory
-----------------------------------------------------------------------------------------------------------
...
```

3.3.8 Replicate Simulation Cell

This action replicates the input structure along the three crystallographic directions.

Command: *--repl [replicas]*

Prerequisite: *--top*

Examples:
```
gpta.x --i coord.pdb --repl 2,2,1
gpta.x --i coord.pdb --repl 1 2 3
```

Screen output: ...
```
...
```

3.3.9 Remove Overlapping Molecules

This action removes overlaps between molecules by deleting the molecule that comes second in the coordinates files. If the topology is computed before this action the entire molecule is removed when any of its atoms overlaps with another molecule.

Command: *--noclash*

List of flags: *+rmin [0.5]* sets the minimum distance to define when atoms are considered to be overlapping.

+dry
performs a dry run to give information about how many molecules would be considered overlapping with different values for *+rmin*. The values of *+rmin* that are tested are: 0.1Å, 0.5Å and 1Å, plus the value chosen with the *+rmin* flag.

Examples:
```
gpta.x --i coord.pdb --noclash +rmin 0.75
gpta.x --i coord.pdb --top --noclash +rmin 0.75 +dry
```

3.3.10 Rescale the Simulation Cell

This action deforms the cell to a new shape and rescales the internal coordinates. At the moment the system topology is ignored, which means that the molecules will be stretched after the rescaling.

Command: *--rescale*

List of flags: *+cell [cell]*
defines the size and shape of the region of space to be filled by the new molecules. If no cell was present, this will be the new global cell. See the reading coordinates action (*--i*) for how the cell can be specified.

Examples:
```
dmap3D.tex:gpta.x --i coord.pdb --rescale +cell 10
dmap3D.tex:gpta.x --i coord.pdb --rescale +cell 10,10,20
dmap3D.tex:gpta.x --i coord.pdb --rescale +cell 10,10,10,90,90,90
dmap3D.tex:gpta.x --i coord.pdb --rescale +cell 10,0,0,0,10,0,0,0,10
```

Screen output: ...
...

3.3.11 Substitute Molecules with a Different One

This action can be used to replace selected molecules with a different molecule read from a file. Alternatively this action could be used to simply compute the RMSD of the selected molecules with respect to a reference structure. At the moment the RMSD is simply written to a file, but it would be easy to compute its average or distribution. Currently only molecules smaller than 50 atoms can be replaced.

Command: *--replace*

List of flags: *+f [filename]*
defines the name of the file containing the new molecules.

+mol [mol ID]
defines the ID of the molecules that will be replaced.

+ir [indices]
defines the indices of the atoms in the reference molecules that are to be used in the alignment.

+is [indices]
defines the indices of the atoms in the current (system) molecules that are to be used in the alignment.

+all
this flag can be used in place of the previous two and implies that the current and reference molecules have the same number of atoms and they are all to be used for the alignment.

+rmsd [output file]
when this flag is used no substitution if performed, but the RMSD for each molecule is written out.

Notes: If the molecules' substitution is performed on a multi-frames input, the topology must be updated at every frame, *i.e. --top +update* must be used. The topology is also recalculated after the substitution.

Although the selection in principle works for multiple molecule's types, this has never been tested.

Examples:
```
gpta.x --i co3.pdb --top --subs +mol M1 +is 1,2,3,4 +ir 1,4,3,2 +f hco3.pdb --o new.pdb
gpta.x --i slab.pdb --top --subs +mol M1 +all +f co3.pdb +rmsd rmsd.out
```

Screen output: ...
```
-------------------------------------------------------------------------------------------------------------
Replace Molecules Action
...Target molecule.................................: M1
...New molecule read from..........................: hco3.pdb
...Atoms used for the alignmen
......Current molecule.............................: 1 2 3 4
......New molecule.................................: 1 4 3 2
-------------------------------------------------------------------------------------------------------------
Running required action............................: topology
...Action flags....................................: +update
-------------------------------------------------------------------------------------------------------------
...
```

3.3.12 Define Atoms' Properties

This action overrides some of the basic properties of the atoms in the system that can be used by other actions.

Command: *--set*

List of flags: *+s [species]* or *+i [indices]* or *+mol [molecule names]*
select the species to be used in this action using the atoms' names or their indices or the name of the molecules they belong to. The species selections are done by providing a comma separated list of atom names. The index selections can be done by providing a comma separated list of indices or in loop format B:E:(S), where the stride is optional. The *+mol* flag requires *--top* flag.

+l [new labels]
defines the new atom names (labels) for the selected atoms.

+q [new charges]
defines the new atom charges for the selected atoms.

+tq [TINKER types]
defines the TINKER types for the selected atoms to be used in the *--amoeba* action

Notes: The number of new labels, charges or TINKER types provided must be either:

- consistent with the number of different species that have been selected with *+s*
- one if the atoms have been selected with *+i*
- consistent with the number of atoms in the selected molecules if the *+mol* flag has been used

Examples:

```
gpta.x --i coord.pdb --top --set +s O2,H2 +l OW,HW
gpta.x --i coord.pdb --top --set +i 1:300:3 +l OW
gpta.x --i coord.pdb --top --set +mol M1 +l OW,HW,HW
gpta.x --i coord.pdb --top --set +s O2,H2 +tq 1,2
```

Screen output: ...

```
-------------------------------------------------------------------------------------------------------------
Modify atoms' Attributes
...Atoms selected by label.....................: O2 H2
...Renaming selected atoms as..................: OW HW
-------------------------------------------------------------------------------------------------------------
...
-------------------------------------------------------------------------------------------------------------
Modify atoms' Attributes
...Atoms selected by molecule..................: M1
...Renaming selected atoms as..................: OW HW HW
-------------------------------------------------------------------------------------------------------------
...
-------------------------------------------------------------------------------------------------------------
Modify atoms' Attributes
...Atoms selected by index.....................: 1:300:3
...Renaming selected atoms as..................: OW
-------------------------------------------------------------------------------------------------------------
...
-------------------------------------------------------------------------------------------------------------
Modify atoms' Attributes
...Atoms selected by label.....................: O2 H2
...Setting TINKER types to.....................: 1 2
-------------------------------------------------------------------------------------------------------------
...
```

3.3.13 Shift Atoms

This action shifts the atoms in the system by a fixed displacement. The atoms are not modified in any other way, *e.g* by rewrapping them into the simulation box.

Command: *--shift [displacement]*

Examples:

```
gpta.x --i coord.pdb --shift 0.0,10.5,-12.1
```

Screen output: ...

