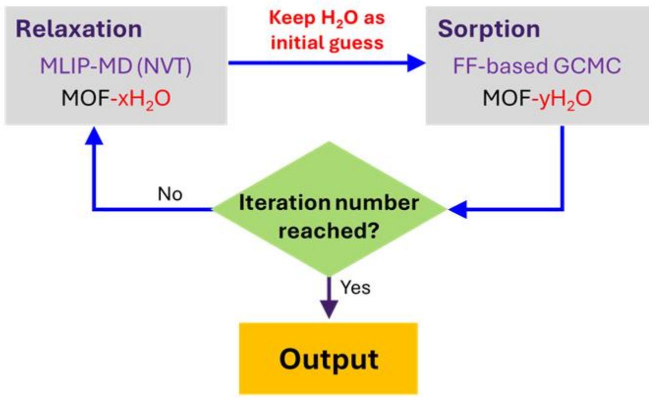
MOFAFF Code Overview and Tutorial

Xijun Wang

1. Introduction of MOFAFF

The MOF Adsorption with Framework Flexibility (MOFAFF) package is a Python tool designed to run **Sorption-Relaxation (SR) simulations**: iteratively run machine-learning potential (MLP)-based molecular dynamics (MD) simulations using LAMMPS and Monte Carlo (MC) simulations using gRASPA. It enables the calculation of gas uptake in porous materials while explicitly accounting for framework flexibility.



Download the code from Xijun's github repository: https://github.com/xwsci/mofaff/tree/main

2. Installing and Running MOFAFF.

2.1. Clone the MOFAFF Repository

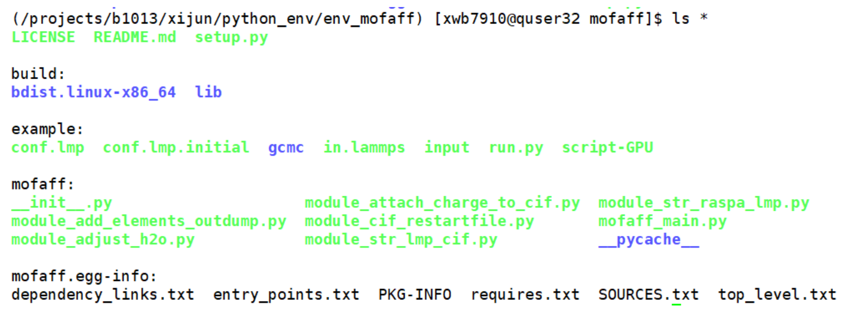
Download: Go to your directory on Quest and download the code from github:

```
cd /your_directory/
```

```
git clone https://github.com/xwsci/mofaff.git
```

```
cd /your_directory/mofaff/
```

```
ls
```


```
(/projects/b1013/xijun/python_env/env_mofaff) [xwb7910@quser32 mofaff]$ ls *
LICENSE  README.md  setup.py

build:
bdist.linux-x86_64  lib

example:
conf.lmp  conf.lmp.initial  gcmc  in.lammps  input  run.py  script-GPU

mofaff:
__init__.py                  module_attach_charge_to_cif.py  module_str_raspa_lmp.py
module_add_elements_outdump.py  module_cif_restartfile.py     mofaff_main.py
module_adjust_h2o.py         module_str_lmp_cif.py          __pycache__

mofaff.egg-info:
dependency_links.txt  entry_points.txt  PKG-INFO  requires.txt  SOURCES.txt  top_level.txt
```

2.2. Set Up the Conda Environment

conda create -n mofaff_env python=3.8

conda activate mofaff_env

conda install numpy=1.24.4

conda install -c conda-forge ase=3.22.1

2.3. Install MOFAFF

cd /your_directory/mofaff/

pip install . Or to update an existing installation: pip install . –upgrade

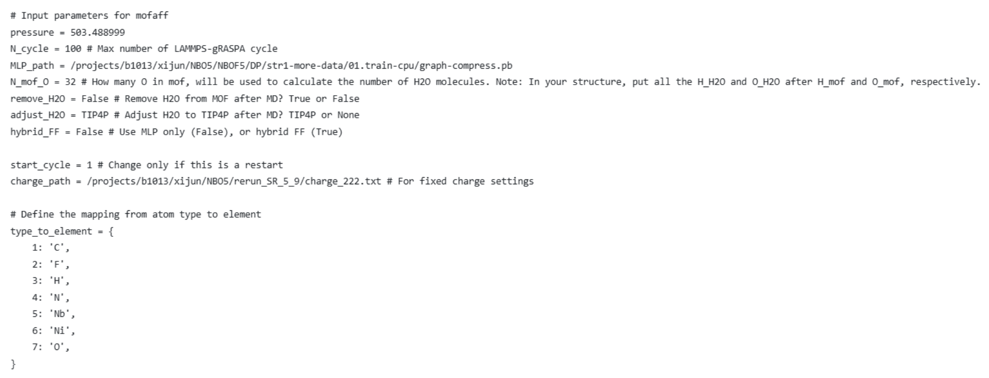
2.4. Preparing Input Files

Example input files can be found on Quest:

/projects/b1013/2ijun/NBO5/rerun_SR_5_9/example_SR_input/

In this directory, you can find the following files:

- charge_222.txt: charge files containing the DDEC06 charge for each atom.
- conf.lmp: LAMMPS input structure file. *Make sure in your conf.lmp, put all O_h2o after O_mof, and all H_h2o after H_mof.*
- conf.lmp.initial: Backup copy of conf.lmp (since conf.lmp updates over the simulation).
- in.lammps: LAMMPS input file, containing all MD settings.
- gcmc/: This is a folder containing all gRASPA files, including force_field.def, force_field_mixing_rules.def, pseudo_atoms.def, simulation.input, TIP4P.def
- input: Main SR simulation settings, including N_cycle: the maximum SR iteration you want to reach; MLP_path: the path to the MLP file (graph-compress.pb); N_mof_O=32: the first 32 O atoms belong to MOF.



```
# Input parameters for mofaff
pressure = 503.488999
N_cycle = 100 # Max number of LAMMPS-gRASPA cycle
MLP_path = /projects/b1013/xijun/NBO5/NBOF5/DP/str1-more-data/01.train-cpu/graph-compress.pb
N_mof_O = 32 # How many O in mof, will be used to calculate the number of H2O molecules. Note: In your structure, put all the H_H2O and O_H2O after H_mof and O_mof, respectively.
remove_H2O = False # Remove H2O from MOF after MD? True or False
adjust_H2O = TIP4P # Adjust H2O to TIP4P after MD? TIP4P or None
hybrid_FF = False # Use MLP only (False), or hybrid FF (True)

start_cycle = 1 # Change only if this is a restart
charge_path = /projects/b1013/xijun/NBO5/rerun_SR_5_9/charge_222.txt # For fixed charge settings

# Define the mapping from atom type to element
type_to_element = {
    1: 'C',
    2: 'F',
    3: 'H',
    4: 'N',
    5: 'Nb',
    6: 'Ni',
    7: 'O',
}
```

- run.py: Script to launch the SR simulation: python run.py.

Note: In in.lammps, the pressure should be set as the pressure value (if using NPT), whereas in gcmc/simulation.inp, the pressure should be set as the fugacity value.

2.5. Run SR simulation:

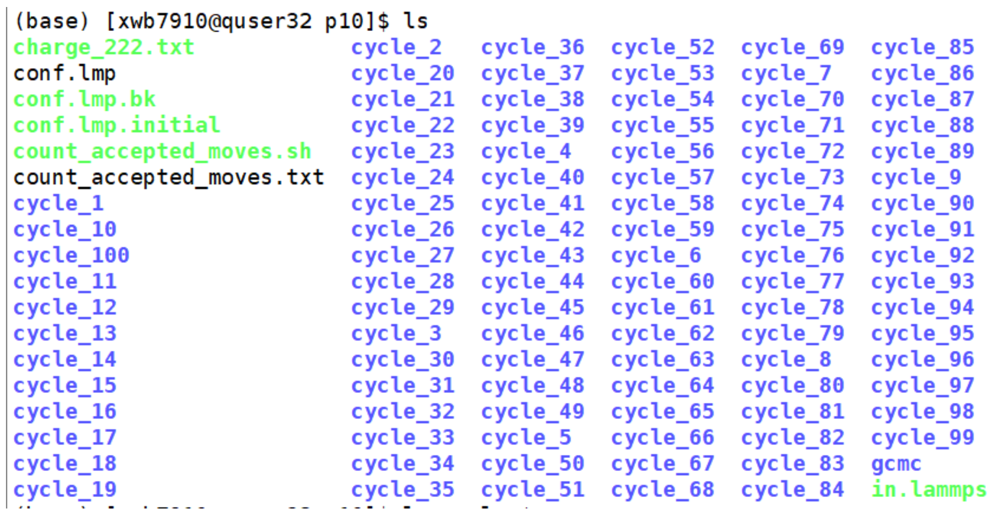
Example Slurm script for submitting SR jobs:

```
#!/bin/bash
#SBATCH -A p31504
#SBATCH -p gengpu
#SBATCH –gres=gpu:a100:1
#SBATCH -N 1
#SBATCH -n 1
#SBATCH -t 48:00:00
#SBATCH –mem=32G
#SBATCH –error=e.%J
#SBATCH –output=o.%J
python3 run.py
```

