**MolCluster V2.5, a molecule cluster cutting tool: User manual**

1. **Overview**
   1. Background

The original purpose of developing MolCluster was to facilitate the transition from a periodic classical molecular dynamics (MD) simulation using DL\_POLY to (embedded) finite cluster QM/MM calculations using ChemShell. ChemShell achieves this by first sampling the trajectory to obtain a set of snapshots, and then generates a finite cluster for each snapshot. The sampling and specification of the cluster parameters is made an interactive process on the command line giving the user a significant level of flexibility. In addition, MolCluster has the option to generate the necessary input files to run geometry optimisation calculations in ChemShell for each cluster. Finally, additional features have also been implemented in ChemShell to analyse the trajectory from a DL\_POLY MD simulation to obtain geometric properties such as the bond correlation function between two atoms, or the evolution of the distance between specified atoms or molecules over the simulation.

* 1. Concept of the (embedded) finite cluster model

Sometimes it is necessary to use a finite cluster system instead of a periodic system for QM/MM calculations. This is because some QM codes are not capable of dealing with a periodic point charge distribution, encountered in electrostatic embedding. In addition, if the system includes features such as a single defect, the use of finite clusters will avoid artifacts such as defect-defect interactions that would arise if periodic boundary conditions are used. To reproduce the bulk electrostatic interactions missing as a result of cutting a finite cluster from a periodic system, embedding point charges can be used. The magnitudes of these charges are usually fitted to minimise the error in the electrostatic potential and field. ChemShell itself can generate embedded clusters from a periodic system. However, there is little flexibility in choosing the attributes of the cluster - only spherical embedded clusters are allowed. MolCluster provides more flexibility into the construction of finite clusters, which may be needed particularly for molecular systems with large molecules such as ionic liquids.

* 1. Running MolCluster

To run MolCluster, go to the directory containing the MolCluster scripts and simply enter ./MolCluster.py in the command line. MolCluster requires the DL\_POLY FIELD (input force field info file) and HISTORY (output trajectory file) files to run and they both need to be in the same directory. At the start of a MolCluster process, the path of the directory containing these two files is requested. Following successful extraction of the system contents and trajectory, the user is given a choice of 5 main tasks. These are described in detail in sections 2 to 6.

1. **Task 1: Cutting bare clusters (for ChemShell QM/MM simulations)**
   1. Capability and options

The overall aim of this function is to sample the MD trajectory and extract bare clusters out of the resulting set of snapshots. The trajectory can either be sampled at regular intervals or the user can specify a custom list of snapshots. The definition and parameters of the finite clusters cut from the snapshots is highly customisable. The customisable options are as follows:

🡪 Cluster origin: the origin of the clusters can be an atom, a molecule or specified coordinates.

🡪 Cluster size: the size of the clusters can be specified by a radius or by a fixed number of each type of molecules. If the former option is chosen, the user can choose either for all atoms to be within the boundary defined by the radius (hard boundary) or allow atoms to be outside the boundary as long the centre of mass of its parent molecule is within the boundary (soft boundary).

🡪 Neutrality of the clusters: the user can request for the clusters to be neutral. If required, MolCluster will include extra or remove existing molecules to force neutrality.

MolCluster will generate an XYZ file of each cluster. MolCluster can also generate ChemShell fragment, force field and QM/MM optimisation input files if requested by the user. If requested, the following QM/MM options are available:

🡪 QM region: the QM region can be defined either as a spherical region about the cluster origin, as a specified number of molecules (overall, or for each molecule type) closest to the cluster origin, as a specified upper limit of atoms closest to the cluster origin, or as multiple spherical volumes centred about different origins. Alternatively, the user can manually select the molecules to be included in the QM region for each cluster.

🡪 Active region: the active region can be defined either as a spherical region about the cluster origin, as a specified number of molecules (overall, or for each molecule type) closest to the origin, or as a specified upper limit of atoms closest to the cluster origin.

🡪 Co-ordinate system: the user can choose out of three co-ordinate system types available in ChemShell geometry optimsation: Cartesian, delocalised internal co-ordinates or hybrid delocalised internal coordinates.

* 1. Output files

The output directory path is specified by the user. The files associated with each cluster generated are placed in a subdirectory of the output directory. The subdirectories are named [BASENAME]\_[n], where [BASENAME] is specified by the user and [n] is the number of the cluster. The following output files are written to the parent output directory:

🡪 bulk\_system\_info.txt: a table containing the number of each type of molecule in the bulk periodic system along with information about the constituent atoms and charge.

🡪 molcluster.log: a log file listing all the options chosen by the users and details of some operations performed by MolCluster.

In each cluster subdirectory, the following files are generated:

🡪 system\_info.txt: same as bulk\_system\_info.txt, but for the cluster system.

🡪 atom\_by\_atom.info: Contains information of each atom in the cluster. AtomNo. refers to the atom number, MolNo. refers to the number of the parent molecule, Dist2Or refers to the distance from the atom to the cluster origin, in Angstroms.

🡪 mol\_by\_mol.info: Contains information of each molecule in the cluster. MolNo. refers to the molcule number, AtomNos refers to the range of atoms associated with the corresponding molcules, COM2Or refers to the distance from the molecular centre of mass to the origin, MinD2Or refers to the minimum distance from the molecule to the cluster origin, MaxD2Or refers to the maximum distance from the molecule to the origin. All distances are in Angstroms.

🡪 [BASENAME]\_[n].xyz: XYZ file of cluster.