```
-------------------------------------------------------------------------------------------------------------
Translating atom's coordinates by..............:      0.0000        10.5000       -12.1000
-------------------------------------------------------------------------------------------------------------
...
```

3.3.14 Create Surface

This action creates a new system unit cell orientated with the chosen Miller index parallel to the z axis. The output file contains all the possible surface cuts across the centre of mass of each molecule present in the cell. The structures are all still 3D periodic and give the same X-ray (*--xray*) scattering. Depending on the Miller indices that are chosen the output structure will be a supercell of the initial system. Note that the molecules must not be broken across the periodic boundary conditions for this action to work properly. If the atoms' chrges are present the cell dipole normal to the surface is also provided.

Command: *--surface*

Prerequisite: *--top*

List of flags: *+out [surface.pdb]*
defines the output file name.

+hkl [indices]
selects the Miller indices of the surface to be create.

Examples:
```
gpta.x --i coord.pdb --surface +hkl 1,0,4
```

Screen output: ...
```
-----------------------------------------------------------------------------------------------------------
Create Surface
...Miller indices.............................: 1 0 4
-----------------------------------------------------------------------------------------------------------
...
```

3.3.15 Cluster Extraction

This action extracts a spherical cluster of radius R around the selected atom. If the molecular topology has been calculated prior to this command the whole molecule is selected when at least of of its atoms is within the cutoff radius.

— —

Command: *--test*

List of flags: *+out [cluster.xyz]*
defines the root of the output file name. The output is currently only in XYZ format and one file is created for each frame processed.

+i [index]
selects the atom at the centre of the cluster.

+r [0.0]
sets the radius of the cluster.

Examples:
```
gpta.x --i coord.pdb --cluster +i 12 +r 6.0 +out clusters.xyz
```

Screen output: ...
```
-----------------------------------------------------------------------------------------------------------
Extract Clusters Action
...Central atom...............................:          12
...Radius.....................................:      6.0000
-----------------------------------------------------------------------------------------------------------
...
```

---

4 Description of Actions That Compute Properties

In this section we describe all the actions that compute properties based on the atomic coordinates. Most of these actions can be used with MPI.

— —

4.0.1 Density Profile (1D)

This action computes the 1D density profile along one of the crystallographic directions. The density profile can be computed for a subset of atoms that can be selected either by their label (*+s*), their indices (*+i*) or by the molecule they belong to (*+mol*). Mixed selections, *e.g.* based on labels and indices, are not allowed within one command. The command can be repeated.

Command: *--dmap1D*

List of flags: *+out [dmap1D.out]*
defines the output file name.

+s [species] or *+i [indices]* or *+mol [molecule name]*
select the species to be used in the calculation using the atoms' names or their indices or the name of the molecules they belong to. The species selection is done by providing a comma separated list of atom names. The index selection can be done by providing a comma separated list of indices or in loop format B:E:(S), where the stride is optional. The *+mol* flag requires *--top* flag.

+x
sets the x direction for the calculation of the 1D density profile.

+y
sets the y direction for the calculation of the 1D density profile.

+z
sets the z direction for the calculation of the 1D density profile.

+nbin [100]
defines the number of bins for the 1D density profile.

Examples:
```
gpta.x --i coord.pdb --dmap1D +x +s O
gpta.x --i coord.pdb --dmap1D +z +s O +out dmap.out
gpta.x --i coord.pdb --dmap1D +z +s Ca +nbin 200 +out dmap.out
gpta.x --i coord.pdb --dmap1D +z +i 1:300:3 +out dmap.out
```

Screen output: ...
```
-----------------------------------------------------------------------------------------------------------
Computing 1D density profile
...Output file.....................................: dmap.out
...Atoms selection for --dmap1D
......Selection command............................:
......Atoms selected in the first group............:        520
-----------------------------------------------------------------------------------------------------------
...
```

File output:
```
# Position  | Density [atom/angstrom^3]
    0.4320800000E+00     0.1376089546E-01
    0.1296240000E+01     0.4867850937E-02
...
```

4.0.2 Density Map (2D)

This action computes the 2D density map projected on an arbitrary plane defined by its Miller indices. The location of the plane in the cell is chosen by defining the origin of the axes of the new cell. The calculation is done by computing the surface unit cell, normal to the defined crystallographic plane, and projecting the selected atoms onto it. Only the atoms within the selected thickness are considered. Because the projection plane is defined by the Miller indices, GPTA searches for the corresponding surface unit cell to use for the grid. Although the smallest surface unit cell is chosen, its orientation in the cartesian space is not unique, hence care must be paid when interpreting the results. However, it is also possible to manually define a new cell and in that case the density map is computed on the (0,0,1) plane. Depending on the orientation of the plane, portions of the surface cell may not contain any atoms, however it is possible to replicate the system to fill the space and have a complete 2D map (*+fill*).

Command: *--dmap2D*

List of flags: *+out [dmap2D.out]*
defines the output file name.

+s [species] or *+i [indices]* or *+mol [molecule name]*
select the species to be used in the calculation using the atoms' names or their indices or the name of the molecules they belong to. The species selection is done by providing a comma separated list of atom names. The index selection can be done by providing a comma separated list of indices or in loop format B:E:(S), where the stride is optional. The *+mol* flag requires *--top* flag.

+hkl [0,0,1]
sets the Miller indices of the plane to project the atoms' density onto.

+cell [hmat]
allows for defining a new system cell via the use of three three vectors, that must for a right-handed set. These vectors will become the new new metric matrix ($[\vec{a}, \vec{b}, \vec{c}]$) for the system and the 2D density map will be calculated on the (0,0,1) plane.

+origin [0.d0,0.d0,0.d0]
defines the origin of the 2D density map.

+thick [1.d0]
sets the thickness of the "slice" use for the projection.

+nbin [100,100]
sets the number of bins for the density map.

+fill [0]
sets the number of replicas to use for filling the surface unit cell. This flag is used to replicate the system to have a 2D density map on the entire surface unit cell. The number indicates the number of replicas in the positive and negative directions of the cartesian axes. If 0 is chosen only the initial cell is used, if 1 set the program uses 27 replicas, etc. The larger the number the slower the calculation becomes.

Examples:

```
gpta.x --i coord.pdb --dmap2D +s O2 +hkl 1,0,0 +origin 1,0,0 +nbin 50,50 +out dmap2D.out
gpta.x --i coord.pdb --dmap2D +s O2 +hkl 1,1,0 +origin 46.12,0,0  +nbin 50,50 +thick 1.
gpta.x --i coord.pdb --dmap2D +s O2 +hkl 1,1,0 +origin 0,0,0 +thick 1. +fill 1
gpta.x --i coord.pdb --dmap2D +s O2 +cell 46.12,0,0,-30.7036,44.9762,0,0,0,86.4160
```

Screen output: ...