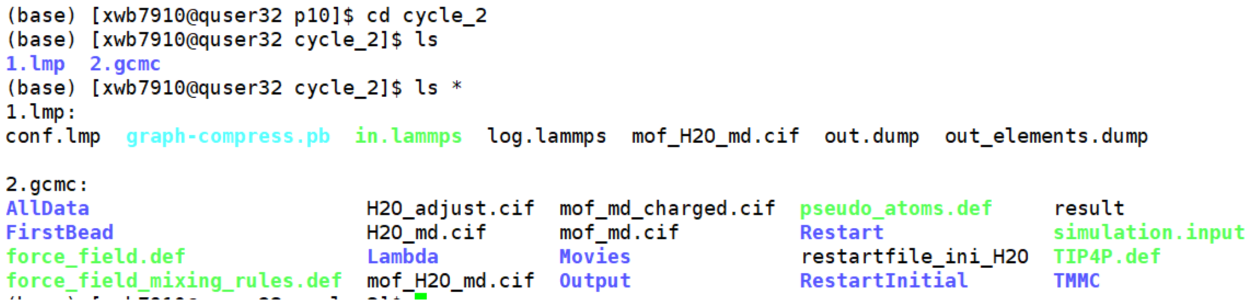
2.6. Output files

As the simulation runs, you will see folders: cycle_1/ cycle_2/ … cycle_100/

Inside each, you will see 1.lmp/ and 2.gcmc/ subfolders containing LAMMPS and gRASPA input and output files.



```
(base) [xwb7910@quser32 p10]$ ls
charge_222.txt             cycle_2   cycle_36  cycle_52  cycle_69  cycle_85
conf.lmp                   cycle_20  cycle_37  cycle_53  cycle_7   cycle_86
conf.lmp.bk                cycle_21  cycle_38  cycle_54  cycle_70  cycle_87
conf.lmp.initial           cycle_22  cycle_39  cycle_55  cycle_71  cycle_88
count_accepted_moves.sh    cycle_23  cycle_4   cycle_56  cycle_72  cycle_89
count_accepted_moves.txt   cycle_24  cycle_40  cycle_57  cycle_73  cycle_9
cycle_1                    cycle_25  cycle_41  cycle_58  cycle_74  cycle_90
cycle_10                   cycle_26  cycle_42  cycle_59  cycle_75  cycle_91
cycle_100                  cycle_27  cycle_43  cycle_6   cycle_76  cycle_92
cycle_11                   cycle_28  cycle_44  cycle_60  cycle_77  cycle_93
cycle_12                   cycle_29  cycle_45  cycle_61  cycle_78  cycle_94
cycle_13                   cycle_3   cycle_46  cycle_62  cycle_79  cycle_95
cycle_14                   cycle_30  cycle_47  cycle_63  cycle_8   cycle_96
cycle_15                   cycle_31  cycle_48  cycle_64  cycle_80  cycle_97
cycle_16                   cycle_32  cycle_49  cycle_65  cycle_81  cycle_98
cycle_17                   cycle_33  cycle_5   cycle_66  cycle_82  cycle_99
cycle_18                   cycle_34  cycle_50  cycle_67  cycle_83  gcmc
cycle_19                   cycle_35  cycle_51  cycle_68  cycle_84  in.lammps
```



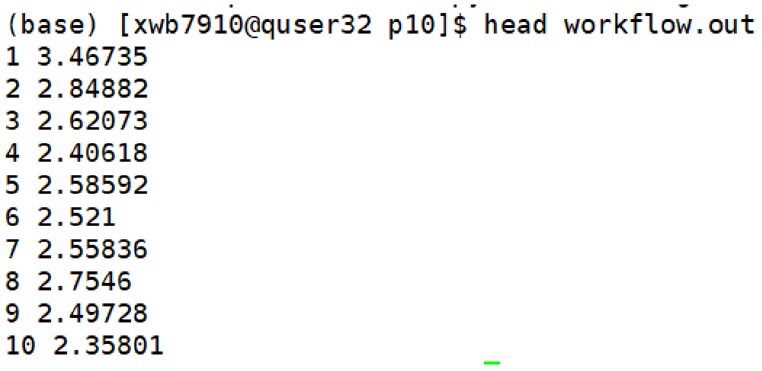
```
(base) [xwb7910@quser32 p10]$ cd cycle_2
(base) [xwb7910@quser32 cycle_2]$ ls
1.lmp  2.gcmc
(base) [xwb7910@quser32 cycle_2]$ ls *
1.lmp:
conf.lmp  graph-compress.pb  in.lammps  log.lammps  mof_H2O_md.cif  out.dump  out_elements.dump

2.gcmc:
AllData                        H2O_adjust.cif  mof_md_charged.cif  pseudo_atoms.def     result
FirstBead                      H2O_md.cif      mof_md.cif          Restart              simulation.input
force_field.def                Lambda          Movies              restartfile_ini_H2O  TIP4P.def
force_field_mixing_rules.def   mof_H2O_md.cif  Output              RestartInitial       TMMC
```

