**Description of input files:**

1. data.in: contains initial geometry of the system in LAMMPS ReaxFF format.
2. input.xyz: contains initial geometry of the system in xyz format.
3. in.reaxc: main input file that is run by LAMMPS at the first. The file contains the entire GCMC algorithm and calls other files when they are needed.
4. control\_gc: contains a list of all input parameters for the code to run.
5. control\_emin: contains all the commands for running ReaxFF energy minimization.
6. control\_md: contains all the commands for running ReaxFF MD.
7. getcoord.py: Python script that contains the algorithm for computing random coordinate during addition and displacement steps of GCMC. In the simplest case, this code selects coordinates which are within rmin and rmax distance (defined in control\_gc file) of any existing atom. Users can modify/write their own code in this file to compute random coordinates.
8. ffield: contains ReaxFF force field.
9. get\_last: bash script to obtain the entry at last timestep from dump and bond output files.
10. lmp\_control: contains ReaxFF inputs for LAMMPS (optional).

**Description of output files:**

1. Thermo: contains the thermodynamic data of the system at all the accepted GCMC steps (number of GCMC atoms, potential energy, kinetic energy, total energy, volume, temperature, pressure, simulation cell dimensions, density)
2. Refile: binary file (LAMMPS restart file) that contains the state of the system (geometry, velocities, etc.) at the latest GCMC step.
3. dump.reax: contains the geometry of the system at all the accepted GCMC steps.
4. bonds.out: contains the bond information of the system at all the accepted GCMC steps.
5. dump.strs: contain stress tensor of all atoms at all the accepted GCMC steps.
6. dump.last: contains the latest geometry of the system (can be the actual or trial state)
7. bonds.last: contains the latest bond information of the system (can be the actual or trial state)
8. dump.strs.last: contains the latest stress tensor information of the system (can be the actual or trial state)
9. ener\_comp: contains the decomposition of ReaxFF potential energy at all the accepted GCMC steps.
10. status.log: contains a log of all the steps the code went trhough (used for debugging)

**Input files to be modified:** data.in, control\_emin, control\_md, control\_gc

**Steps to use hybrid MD/GCMC:**

1. Obtain the LAMMPS datafile in ReaxFF format and name it as data.in, this file will contain the initial geometry of the system. Make sure that the GCMC atom (Li atom) is mentioned in the data.in file, even if the GCMC atom is not present in the initial geometry. This can be done by simply incrementing the *atom types* by 1 and adding the atomic mass of GCMC atom in the list of atom masses.
2. Update *pair\_coeff* in control\_md and control\_emin files. Specify all the atoms types at the end of this command in the order they appear in the data.in file. For details, refer to the *pair\_coeff* command for ReaxFF in LAMMPS documentation.
3. Update input parameters in control\_gc. The description of each parameter is given in the control\_gc file.
4. Run LAMMPS: use “srun lmp\_mpi -var seed `echo $RANDOM` -i in.reaxc”

**Steps to continue or restart a simulation:**

1. The code writes a binary file named “refile” after every accepted GCMC step. This file stores the latest state of the system and should be used to restart a simulation.
2. The “refile” should be renamed as “infile” (input restart file).
3. Change input parameter named *readfile* in the control\_gc file from 0 to 1.
4. Run LAMMPS in the same way: “srun lmp\_mpi -var seed `echo $RANDOM` -i in.reaxc”