```
-------------------------------------------------------------------------------------------------------
Compute 2D density map in a plane
...Miller indices of the plane.....................: 1 1 0
...Surface cell origin.............................:     46.1200         0.0000         0.0000
...Surface vector A................................:    -30.7036        44.9762         0.0000
...Surface vector B................................:     -0.0000        -0.0000        86.4160
...Slice thickness.................................:      1.0000
...Number of replicas to fill surface cell.........:           0
...Number of bins..................................: 50 50
...Atoms selection for --dmap2D
......Selection command............................: +s O2
......Atoms selected in the first group............:        3137
-------------------------------------------------------------------------------------------------------
...
```

File output:

```
# X [angstrom] | Y [angstrom] | Density [atom/angstrom^3]
      0.47545      0.86416      0.00050
      0.95090      0.86416      0.00150
...
```

4.0.3 Density Map (3D)

This action computes the 3D density map for the selected atoms in a portion of the system. The output is currently only in "cube" format, which can be directly read by VMD. The units in the cube files are bohr for the definition of the grid and atoms per angstrom cube for the density.

Command: *--dmap3D*

List of flags: *+out [dmap3D.cube]*
defines the output file name.

+s [species] or *+i [indices]* or *+mol [molecule name]*
select the species to be used in the calculation using the atoms' names or their indices or the name of the molecules they belong to. The species selection is done by providing a comma separated list of atom names. The index selection can be done by providing a comma separated list of indices or in loop format B:E:(S), where the stride is optional. The *+mol* flag requires *--top* flag.

+nbin [50,50,50]
defines the number of bins to be used for the 3D map.

+cell [metric matrix]
defines the portion of the system where to compute the 3D density map. It can be defined like the system cell using 1, 3, 6 or 9 number for a cubic, orthorhombic or triclinic cell. The triclinic cell can be defined by either 6 numbers $(|a|, |b|, |c|, \alpha, \beta, \gamma)$ or by the full metric matrix.

+origin [0.,0.,0.]
defines the origin of the subsystem.

Examples:
```
gpta.x --i coord.pdb --dmap3D +out dmap3D.cube +s O2 +origin 12,34,56 +cell 10
gpta.x --i coord.pdb --dmap3D +out dmap3D.cube +s O2 +origin 12,34,56 +cell 10,10,2
gpta.x --i coord.pdb --dmap3D +out dmap3D.cube +s O2 +cell 10,10,10,90,90,90
gpta.x --i coord.pdb --dmap3D +out dmap3D.cube +s O2 +cell 10,0,0,0,10,0,0,0,10
```

Screen output: ...
```
-----------------------------------------------------------------------------------------------------
Compute 3D density map
...Number of bins.................................: 50 50 50
...Origin.........................................:     12.0000        34.0000        56.0000
...Region size A..................................:     10.0000         0.0000         0.0000
...Region size B..................................:      0.0000        10.0000         0.0000
...Region size C..................................:      0.0000         0.0000        10.0000
...Output file....................................: dmap3D.cube
...Atoms selection for --dmap3D
......Selection command...........................: +s O2
......Atoms selected in the first group...........:        3137
-----------------------------------------------------------------------------------------------------
...
```

File output:
```
CUBE file generate by GPTA
Density reported in atoms/angstom^3
     8      0.00000      0.00000     81.25822
    50      1.74308      0.00000      0.00000
    50      0.58265      1.69986      0.00000
    50      0.00000      0.00000      1.13384
     1      1.00000      0.00000      0.00000     81.25822
...
  0.38127E-01  0.11037E+00  0.00000E+00  0.51171E-01  0.71238E-01  0.29097E-01
...
```

4.0.4 Solvent Density Map

This action computes the 3D density map for the selected atoms around a solute molecule. The solvent density is computed only inside the largest ellipsoid the fits inside the box around the solvent. The output is currently only in "cube" format, which can be directly read by VMD. The units in the cube files are bohr for the definition of the grid and atoms per angstrom cube for the density.

Command: *--solvation*

List of flags: *+id* defines the ID of the molecule to be use as solute. It has never been tested in cases where more than one solute is present.

+out [solvation.cube]
defines the output file name.

+ref [filename]
read the orientation of the solute molecule from file; the atoms in this file must be listed in the same order as they appear in the trajectory file. There is currently no check that this is the case. If this flag is not used the orientation of the solute molecule in the first frame is taken as the reference orientation.

+s [species] or *+i [indices]* or *+mol [molecule name]*
select the species to be used in the calculation using the atoms' names or their indices or the name of the molecules they belong to. The species selection is done by providing a comma separated list of atom names. The index selection can be done by providing a comma separated list of indices or in loop format B:E:(S), where the stride is optional. The *+mol* flag requires *--top* flag.

+nbin [50,50,50]
defines the number of bins to be used for the 3D map.

+cell [metric matrix]
defines the size of the 3D box around the solute molecule that is considered for the calculation of the solvent density. It can be defined like the system cell using 1, 3, 6 or 9 number for a cubic, orthorhombic or triclinic cell. The triclinic cell can be defined by either 6 numbers $(|a|, |b|, |c|, \alpha, \beta, \gamma)$ or by the full metric matrix. Although a triclinic box can be specified, it currently works only with orthorhombic boxes and the solvent density is compute only inside an ellipsoid that fits in the box By default the solvent density is calculated inside a 5Å sphere around the solute.

+smooth
enables the smoothing of the 3D map. At the moment it is a pretty rudimentary triangular smoothing.

Examples:
```
gpta.x --i coord.pdb --top --solvation +s O2 +id M2  +cell 10,12,14  +nbin 50,50,50
gpta.x --i coord.pdb --top --solvation +id M2 +s O +cell 9 +nbin 20,20,20 +smooth +ref ace.xyz
```

Screen output: ...
```
-----------------------------------------------------------------------------------------------------------
