# LAMMPS data analysis

## Introduction

Lammps data analysis is a set of scripts designed to easily manage, manipulate and extract information from LAMMPS data files.

This set of tools allow you to examine the contents of a lammps data file in multiple ways. To remove and extract part of the data file. And to export the data file to a number of other formats.
These were developed to ease the use of LAMMPS in connection to other applications (such as Cassandra, Polymatic, Gromacs…) and to simplify the production of polymers with lammps bond/react and AutoMapper (for example to fix connectivity info that can sometimes be broken during the process).
We concentrated our effort on the most common form of LAMMPS data files. However we tried to offer a variety of options, including the possibility of including personalized labels to various elements.

All the scripts can be used as stand alone or be called by other scripts; for this purpose refer to the documentation about the `no_interactive` flag.

## Requirements and data preparation:

### Python env:

These scripts require the use of a Python 3.6 environment with `networkit` module installed, `Pandas` is recommended but not strictly needed.
Remember that the environment must be activated before use (e.g “conda activate <env_name>)

### LAMMPS data format:

The software accept LAMMPS data files of the following format:

- atom style: full
- units: real
- coordinates/angles: orthogonal or restricted triclinic
- Coefficients headers in the form: <Coeff type> Coeffs (e.g. Bond Coeffs, Improper Coeffs,...). These sections can be omitted.

It is possible to insert personalized labels to each line in the sections: PairIJ Coeffs, Masses, Atoms, and all the coefficients sections (Pair, PairIJ, Bond, Angle, Dihedral and Improper Coeffs). Labels need to be given in the form:

```
Masses
1 1.001 # LABEL
2 15.996 # LABEL2
…
```

The label can be of any size (provided it is contained in the same line) and can contain any characters (excluding #)
However, in order to export the data into .pdb format the labels length need to be of max 4 characters (the software will otherwise only use the first 4 characters).

## Functionalities

The functionalities are divided in 4 categories:

- Informations about the data file: handled by `lammps_data_info` returns a variety of useful values:
    - total mass of the system
    - average mass of the particles
    - total volume
    - density
    - total charge of the system
    - number of atoms for each given type
    - number of molecules of each given size
    - size of each molecule
    - a summary.txt file containing all the above

  On top of this, the script can output:
    - a file describing the charges for use with the software Cassandra
    - a data file with atoms IDs renumbered (from 1 onwards with no gaps)
    - a data file with atoms reordered
    - a data file with the connectivity completely recalculated and each atom reassigned to its corresponding molecule
- Other data formats in which the data can be exported: handled by `lammps_data_export` returns the following formats:
    - .xyz
    - .pdb
    - .ciff
    - .mcf
    - .mol
    - types_list (list of atom/bonds/etc. types for Polymatic software)

  On top of this, the script can output:
    - a file describing the charges for use with the software Cassandra
- Functions to manipulate the data file in several ways: handled by `lammps_data_manipulate:`
    - remap labels
    - bring total charge to 0 (by averaging the change over all atoms)
    - shifts charge of all atom of a specific type by a specific amount (important for use with bond/react)
    - remove specified atoms (ID or type), bonds or molecules (by size or ID)
    - extract specified atoms (ID or type) or molecules (by size or ID) to a new data file

    - extract and separate, extracts specific molecules (by size or IDs) to a new data file for each molecule
- Function to extract (from a data file) all fragments of molecules corresponding to a given structure: handled by `lammps_data_extract_fragment` This script can also iterate over all frames of a trajectory keeping track of fragments across the frames.

## General syntax for all scripts

All scripts are called in the same way and the following flags are common to all 4 scripts

### Mandatory arguments

| Command | Possibilities | Explanation |
|---|---|---|
| `lammps_data_<script>` | `info`<br>`export`<br>`manipulate`<br>`extract_fragment` | |
| `-lammps` | `PATH to lammps data file (INPUT)` | `(we suggest to call from the data folder, although not strictly necessary)` |

```
lammps_data_<script> -lammps <PATH_to_input>
```

### Optional arguments

| Command | Options/keywords | Explanation/Default |
|---|---|---|
| `--out` | `PATH to lammps data file (OUTPUT)` | `path to the output file`<br>*`Default: data_out.lmps`* |
| `--name` | `name of the system` | `the name can be used for the output and other possibilities seen later.`<br>*`Default:`*<br>*`name=PATH to input (without extension)`* |
| `--print_coeffs` | | `If used, includes coefficients in the output`<br>*`Default = False`* |

## Lammps_data_info

This script is used to obtain information about a lammps data file. While it can take all the general arguments:

```
    lammps_data_info -lammps <PATH/to/data> --out <PATH/to/output>
    --print_coeffs --name
```

bear in mind they are not always needed (the script will still work even if they are redundant).

