# AutoMapper Manual

Matthew Bone, Bristol Composites Institute, University of Bristol, UK

August 2021

## 1 Introduction

AutoMapper is a package of tools designed to accelerate the use of the LAMMPS command *fix bond/react* by automating the time-consuming preprocessing stages. The user must provide two LAMMPS input files, one representing the molecule structure before bonding (the 'pre-bond' file) and a one representing the molecule structure after bonding (the 'post-bond' file). Alongside this the user must supply the LAMMPS Atom IDs for the bonding atoms and a list of elements by type, along with a list of Atom IDs for any atoms to be deleted from the system. With one command line function the user is able to automatically generate the pre- and post-bond molecule files and a map file for a reaction. These files will be reduced to the smallest partial structure possible without any involvement from the user. Additionally, there is a cleaning tool which unifies the types between two or more LAMMPS input files and a coefficient file. These tools have been built to work with the "full" LAMMPS atom style, results with other atom styles may vary.

## 2 Tools

AutoMapper is built around three tools: *map*, *clean* and *molecule*. All of these are accessible with the AutoMapper.py wrapper function. It is recommended that the user add AutoMapper.py to their PATH so that it can be called from your modelling directories. AutoMapper requires Python 3 and the third party Python module natsort to run. The following sections will explain how to use each tool. Further explanations of the arguments can be found with *AutoMapper.py -h*.

### 2.1 *map* Tool

The *map* tool is the primary tool within AutoMapper and outputs two molecule files and a map file from minimal user input. The user should provide one pre-bond and one post-bond LAMMPS input files with consistent types (e.g. Atom, Bond, Angle, etc.) between each file. The user then must provide the Atom IDs of the bonding atoms, a list elements by type, and Atom IDs of delete atoms, if present. The tool will output two molecule files, reduced to partial structures if possible, with file names specified by the user, as well as a map file called "automap.data". If partial structures can be produce the molecule and map files will be reduced and renumbered automatically. If no partial structure is possible the original input numbering system will be used. Partial structures are recommended for *fix bond react* as it speeds up the LAMMPS computation and makes modelling polymerisation possible. Below is an example command line call with a breakdown of details for each argument.

```
AutoMapper.py . map cleanedpre-reaction.data cleanedpost-reaction.data --save_name pre-
    molecule.data post-molecule.data --ba 2 5 3 7 --da 1 8 1 8 --ebt H H C C N O O
```

- *AutoMapper.py*: This is the call to the AutoMapper function wrapper.
- . *(Period)*: This is the directory where the files are to be found and saved. Using . or *$PWD* in Linux will get the current directory, but any directory can be given.
- *map*: This is the name of the tool to be used.
- *cleanedpre-molecule.data cleandedpost-molecule.data*: These are the file names of pre- and post-bond molecules in the LAMMPS input format. The files must be in the order "pre-bond post-bond".
- *--save_name pre-molecule.data post-molecule.data*: These are the save names for the output molecule files.
- *--ba 2 5 3 7*: The Atom IDs of the bonding atoms in the pre-bond (2 5) and post-bond (3 7). The order of these numbers is important: the first and third must represent the same atom, the same with the second and forth (i.e. 2/3 and 5/7).

- *--da 1 8 1 8*: The Atom IDs of atoms to be deleted in the pre-bond (first two) and post-bond (last two). This argument can be ignored if no atoms are to be deleted. The order of these numbers is important: An even number of atoms N must be provided so that the range 1 to N/2 atoms represent the pre-bonding atoms, whilst the range N/2+1 to N represent the corresponding post-bonding atoms.
- *--ebt H H C C N O O*: The elements by type as found in the input files. Any elements may be given. Elements must be separated by a space.

## 2.2 *clean* Tool

The *clean* tool automates the type unification of two or more LAMMPS input files and removes unused types if present. It takes a list of coefficients and reduces it down to those needed for the provided molecules. The output is then cleaned LAMMPS input files and coefficient file with the same Atom, Bond, Angle, Dihedral and Improper types shared across all files. This is a requirement of the *map* tool and *fix bond/react*. The tool is similar to the Moltemplate tool *cleanupmoltemplate.sh* except with the ability to take any number of input files. Below is an example command line call with a breakdown of details for each argument.

```
AutoMapper.py . clean pre-reaction.data post-reaction.data --coeff_file system.in.settings
```

- *AutoMapper.py*: This is the call to the AutoMapper function wrapper.
- *. (Period)*: This is the directory where the files are to be found and saved. Using . or *$PWD* in Linux will get the current directory, but any directory can be given.
- *clean*: This is the name of the tool to be used.
- *pre-reaction.data post-reaction.data*: These are LAMMPS input files to be cleaned and unified. Two or more file names can be given here
- *--coeff_file system.in.settings*: This is a file containing a list of coefficients that will be reduced to just the required coefficients for the reaction.

**Assumption:** Pair coefficients in the coefficient file must be defined with numbers and not use wildcards (*).

## 2.3 *molecule* Tool

The *molecule* tool is the LAMMPS input file to molecule file converter used within the *map* tool. This will output a molecule file directly converted from the LAMMPS input file; this tool will not determine partial molecules. Whilst not necessary for the *map* tool workflow, this tool is useful should the user need molecule files for some other purpose. Below is an example command line call with a breakdown of details for each argument.

```
AutoMapper.py . molecule cleanedpre-reaction.data --save_name pre-molecule.data
```

- *AutoMapper.py*: This is the call to the AutoMapper function wrapper.
- *. (Period)*: This is the directory where the files are to be found and saved. Using . or *$PWD* in Linux will get the current directory, but any directory can be given.
- *molecule*: This is the name of the tool to be used.
- *cleanedpre-reaction.data*: The name of the LAMMPS input file to be converted to a molecule file.
- *--save_name pre-molecule.data*: This is the save name for the output molecule file.

## 3 Typical Workflow

Users need to provide initial LAMMPS input files containing information on molecules before and after a reaction has taken place: these are referred to as the pre-bond and post-bond molecules. You may wish to use tools like Avogadro2 and Moltemplate which can be used to draw molecules and create LAMMPS input files from a template. The pre-bond and post-bond files should only include the molecules required for a reaction (or one molecule, in the case of an intramolecular reaction). Once these input files are created, the *clean* tool can be used to unify the types between the input files. A coefficient file is also required that contains all the forcefield information: this is the 'system.in.settings' file if using Moltemplate.

Once generated and cleaned, the LAMMPS input files can be used by the *map* tool along with the Atom IDs of the bonding atoms, the elements by type list, the save names of the molecule files, and the Atom IDs of any delete atoms. These values are supplied by user when calling the AutoMapper function. The output molecule and map files can then be used within a LAMMPS simulation that calls *fix bond/react*. If a partial structure is possible then the molecule and map files will be reduced and renumbered automatically. If no partial structure is found the numbering system will be the same as the input files. For a worked example with starting files, see the Example_Workflow folder of the repository.

## 4 Problems

If you have any problems whilst using AutoMapper please raise an issue in the Issues section of the GitHub repository. Please check the Issues section to see if your query has already been answered.