Computing solvation shell
...Output file.................................: solvation.cube
...Number of bins..............................: 50 50 50
...Region size A...............................:     10.0000         0.0000         0.0000
...Region size B...............................:      0.0000        10.0000         0.0000
...Region size C...............................:      0.0000         0.0000        10.0000
...Atoms selection for --solvation
......Selection command........................: +s O2
......Atoms selected in the first group........:        3137
-----------------------------------------------------------------------------------------------------------
...
```

File output:
```
CUBE file generate by GPTA
Density reported in atoms/angstom^3
     8      0.00000      0.00000     81.25822
    50      1.74308      0.00000      0.00000
    50      0.58265      1.69986      0.00000
    50      0.00000      0.00000      1.13384
     1      1.00000      0.00000      0.00000     81.25822
...
  0.38127E-01  0.11037E+00  0.00000E+00  0.51171E-01  0.71238E-01  0.29097E-01
...
```

4.0.5 Extract System Properties

This action extracts a property of the system, which can then be written to a file (*+dump*), averaged (*+avg*) or used to compute its distribution (*+dist*). The action can be used multiple times, but only one property can be computed by each occurrence. The properties that can currently be extracted are:

- The system volume
- The cell lengths and angles

- The cell matrix
- The system density
- The dielectric constant
- The moment of Inertia tensor for the selected atoms
- The position of one or more atoms
- The distance between two atoms

With this command it is also possible to compute the average cell and write out the coordinates of the last frame with the coordinates rescaled to fit inside the average cell (*+system*).

– –

Command: *--extract*

List of flags: *+out [extract.out]*
defines the output filename. If this flag is not present the output will be written on the screen only.

+dump
dumps property in a file.

+avg
computes the average of the property.

+histo [0,1]
computes the histogram of the property in the given range.

+prob [0,1]
computes the probability distribution of the property in the given range.

+nbin [100]
sets the number of bins for the distribution.

+volume
extracts the cell volume.

+cell
extracts the simulation cell as $|a|$ $|b|$ $|c|$ α β γ.

+hmat
extracts the simulation cell as the cell matrix $\vec{a}$ $\vec{b}$ $\vec{c}$.

+density
extracts the density of the system.

+inertia
extracts the eigenvalues of the inertia tensor of the selected atoms.

+diel
extracts the dielectric constant of the system (it requires the atomic charges).

+distance +i [indices]
extracts the distance between two atoms that can be selected with the *+i* flag.

+system [+out averageSystem.pdb]
computes the average simulation cell and writes the coordinates from the last frame rescaled to the average cell. If the *+s* flag is also used the output file will contain the coordinates of the selected atoms and the scaled final coordinates of the others. The flag accepts a comma separated list of species, whose positions will be averaged, as an optional argument; by default all positions are averaged. For the non selected atoms their last positions (scaled) are written out instead. This is useful to compute the average position of the atoms in a slab, which is in contact with water. The average positions are computed in fractional coordinates so that the command could work with NPT trajectories. Note that there is no check for whether the atoms are being re-wrapped across the PBC boundaries during the simulation. If that is the case, atoms that cross the boundary will have an average position somewhere in the middle of the cell. In order to mitigate that problem something like this can be included in the command line before the *--extract* command: *. . . --unwrap*

--fixcom +s Na,Cl +initial --top ...
which unwraps the coordinated and fixes the centre of mass position to the initial one. The *--top* command ensures that any molecules are rebuilt before being written out. The *+out* flag allows for defining the name of the output coordinates file. The output file name for the coordinates is optional.

+pos
extracts the x, y and z coordinates of the selected atoms (maximum 150 atoms can be selected).

+test
empty container for a one off implementation of a property to extract.

+s [species] or *+i [indices]* or *+mol [molecule name]*
select the species to be used in the calculation using the atoms' names or their indices or the name of the molecules they belong to. The species selection is done by providing a comma separated list of atom names. The index selection can be done by providing a comma separated list of indices or in loop format B:E:(S), where the stride is optional. The *+mol* flag requires *--top* flag.

call assignFlagValue(actionCommand,"+avg " ,averageMultiProperty, .false.) ! Average call assignFlagValue(actionCommand,"+dump ",dumpProperty, .false.) ! Dump on same line call assignFlagValue(actionCommand,"+histo " ,lflag(1), .false.) ! Histogram - non-normalised call assignFlagValue(actionCommand,"+prob " ,lflag(2), .false.) ! Probability distribution - normalised to 1 call assignFlagValue(actionCommand,"+kernel",lflag(4), .false.) ! Probability with Gaussian Kernel call assignFlagValue(actionCommand,"+out",outputFile call assignFlagValue(actionCommand,"+sigma ",rtmp,-1.d0) ! Probability with Gaussian Kernel call assignFlagValue(actionCommand,"+nbin ",numberOfBins,100) call assignFlagValue(actionCommand,"+"//trim(str), distributionLimits(1:2),[1.d0,2.d0]) call assignFlagValue(actionCommand,"+out",coordinatesFile ! call assignFlagValue(actionCommand,"+avg",avera ! call assignFlagValue(actionCommand,"+avg",str,'none') call assignFlagValue(actionCommand,"+s",averagePosit

Examples:
```
gpta.x --i coord.pdb traj.dcd --extract +vol +dist +out volume.dist
gpta.x --i coord.pdb traj.dcd --extract +system
gpta.x --i coord.pdb traj.dcd --extract +system +s Na,Cl
gpta.x --i coord.pdb traj.dcd --extract +inertia +dump +out inertia.dat
gpta.x --i coord.pdb traj.dcd --extract +diel +out diel.dat
gpta.x --i coord.pdb traj.dcd --frames 1000:2000 --extract +system +out averageSystem.pdb
```

Screen output: ...
```
-----------------------------------------------------------------------------------------------------------
...Atoms selection for --extracts
......Selection command.........................:
......Atoms selected in the first group.........:          2