The specific arguments for this script are:

| Command | Options/keywords | Explanation |
|---|---|---|
| `--summary` | | a summary of all the following properties (for all atom types and all molecules) |
| `--tot_mass` | | Total mass of the system, in a.m.u |
| `--ave_mass` | | Average mass of the system particles, in a.m.u |
| `--volume` | | Volume of the system (in cubic Angstrom) |
| `--density` | | Density of the system (in a.m.u/Ang$^3$) |
| `--charge` | | Total charge of the system (in elementary electron units) |
| `--n4type` | list of atom types (integers), comma separated *or* all (for all atom types) | Returns the number of atoms for the specified types |
| `--mol_sizes` | | |
| `--mol_sizes_all` | | |

This script also offers the possibility of exporting a file of all charges with the format required by Cassandra software,  and to rewrite the current file, reordering the file and reassigning the molecules. These options might be needed when a software requires the atoms in order or after using bond/react as it can sometimes break the connectivity:

| | | |
|---|---|---|
| `--cassandra_charges` | | |
| `--reorder` | | Reorders the atoms by atoms index. |
| `--renumber` | | Renumbers atoms from 1 onwards without gaps, keeping the atoms in the same order (i.e if the first two atoms have indexes “342, 4” they will be renumbered as 342 → 1 and 4 → 2) |
| `--reassign_mols` | | Reassign molecules labels based on actual connectivity |

## Lammps_data_export

This script is used to export a lammps data file into another format. It takes all the general arguments:

```
lammps_data_info -lammps <PATH/to/data> --out <PATH/to/output>
--print_coeffs --read_coeffs --name
```

plus the specific flag for this script:

```
-format
```

with one of the following arguments:

- data → outputs the same file (for testing)
- xzy → produces a file of type xyz with of the systems
- pdb → produces a file of type xyz of the systems with or without connectivity info (the interactive prompt will ask, no other argument needed) WARNING labels size, see below*
- ciff → produces a file of type ciff of the systems with different possibilities for the choice of labels◸:
- mcf → produces a file of type mcf of the systems with different possibilities for the choice of labels◸ and LJ parameters▲
- mol → prepares a lammps file of type "molecule", for use with Automapper
- types_list → a list of types in the format required by Polimatic
- cassandra_charges → the output isa file of all charges with the format required by Cassandra software

* As previously stated labels for the **pdb** files need to be a maximum of 4 characters long (in the absence of labels in the *Masses* section, the element name will be used).

◸ For the **mcf** and **ciff** files the script will prompt the following question regarding atom labels:

```
Do you want to use element names as .ciff atom labels? (If no,
you can provide labels either in the data file or as an auxiliary
map file) (y/n)
```

If "n" is chosen the script will ask for a file path to retrieve the labels, if no file path is given it will attempt to use labels given in the *Masses* section (in the absence of labels in the *Masses* section, the element name will be used).

The file stating the types label should have the form of the "types file" required by Polymatic, that is:

```
atoms
<lammps_type1(int)>   <new_label>
<lammps_type2(int)>   <new_label>
…
bonds
<lammps_type1(int)>   <new_label>
```

```
        <lammps_type2(int)>   <new_label>
        …
        angles
        <lammps_type1(int)>   <new_label>
        <lammps_type2(int)>   <new_label>
        …
        dihedrals
        <lammps_type1(int)>   <new_label>
        <lammps_type2(int)>   <new_label>
        …
        impropers
        <lammps_type1(int)>   <new_label>
        <lammps_type2(int)>   <new_label>
        …
```
Only the atoms types section is required, the rest may be skipped

▲ For the LJ parameters in the **mcf** file you have the choice of using the coefficient present in the LAMMPS file or providing them in an auxiliary file. (WARNING! Using the coefficients from the data file will give reasonable results *only* if the LAMMPS data includes pair coeffs parameters for LJ related pair_styles.)

The structure of the coefficient file should be as follows:
```
        nonbonded
        <atom1 type number>
        <sigma>
        <epsilon>
        <charge> (optional)
        nonbonded
        <atom2 type number>
        <sigma>
        <epsilon>
        <charge> (optional)
        …
```

Make sure the atom labels used for the atoms matches the ones in the params file and that there are no blank lines.

## Use FF coefficient from separate lammps input file

```
        --read_coeffs <PATH/to/input/file>
```

This option allows you to use coefficients from a separate lammps input file

## No interactive mode for scripting

In order to use these functionalities in a script acting on several files, it is possible to skip the interactive questions on the creation of **pdb, mcf and ciff** files.

This can be achieved with the flag:

```
--no_interactive
```

The effect will be as follows:

- for **pdb** files, the connectivity will always be printed. No argument required
- for **mcf** and **ciff** files, the `--no_interactive` flag expects one of 3 options:
    - `element` → uses element names as labels
    - own → uses labels in the *Masses section*
    - `<PATH/to/types/file>` → uses labels in the specified file
- for **mcf** files,an extra flag is required to determine the provenience of the LJ parameters:

    ```
    --no_inter_mcf_p
    ```

    It expects one of two arguments:
    - own → tries to take coefficients directly from the data file
    - `<PATH/to/types/file>` → uses coefficients from the specified file

## Lammps_data_manipulate

This script allows the user to apply various manipulations to the data files.