You can see a few ‘cif’ files.

mof_H2O_md.cif: The MOF+H2O CIF from 1.lmp
mof_md.cif: MOF part without charge.
mof_md_charged.cif: MOF part with charge.
H2O_md.cif: H2O part.
H2O_adjust.cif: H2O part but adjust all H2O molecules to TIP4P.

The H2O uptake values for each SR iteration are collected in the ‘workflow.out’ file in your calculation directory:



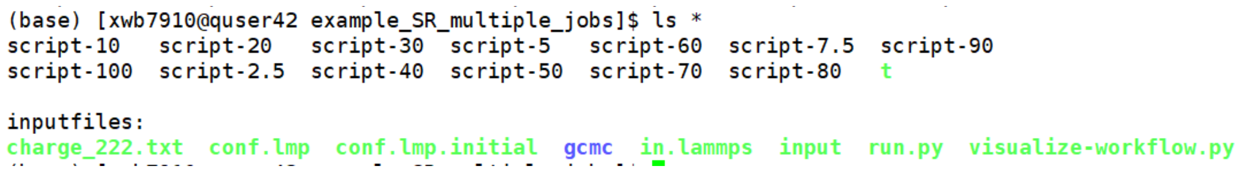
```
(base) [xwb7910@quser32 p10]$ head workflow.out
1 3.46735
2 2.84882
3 2.62073
4 2.40618
5 2.58592
6 2.521
7 2.55836
8 2.7546
9 2.49728
10 2.35801
```

2.7. Create Input Files for Multiple Pressures

A bash script can be used to generate separate folders for each target pressure. An example can be found at: /projects/b1013/xijun/NBO5/rerun_SR_5_9/example_SR_multiple_jobs/

First, prepare all necessary files in

/projects/b1013/xijun/NBO5/rerun_SR_5_9/example_SR_multiple_jobs/inputfiles/



```
(base) [xwb7910@quser42 example_SR_multiple_jobs]$ ls *
script-10    script-20    script-30   script-5    script-60   script-7.5   script-90
script-100   script-2.5   script-40   script-50   script-70   script-80    t

inputfiles:
charge_222.txt  conf.lmp  conf.lmp.initial  gcmc  in.lammps  input  run.py  visualize-workflow.py
```

Then, execute the bash script `./t_NPT` (this is for NPT MD, for example, see the bash script below) or `./t_NVT` (this is for NVT MD) to generate subfolders for each pressure:

```
#!/bin/bash

direct=`pwd`

x_list=(2.5 5 7.5 10 20 30 40 50 60 70 80 90 100)

pressure_list=(0.00130  0.00260  0.00390  0.00520  0.01039  0.01559  0.02079  0.02598  0.03118
0.03638 0.04157 0.04677 0.05197) # unit: bar

fugacity_list=(125.936681   251.83038   377.681175   503.488999   1005.295208   1507.348018
2008.675037 2508.309254 3008.171091 3507.291647 4004.707549 4502.329854 4998.786223)
# unit: Pa

for i in "${!x_list[@]}"; do
x=${x_list[$i]}
pressure=${pressure_list[$i]}
fugacity=${fugacity_list[$i]}
echo $x
mkdir -p $direct/p$x
cp -r $direct/inputfiles/* $direct/p$x/
sed -i "/pressure/c pressure = $fugacity" $direct/p$x/input
sed -i "s/fix             1 all nvt temp 300.0 300.0 0.04/fix             1 all npt temp 300.0 300.0 0.04 iso
$pressure $pressure 1.0/g" $direct/p$x/in.lammps
sed -i "s/fix             1 all npt temp 300.0 300.0 0.04 iso 1.0 1.0 1.0/fix             1 all npt temp 300.0
300.0 0.04 iso $pressure $pressure 1.0/g" $direct/p$x/in.lammps # I used this and previous lines
to control set NPT parameters in in.lammps.
cd $direct/p$x
```

```
#rm -rf cycle_*
#sbatch script-GPU
#python3 run.py
#H2O_uptake=`cat workflow.out|awk '{print $2}'|awk '{sum += $1; count += 1} END {if (count >
0) print sum / count}'`
#echo $x $H2O_uptake
cd $direct
done
```

Feel free to modify the script as needed to meet your specific goals.