-----------------------------------------------------------------------------------------------------------
System property
...Extracting...................................: Cell Parameters
     1        49.318       48.550       84.042       90.000       90.000       90.000
     2        49.318       48.550       84.067       90.000       90.000       90.000
     3        49.318       48.550       84.250       90.000       90.000       90.000
     4        49.318       48.550       84.302       90.000       90.000       90.000
-----------------------------------------------------------------------------------------------------------
...
```

File output:
```
#System property
#...Extracting Bond Length
     1         3.0341183233
...
```

4.0.6 Radial Pair Distribution Function - g(r)

This action computes the Radial Pair Distribution function between species α and β, $g_{\alpha\beta}(r)$, which is defined as

$$g_{\alpha\beta}(r) = \frac{1}{4\pi r^2 N_\alpha \rho_\beta} \sum_{i\in\alpha} \sum_{j\in\beta} \langle \delta(r_{ij} - r) \rangle$$

where r_{ij} is the distance between atoms i and j computed with the appropriate PBC, $\langle\rangle$ indicates the average over all the selected frames, N_α is the number of central atoms, and $\rho_\beta = N_\beta/V$ is the number density of the neighbouring atom.

This action also compute the Kirkwood-Buff integral as:

$$G_{\alpha\beta}(r) = \int_0^r 4\pi r^2 [g_{\alpha\beta}(r) - 1] \mathrm{d}r.$$

– –

Command: *--gofr*

List of flags: *+out [gofr.out]*
defines the output file name

+s [species1] [species2] or *+i [indices1] [indices2]*
select the species to be used in the calculation of the radial pair distribution function using either the atoms' names or their indices. The species selections are done by providing a comma separated list of atom names. The index selections can be done by providing a comma separated list of indices or in loop format B:E:(S), where the stride is optional. The *+mol* flag requires *--top* flag.

+r [6Å]
sets the cutoff distance for the calculation of the radial pair distribution function.

+nbin [100]
defines the number of bins used for the calculation of the radial pair distribution function.

+solute
computes the g(r) for the selected atoms without using the neighbours' list. It is a more efficient algorithm for computing the g(r) of a solvent around a few solute atoms.

Examples:
```
gpta.x --i coord.pdb traj.dcd --gofr +s Ca O1,O2 +out gofr.out +nbin 100
```

Screen output: ...
```
-----------------------------------------------------------------------------------------------------
Settings for the radial pair distribution function
...Cutoff distance.............................:      6.0000
...Number of bins..............................:         100
...Output file.................................: gofr.out
...Atoms selection for --gofr
......Selection command........................:
......Atoms selected in the first group.........:           1
......Atoms selected in the second group........:          99
-----------------------------------------------------------------------------------------------------
...
```

File output:
```
# Distance | g(r) | Coordination Number | KB integral
   0.09000    0.00000    0.00000   -0.00724
   0.15000    0.00000    0.00000   -0.02443
...
```

4.0.7 Molecular Properties

This action computes a variety of molecular properties of the system. The properties that can currently be computed are

- The molecular dipole.
  In order to compute this property the atomic charges must be present in the input files or defined using the *--define* action. Regardless of the method, the atomic charges are set in the first frame and kept unchanged throughout the entire trajectory. This is because not all coordinates formats contain the coordinates, but it could be easily changed by commenting a line in the `readCoordinatesModule.F90` file, around line 335.

- The polar angle of the molecular dipole.
  See the previous property for a discussion about the atomic charges.
- The polar angle for the vector normal to the plane defined by three atoms in the molecule.
- The angle between three atoms in the molecule.
- The torsional angle between four atoms in the molecule.
- The radius of gyration.

$$R_{\mathrm{g}}^2 \stackrel{\mathrm{def}}{=} \frac{1}{M}\sum_{k=1}^{N} m_k \left(\mathbf{r}_k - \mathbf{r}_0\right)^2$$

  where $\mathbf{r}_0$ is the position of the centre of mass, $\mathbf{r}_k$ and m_k are the position and mass of atom k, and M is the total mass of the N selected atoms.

The properties can then dumped to a file (*+dump*) or internally processed to compute

- their average: *+avg*
- their distribution: *+dist*
- their average in slices normal to the z direction: *+distZ*
- thier 2D distribution with the z coordinate of the molecule's centre of mass: *+dist2D*

Command: *--molprop*

Prerequisite: *--top*

List of flags: *+out [molprop.out]*
defines the output file name

+mol [molecule name]
selects the molecule(s) to be used in the calculation.

+normalZ [2,1,3]
computes the cosine of the polar angle of the vector normal to the plane formed by the three atoms specified.

+dipole
computes the molecules' dipole moment, requires the charges to be defined.

+dipoleZ
computes the cosine of the polar angle of the molecules' dipole.

+torsion [4,1,5,8]
computes the torsion angle (in radians) for the four atoms specified.

+angle [2,1,3]
computes the angle (in radians) between the three atoms specified. The first atom in the list is the central atom.

+rgyr
computes the radius of gyration of the molecules.

+dumpSeq
dumps the computed property to the output file sequentially and two empty lines are added to separate the data of the individual frames. This is the default option.

+dump
dumps the computed property to the output file, one line per frame and N_{mol} columns.

+avg
computes the average of the property.

+dist [0,4]
computes the distribution of the property within the specified limits. For the distribution of angles

it is often useful to set the limits in terms of π. This can be done by using the strings: `pi`, `-pi`, `twopi`, `-twopi`, `pih`, `-pih`, `piq` and `-piq`, which are interpreted as π, $-\pi$, 2π, -2π, $\pi/2$, $-\pi/2$, $\pi/4$ and $-\pi/4$, respectively.

+distZ
computes the average property in slices along the z axes.

+dist2D [0,4]
computes the distribution of property within the specified limits in slices along the z axes.

+nbin [100]
defines number of bins to be used in the calculation of the property's distribution.

+nbinZ [100]
defines number of slices to be used in the z direction.

Examples:

```
gpta.x --i c.pdb --top --molprop +mol M1 +angle 1,2,3 +avg
gpta.x --i c.pdb --top --molprop +mol M1 +torsion 1,2,3,4 +dist -pi,pi +nbin 180
gpta.x --i c.pdb --top --molprop +mol M1 +dipole +dist 0,4 +nbin 50 +out dipole.out
gpta.x --i c.pdb --top --molprop +mol M1 +dipoleZ +distZ +nbinZ 100
gpta.x --i c.pdb --top --molprop +mol M1 +normalZ 1,2,3 +dist2D 0,pi +nbinZ 100 +nbin 90
gpta.x --i c.pdb --top --molprop +mol M1 +dipole +dump +out dipole.out
```

Screen output: ...

```
-------------------------------------------------------------------------------------------------------------
Computing molecular properties
...Output file.....................................: molprop.out
...Molecule's type selected........................: M1
...Number of molecules selected....................:       3137
...Computing -> Covalent angle
......Atom indices used............................: 2 1 3
...Computing the property distribution
......Distribution limits..........................:     90.0000       110.0000
......Number of bins...............................:          100
-------------------------------------------------------------------------------------------------------------
...
```

4.0.8 Mean Square Displacement

This action computes the Mean Square Displacement (MSD) of the selected species. The MSD after a certain time interval t is computed as

$$MSD(t) = \frac{1}{N(t)} \sum_{j=1}^{N_t} \sum_{i=1}^{N_{at}} [x_i(t_0 + t_j) - x_i(t_0)]$$

where i runs over the selected atoms (N_{at}), t_0 indicates the reference frame, which is reset every N_t frames and t_j are all the frames between two successive resets of the reference structure. $N(t)$ is the number of configurations that have been found at a time $t = t_j - t_0$ from any given reference frame. The periodic reset of the reference frame helps improve the statistics when computing the MSD for dilute solutes, *i.e.* one ion in water. The flag *+nt* allows for controlling how often the reference frame is reset.

The mean square displacement is then used to compute the self-diffusion coefficient of the selected species. The last 75% of the MDS is fitted with e straight line and the slope is converted to the self-diffusion coefficient, D, using the standard Einstein relations

$$MSD(t) = 6Dt.$$

Note that for the calculation of self-diffusion coefficient to be correct the frames must be uniformly separated. The time interval between the frames is set by default to 1 ps, but this can be changed using the *+dt* flag. It is important to plot the MSD from the output file to ensure that it is indeed linear to ensure that the computed self-diffusion coefficient is converged.

It is important to note that the results obtained from this action with the parallel version of GPTA depend on the number of CPUs used. This is because each CPU computes the MSD up to N_t frames, counted on the frames it receives. Hence, because each CPU processes every N^{th} frame, where N is the

number of CPUs devoted to processing, the maximum time separation between the frames is $N \times N_t \times \delta t$ picoseconds. It is however possible to obtain the same result regardless of the number of CPU used if $N \times N_t$ is kept constant. Please keep in mind that when more than 5 MPI processes are used, one is devoted to only reading the trajectory.

Command: *--msd*

List of flags: *+out [msd.out]*
defines the output file name

+s [species]
species to be used for the calculation of the Mean Square Displacement

+i [indices]
indices of the atoms to be used for the calculation of the Mean Square Displacement

+mol [molecule name]
name of the molecules to be used for the calculation of the Mean Square Displacement

+nt [100]
maximum time separation in number of frames, N_t, from the reference structure to compute the MSD

+dt [1.0]
time separation between the frames in ps, which is used to compute the Self Diffusion Coefficient

Examples:
```
gpta.x --i coord.pdb traj.dcd --msd +s Ca +out msd.out
gpta.x --i coord.pdb traj.dcd --msd +s O1,O2 +nt 1000
gpta.x --i coord.pdb traj.dcd --msd +mol M2 +dt 0.5
gpta.x --i coord.pdb traj.dcd --msd +i 1:300:3
```

Screen output: ...
```
-------------------------------------------------------------------------------------------------------------
Computing mean square displacement
...Maximum number of frames.............................:       1000
...Output file..........................................: msd.out
...Atoms selection for --msd
......Selection command.................................:
......Atoms selected in the first group.................:          1
-------------------------------------------------------------------------------------------------------------
Self diffusion coefficient calculation
  Average D0 [10^-5 cm^2/s]...............................:     0.0734
  Average correlation factor..............................:     0.9444
-------------------------------------------------------------------------------------------------------------
...
```

File output:
```
# Time [ps] | MSD [A^3]
  1.000      0.90433
  2.000      2.47421
...
```

4.0.9 Residence Time

This action computes the solvent residence time around selected solute atoms as

$$F(t) = \sum_i \Theta(n_i - t)$$

where the sum runs over all the exchange events in the coordination shell of the central atom(s), n_i is the number of frames the molecule remained in the coordination shell and Θ is the heavy-side function, which is equal 1 for $t < n_i$ and 0 otherwise. In output also the distribution of exchange events is written.

For efficiency reasons and to simplify the MPI implementation, the coordination shell of each selected atom is computed and stored for all the frames in the trajectory, which are then processed at the end in serial. This unfortunately requires to provide as input the total number of frames that will be analysed.

Command: *--restime*

List of flags: *+out [restime.out]*
defines the output file name

+s [species1] [species2] or *+i [indices1] [indices2]*
select the species to be used in the calculation of the residence time using either the atoms' names or their indices. The species selections are done by providing a comma separated list of atom names. The index selections can be done by providing a comma separated list of indices or in loop format B:E:(S), where the stride is optional. The *+mol* flag requires *--top* flag.

+rcut [cutoff]
sets the cutoff for the definition of the coordination shell.

+ntraj [1000]
sets the number of frames in the trajectory. This is required for efficiency reasons.

+nmax [10]
sets maximum number of atoms in the coordination shell. This is required for efficiency reasons.

+thres [0]
sets the threshold in units of frames for an atom to be considered inside or outside of the coordination shell.

Examples:
```
gpta.x --i coord.pdb --restime +s Ca OW +rcut 3.2 +thres 1 +ntraj 1000
```

Screen output: ...
...

File output:
```
# Frames   | Survival func | Exchange probability
     0          1.00000          0.00000
     1          0.99798          0.00000
     2          0.99706          0.22976
...
```

4.0.10 X-ray Powder Spectrum

This action computes the X-ray powder diffraction spectrum for the input structure, which must contain the cell definition. The output file contains the list of reflections and the powder diffraction spectrum.

The scattering intensity was computed using the traditional formula

$$I_{hkl} \propto F_{hkl}^2 \Theta_{hkl}$$

where

$$F_{hkl}^2 = \sum_i f_i \exp\left[-B\left(\frac{\sin\theta}{\lambda}\right)^2 \right] \exp\left[2\pi i(\vec{q}\cdot\vec{x}_i)\right].$$

f_i are the atomic structure factors, $\vec{q} = (h,k,l)$ is the reciprocal space vector, $\vec{x}_i$ are the atomic fractional coordinates, B is the Debye–Waller factor, $\lambda = 2d_{hkl}\sin\theta$ is the wavelength and d_{hkl} is the distance between the hkl planes.

Θ_{hkl} is the Lorentz polarisation factor

$$\Theta_{hkl} = \frac{1+\cos^2 2\theta}{\sin^2\theta}\frac{1}{\sin\theta}$$

where $(1+\cos^2 2\theta)$ denotes the polarisation factor, $\sin 2\theta$ describes the change in irradiated volume of the crystal as a function of 2θ and the last term is the powder ring distribution factor for a random

distribution of crystals, *i.e.* for a powder sample. The atomic scattering factors were taken from the international tables for crystallography and have the form

$$f = \sum_{i=1}^{4} a_i \exp\left[- b_i \left(\frac{\sin\theta}{\lambda} \right)^2 \right] + c.$$

The Debey-Waller parameter B was assumed to be constant, which is reasonable for harmonic vibrations,

$$B = 8\pi^2\eta$$

where $\eta = 0.05$ for hydrogen and $\eta = 0.06$ for all other elements.

Command: *--xray*

List of flags: *+out [xray.out]*
defines the output file name.

+mini [5]
sets the minimum value of 2θ for the spectrum

+max [50]
sets the maximum value of 2θ for the spectrum

+d2t
sets the spectral resolution ($\delta 2\theta$)

+lambda [1.540560]
sets the wavelength of the x-rays in nm

+kmax [30]
sets the maximum number of k-vectors to use in the calculation

+fwhm
sets the Full Width at Half Maximum for the thermal broadening of the peaks

+cauchy [0.5]
sets the Cauchy n parameter

Examples:

```
gpta.x --i coord.pdb --xray
```

Screen output: ...

```
-----------------------------------------------------------------------------------------------------------
Computing Xray Powder Diffraction Pattern
...Output file.................................: xray.out
...Wavelength [nm].............................:      1.5406
...Twotheta range..............................:      5.0000        50.0000
...Twotheta resolution.........................:      0.0100
...Number of kspace vectors....................:          30
...Smoothing parameters
......FWHM.....................................:      0.1000
......Cauchy n parameter.......................:      0.5000
-----------------------------------------------------------------------------------------------------------
...
```

File output:

```
# index    d_hkl    twoth  intensity      (h , k , l) ...
#     1   17.062    5.175         0.       0   0  -1        0   0   1
#     2    8.531   10.361         0.       0   0   2        0   0  -2
...
# Two Theta | Intensity [a.u.]
  5.000      0.00000
...
```

---

5 Miscellaneous actions

5.0.1 Test Routine

This action is an example routine that can be modified to quickly implement the calculation of a new property or manipulation of the coordinates. In the example, the action shows how the calculation of the radial distribution function can be implemented using the routines to compute distances with periodic boundary conditions and how to use the routine to compute averages and distribution of properties. Note that this implementation of the calculation of the radial pair distribution function is about 10 times slower that the *--gofr* action. That is partially due to the use a bit of openMP parallelisation in the *--gofr* action and to the fact that the property class are generic and designed to work with MPI, which is detrimental to speed.

Command: *--test ...*

Examples:
```
gpta.x --i coord.pdb --test ...
```

5.0.2 PLUMED interface

This action launches the PLUMED interface. It can be used to compute any collective variables (CV) that is available in PLUMED.

Command: *--plumed*

+f [plumed.inp]
defines the name of the file that contains the PLUMED commands. Please refer to the PLUMED manual (https://plumed.org) for the appropriate syntax of this file.

+out [plumed.out]
defines the output file for the PLUMED log, the CV(s) output is defined in the PLUMED input file.

Examples:
```
gpta.x --i coord.pdb --plumed +f plumed.inp +out plumed.out
```

Screen output: ...
...

5.0.3 OpenMM interface

This action calls the openMM library to compute the energy and the molecular dipoles. The interface could be modified and used as a molcular dynamics engine.

Command: *--amoeba*

Prerequisite: *--top*

List of flags: *+out [amoeba.out]*
defines the output file name

Examples:
```
gpta.x --i coord.pdb
```

Screen output: ...
...

File output: ...

---

6 Appendix A
Force Field Format for LAMMPS Input Coordinates

Although there are other several tools available to automatically generate input coordinates files for LAMMPS, such as MOLTEMPLATE (http://www.moltemplate.org), I preferred to have my own tool embedded in the program I use to manipulate and post-process the coordinates files. Similar to MOLTEMPLATE, in my workflow I require three files to run a LAMMPS simulation; one file with all the LAMMPS commands (*i.e.* the timestep, ensemble...), one file with the force field parameters and one file with the coordinates and connectivity. The force field is simply "included" in the LAMMPS input script, while the coordinated file is read with the "read_data" command. I then use GPTA to read the coordinates file, in any supported format, compute the topology and match it with the force field information to produce the LAMMPS coordinates file. However, for GPTA to successfully produce the LAMMPS file some GPTA-specific flags need to be included as comments in the force field parameters file, marked with the "#@" tag, which are ignored by LAMMPS. For example

```
#@ 1 bond types
#@ O2 - H2
bond_coeff   1   harmonic   22.965000      1.0120000
```

means that in the force field there is only one bond type, which is between atoms type O2 and H2. Hence, for each O2-H2 bond that GPTA finds in the topology calculation one line is added in the "Bonds" section of the LAMMPS coordinates file

```
Bonds

      1       1       1       2
      2       1       1       3
      ...
```

A similar philosophy applies to the angle, dihedral and improper types of interactions.

However, GPTA is not too smart and a few rules needs to be followed while constructing the force field file

- only the `atom_type full` is supported
- the `xxx_coeff` line must follow the definition of the atoms involved in the intramolecular interaction (`#@ O2 - H2` or `#@ H2 - O2 - H2`)
- the force field coefficient ID must be consecutive and add up to the number of types specified
- for the angle interactions, the central atom is the second in the list
- for the improper interactions the central atom is the first in the list
- for complicated systems, there might not be a unique charge for each van der Waals type. Hence, GPTA can read the atomic charges from a non-standard PDB file while where the atomic partial charges are written after the 80^{th} column. In this case the charges will be written in the coordinates file. However, because the coordinates file has to be read before the force field file, any charges defined there will be overridden by the definitions in the force field file.

Although GPTA tries to catch some errors in the preparation of the force field file, these more often than not become apparent while running a simulation.

6.1 Force Field File for SPC/FW Water

```
#########################################################################
# Y. J. Wu, H. L. Tepper, and G. A. Voth
# J. Chem. Phys., vol. 124, no. 2, p. 024503, 2006.
#########################################################################
# Global force field settings
#########################################################################
# Bonded interactions styles
bond_style hybrid harmonic
angle_style hybrid harmonic