This script is used to manipulate a lammps data file. It always takes the generic flags:

```
    lammps_data_info -lammps <PATH/to/data> --out <PATH/to/output>
    --print_coeffs --name
```

plus the specific flags for this script, which are:

| Command | Options/keywords | Explanation |
|---|---|---|
| `--reorder` | | Reorders the atoms by atoms index. |
| `--renumber` | | Renumbers atoms from 1 onwards without gaps, keeping the atoms in the same order (i.e if the first two atoms have indexes “342, 4” they will be renumbered as 342 → 1 and 4 → 2) |
| `--reassign_mols` | | Reassign molecules labels based on actual connectivity |
| `--remap_types` | `<path/to/types /file>*` | Remaps types based on a types_map file, does not change the total number of types |
| `--insert_coeffs` | `<path/to/input /file>` | Copies coefficients data from a lammps input file into a lammps .data file |
| `--zero_tot_q` | | Shift all charges to achieve zero total charge (total charges is divided by the number of particles, and all particles’ charges are shifted by that amount) **BE CAREFUL with adjusting large charges to zero** |
| `--qshift` | `atom_type_A, charge_shift_A, atom_type_B, charge_shift_B,...` <br><br> `comma separated` | Shifts the charge of all particles of a certain type by a determined amount. <br><br> Useful to foster reactions (e.g. shifting of opposite amount on reactive atoms, before using bond/react, will allow the atoms to get close enough to react) |

* see **lammps_data_export** section on how to build a types file

In addition the flags:

```
    --remove
    --extract
    --extr_separate
```

can be used to obtain subsets of the current file. In particular:

| Command | Explanation |
|---|---|
| `--remove` | Remove **atoms, molecules** or **bonds** from lammps data file, according to the appropriate flags (see below). |
| `--extract` | Extract specific **atoms** or **molecules** from lammps data file, according to the appropriate flags (see below) and creates a data file containing only the extracted particles |
| `--extr_separate` | Extract a set of **molecules** from a lammps data file, according to the appropriate flags (see below) and create a separate data file for each single molecule. |

This set of flags should be used in combination with the ones above:

| Command | Options/keywords |
|---|---|
| **ATOMS** | |
| `--number` | list of atom IDs to be removed, comma separated.<br><br>Ranges expressed as n_start-n_end.<br><br>1,3-5,10-12 → will remove atoms 1,3,4,5,10,11 and 12 |
| `--type` | list of atom types to be removed from system, numerical and comma separated |
| **BONDS** | |
| `--bond` | IDS of pairs of atoms between which to remove bonds. Atoms in pair dash separated, pairs comma separated<br><br>atom1- atom2, atom6-atom8 |
| **MOLECULES** | |
| `--mol` | list of molecules numbers to be removed from system, numerical and comma separated<br><br>Ranges expressed as n_start-n_end.<br><br>1,3-5,10-12 → will remove molecules1,3,4,5,10,11 and 12 |
| `--molsize` | Integer. Size of molecules to be removed (number of atoms). Only one size at the time. |

## Lammps_data_extract_fragment

This script allows the user to select from a datafile all occurrences of a specific topology, even when it is a part of a larger molecule.

It will then create a single data file for each occurrence.

It will also work with a whole trajectory, extracting the fragments from each frame and naming the data files in a way that allows the user to follow the same fragment across the length of the trajectory.

***Warning: The current implementation is likely to overcount symmetric fragments.***

In can be called with:

```
lammps_data_extract_fragment -lammps <PATH/to/data>
--print_coeffs --name
--out <prefix for output files (string)>
-fragment <PATH/to/data/topology>
--start <atom ID> --traj  <PATH/to/xyz/trajectory>
--dir <PATH/out/dir>
```

The topology of the **fragment to extract** is read from a **lammps data file** containing only that fragment.

The **starting atom** for the reading of the topology can be chosen, it will otherwise be set at 1 (make sure that the first atom is labeled 1 in your fragment data file!)

All files produced will be written to a directory determined by the **dir** flag (default: “fragments”). The directory does not need to exist already, the software will create one if needed.

The **trajectory** file is a standard .xyz trajectory as produced by lammps.

The **output** is a .xyz file for the trajectories and a .data for the data files.
The **naming** convention on the **output** files is as follows:

- fragments from **trajectory**:
  “name taken from `--out” + “counter for each fragment” + “frame”`
  (so the 1st fragment of the 2nd frame will be named `<out_prefix>_frag1_2.xyz` ).
- fragments from **data file**:
  “name taken from `--out” + “counter for each fragment”`
  (so the 1st fragment will be named `<out_prefix>_1.data` ).
- (--out default:”fragment”)