🡪 [BASENAME]\_[n]\_tiered.xyz: XYZ file of cluster in which a number is added to label of each atom corresponding to the region the atom is in. ‘1’ corresponds to the QM region, ‘2’ to the MM active region and ‘3’ to the MM frozen region.

The following ChemShell files are generated, if requested by the user:

🡪 [BASENAME]\_[n].chm: Fragment input file.

🡪 opt.chm: QM/MM optimisation input file.

🡪 ff.dat: Force field definition file.

The following Python object files are also generated, if requested by the user:

🡪 bulk.system: Object representing the original periodic system.

🡪 cluster.system: Object representing the cluster system.

These object files can be read by python and are used mainly for debugging purposes.

It should be noted that the clusters generated using in this task are bare clusters without embedding point charges. To generate embedded clusters, the user should first run *Task 2a*, to generate the ChemShell input files to parameterise the embedding charges, followed by *Task 2b* to generate bare clusters and merge them with the parameterised embedding charges.

1. **Task 2a: Generation of ChemShell input for cutting embedded clusters**

This is the first of two tasks that should be performed to generate embedded QM/MM clusters. In this task, the user first selects a set of snapshot to extract from the trajectory. A ChemShell fragment input file is generated for each snapshot. In addition the user specifies the following information required by ChemShell to parameterise the embedding point charges:

🡪 Cluster origin: same as in *Task 1*.

🡪 Cluster radius: the cluster is a spherical region centered about the origin with a specified radius.

🡪 Active region radius: the active region is also a spherical region centered about the origin with a specified radius. Here, the active region refers to the region in which the electrostatic potential and field are sampled to parameterise the magnitude of the embedding point charges. The radius of the active region should be similar if not the same as the active region intended for the QM/MM optimisations.

🡪 Distance from embedding point charges to the cluster: a suitable distance from the embedding point charges to the cluster is roughly one typical bond length in the system.

🡪 Density of embedding point charges: this is specified by choosing an integer value for the ChemShell variable bq\_density. The total number of embedding point charges is (4\*bq\_density\*bq\_density)+2.

As in *Task 1*, the user specifies an output directory, in which sub-directories are created for each cluster. A ChemShell input file, gen\_embedded\_cluster.chm, to generate an embedded cluster is created in each sub-directory. The ChemShell embedded clusters must be generated to run *Task 2b*. The cluster parameters are stored in the Python object file embedded\_cluster\_specs.obj in the output directory and will be read when running *Task 2b*.

1. **Task 2b: Cutting bare clusters and merging with embedding charges**

This is the second of two tasks that should be performed to generate embedded QM/MM clusters. This task should only be performed after embedded clusters (embedded\_cluster.pun) have been generated by ChemShell using the input files created in *Task 2a*. Each ChemShell embedded cluster fragment file should be in the same sub-directory as its input file (created in *Task 2a*). The cluster parameters (active region, cluster radius) specified when performing *Task 2a* are first read from the file embedded\_cluster\_specs.obj. The user will have the option to alter the parameters for generation of the bare clusters if desired. In addition, the user will be asked to specify the QM region. MolCluster will construct each bare cluster, extract the co-ordinates and magnitudes of the corresponding embedding point charges and then merge them together, resulting in a set of embedding clusters. For each embedded cluster, ChemShell fragment and geometry optimisation input files are also generated.

1. **Task 3: Geometric analysis of a DL\_POLY trajectory**

This task is used to conduct spatial analysis of atoms or molecules. The user can either analyse the whole trajectory or a specified segment. The following sub-tasks are available: TBC

1. **Task 4: Calculation of the bond correlation function.**

This task is used to calculate the bond correlation function between two specified atoms. TBC

1. **Running ChemShell**

If running on the Imperial College HPC platform cx1, the user must load the ChemShell module by entering the command: module load chemshell. In general, a ChemShell input file [INPUT].chm is run through the command chemsh.x [INPUT].chm. Instructions to run ChemShell tasks relevant to MolCluster are given in sections 7.1-7.4.

* 1. Generating a fragment

This task requires an input file containing coordinates and optionally, charges, of all atoms in the system and if the system is periodic, specifications of the unit cell. ChemShell will determine the connectivity (bonds) within the system from the input geometry. The original information and connectivity is re-formatted in the output fragment file. The resulting fragment file is needed by ChemShell for all other tasks.

* 1. Generating an embedded cluster

This task requires an input file containing specifications of the embedded cluster to be generated along with the fragment file of the bulk system that the cluster is to be cut out of. ChemShell will parameterise the magnitudes of the embedding charges and generate a fragment file of the final embedded cluster.

* 1. Single point calculation

* 1. Geometry optimisation

For computationally intensive tasks such as generating an embedded cluster (7.2), single point calculations (7.3) and geometry optimisations (7.4), the ChemShell run command should be embedded in a PBS job script and submitted to the HPC queue system.

1. **Description of each script file**

🡪BondCorrelationFunctions.py: Contains functions to calculate the bond correlation function.

🡪CommonFunctions.py: Contains functions that are commonly used across all MolCluster tasks.

🡪ChemShellInputFunctions.py: Contains functions used when generating several ChemShell input files.

🡪GeneratedEmbeddedClusterInput.py: Contains code to generate the ChemShell input file to generate a embedded cluster along with code to merge bare cluster with ChemShell parameterised embedding charges.

🡪GlobalVariables.py: Contains variables that are used across all MolCluster tasks.

🡪MolCluster.py: The main script file to run MolCluster.

🡪MolClass.py: Contains code associated with each Python class used in MolCluster.

🡪TrajectoryGeometryAnalysis.py: Contains functions to analyse the geometry in a DL\_POLY MD trajectory.