# Non bonded interactionstyles
variable rcut equal 9.
pair_style hybrid lj/cut/coul/long ${rcut}

# LJ tail correction
pair_modify pair lj/cut/coul/long tail yes
pair_modify mix arithmetic

# Reciprocal space electrostatics
kspace_style pppm 1.0e-5
kspace_modify fftbench no

# 1-4 scaling
special_bonds lj 0.0 0.0 0.0 coul 0.0 0.0 0.0

#########################################################################
# Force field parameters
#########################################################################
#@ 2 atom types
variable O2  equal 1
variable H2  equal 2

# Atoms' masses
mass  ${O2}    16.000
mass  ${H2}     1.010

# Atoms' charges
set type ${O2}  charge    -0.820000
set type ${H2}  charge     0.410000

# Covalent bonds parameters
#@ 1 bond types
#@ O2 - H2
bond_coeff   1   harmonic   22.965000      1.0120000

# Covalent angles parameters
#@ 1 angle types
#@ H2 - O2 - H2
angle_coeff   1  harmonic    1.6456800      113.24000

# Non bonded parameters
pair_coeff  ${O2}  ${O2}  lj/cut/coul/long  0.00674  3.165492
pair_coeff  ${H2}  ${H2}  lj/cut/coul/long  0.00     1.
```

6.2 Force Field File for $CaCO_3$ and SPC/FW Water

```
##################################################################
# P. Raiteri, R. Demichelis, and J. D. Gale
# J. Phys. Chem. C, vol. 119, no. 43, pp. 24447-24458, 2015.
##################################################################
# Global force field settings
##################################################################
# Bonded interactions styles
bond_style hybrid harmonic
angle_style hybrid harmonic class2
#dihedral_style hybrid harmonic
improper_style hybrid distance

# Non bonded interactionstyles
variable rmin equal 6.
variable rmax equal 9.
variable rcut equal 9.
pair_style hybrid/overlay coul/long ${rcut} &
                             lj/cut ${rcut} &
                           buck/mdf ${rmin} ${rmax} &
                             lj/mdf ${rmin} ${rmax}

# LJ tail correction - for homogenous systems only
pair_modify pair lj/cut tail yes

# Reciprocal space electrostatics
kspace_style pppm 1.0e-5
kspace_modify fftbench no

# 1-4 scaling
special_bonds lj 0.0 0.0 0.0 coul 0.0 0.0 0.0

##################################################################
# Force field parameters
##################################################################
#@ 5 atom types
variable O2  equal 1
variable H2  equal 2
variable Ca  equal 3
variable C4  equal 4
variable O4  equal 5

# Atoms' masses
mass  ${O2}    16.000
mass  ${H2}     1.010
mass  ${Ca}    40.080
mass  ${C4}    12.010
mass  ${O4}    16.000

# Atoms' charges
set type ${O2}  charge    -0.820000
set type ${H2}  charge     0.410000
set type ${Ca}  charge     2.000000
set type ${C4}  charge     1.123285
set type ${O4}  charge    -1.041095

# Covalent bonds parameters
#@ 2 bond types
#@ O2 - H2
bond_coeff   1   harmonic   22.965000      1.0120000
```

```
#@ C4 - O4
bond_coeff   2   harmonic   20.424650      1.3042000

# Covalent angles parameters
#@ 2 angle types
#@ H2 - O2 - H2
angle_coeff   1  harmonic    1.6456800      113.24000
#@ O4 - C4 - O4
angle_coeff   2   class2      120.00000      6.6170000      0.0000000      0.0000000
angle_coeff   2   class2 bb    12.81800      1.3042000      1.3042000
angle_coeff   2   class2 ba     1.53319      1.5331900      1.3042000      1.3042000

#@ 1 improper types
#@  C4 - O4 - O4 - O4
improper_coeff   1     13.647000      360.00000

# Non bonded parameters
pair_coeff    *      *    coul/long
pair_coeff  ${O2}  ${O2}     lj/cut      0.00674    3.165492
pair_coeff  ${O2}  ${Ca}     lj/mdf      0.00095    3.35
pair_coeff  ${O2}  ${O4}   buck/mdf  12534.455133   0.202        0.
pair_coeff  ${H2}  ${O4}   buck/mdf    340.         0.217        0.
pair_coeff  ${Ca}  ${O4}   buck/mdf   3161.6335     0.271511     0.
pair_coeff  ${O4}  ${O4}   buck/mdf  63840.199      0.198913    27.89901
```

6.3 Minimal LAMMPS input file

```
# ------ Units --------------------------------------
units metal

# ------ Atom Style ---------------------------------
atom_style full

# ------ System Definition --------------------------
boundary p p p
read_data coord.lmp # <-- generated by GPTA

# ------ Force field --------------------------------
include forcefield.lmp # <-- used by GPTA

# ------ NVE Simulation -----------------------------
fix md all nve
timestep 0.001
thermo 10
run 100
```

7 Acknowledgments and credits

Here I would like to give credit to various sources from which I have lifted significant chunks of code. I have also made every effort to add the credit lines inside each file, if it was not already present in the original files.

- The Mersenne Twister random number generator was taken from
  http://www.math.sci.hiroshima-u.ac.jp/ m-mat/MT/emt.html
  M. Matsumoto and T. Nishimura
  "Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator"
  ACM Trans. on Modeling and Computer Simulation Vol. 8, No. 1, January pp.3-30 (1998)
- The ranking routines were part of ORDERPACK 2.0 -- Unconditional, Unique, and Partial Ranking, Sorting, and Permutation Downloadable Fortran 90 source code authored by Michel Olagnon
  http://www.fortran-2000.com/rank/
- The routines to read and write the GROMACS `xtc` and `trr` files were taken from the GROMACS XTC Library.
- The GSAS FORTRAN libraries to extract the symmetry operators from the space group name were taken from https://subversion.xray.aps.anl.gov/trac/pyGSAS/browser/trunk/fsource/
- The routines to align molecules are the Fortran version of Quaternion Characteristic Polynomial (QCP) algorithm Copyright (c) 2016 Naoto Hori, who "translated" the original code written in C by Douglas L. Theobald and Pu Liu, and distributed at http://theobald.brandeis.edu/qcp. If you use these functionality in a publication please cite the papers on their website, as appropriate.
- The libdcdfort routines for reading the CHARMM DCD files were written by James W. Barnett (https://github.com/wesbarnett/libdcdfort)

List of Actions

List of Commands