Note: Always double-check all the parameter settings in cycle_XX/1.lmp and 2.gcmc to ensure the simulations are set up correctly.

3. Overview of Source Files

The MOFAFF package consists of several Python modules:

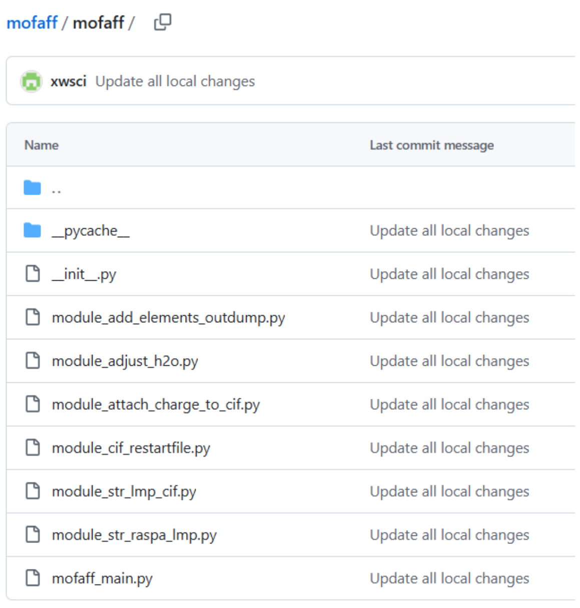
- module_attach_charge_to_cif.py:

  Defines the function attach_charge('filename.cif', 'charge_path') that inserts atomic charge data from charge.txt into a given CIF file.

- module_add_elements_outdump.py:

  Defines the function out_dump_add_element(input_file, output_file, mapping) that adds element labels to the LAMMPS output file.
  Example:

  ```
  out_dump_add_element('out.dump',
  'out_elements.dump', type_to_element)
  ```



mofaff / mofaff /

xwsci Update all local changes

Name	Last commit message
..	
__pycache__	Update all local changes
__init__.py	Update all local changes
module_add_elements_outdump.py	Update all local changes
module_adjust_h2o.py	Update all local changes
module_attach_charge_to_cif.py	Update all local changes
module_cif_restartfile.py	Update all local changes
module_str_lmp_cif.py	Update all local changes
module_str_raspa_lmp.py	Update all local changes
mofaff_main.py	Update all local changes

- module_str_lmp_cif.py:

  Defines 2 functions:

  1. convert_dump_cif(input_dump, output_cif) → Converts LAMMPS dump files with element data to CIF format. For example:

```
convert_dump_cif('out_elements.dump','mof_H2O_md.cif')
```

2. remove_water(input_filename,output_filename,n_mof_o): Remove water from the a MOF+H2O CIF file. For example:

   ```
   remove_water('mof_H2O_md.cif','mof_md.cif', 32)
   ```

   Remove H2O from 'mof_H2O_md.cif' and creating a cif file for dry MOF 'mof_md.cif'. '32' means the first 32 O atoms belong to MOF.

- module_adjust_h2o.py:

Defines 2 functions:

1. def separate_H2O(input_filename, output_filename, n_mof_o): Separate H2O and MOF atoms from a MOF+H2O cif file, creating a cif file for H2O. For example:

   ```
   separate_H2O('mof_H2O_md.cif','H2O_md.cif', 32)
   ```

   Separate mof_H2O_md.cif into H2O_md.cif, where '32' means the first 32 O atoms belong to MOF.
2. def adjust_H2O_TIP4P(input_filename): Adjust all H2O molecules to TIP4P structure, overwriting the original file. For example:

   ```
   adjust_H2O_TIP4P('H2O_adjust.cif')
   ```

- module_cif_restartfile.py:

  Defines convert_H2O_raspa_restartfile(input_cif, output_restartfile) to transform H2O coordinates obtained from LAMMPS MD simulations into a restart file for gRASPA. GCMC simulations will read the restartfile as the initial guess for water. For example:

  ```
  module_cif_restartfile.convert_H2O_raspa_restartfile('H2O_adjust.cif','restartfile_ini_H2O')
  ```

  Convert 'H2O_adjust.cif' into 'restartfile_ini_H2O'

- module_str_raspa_lmp.py:

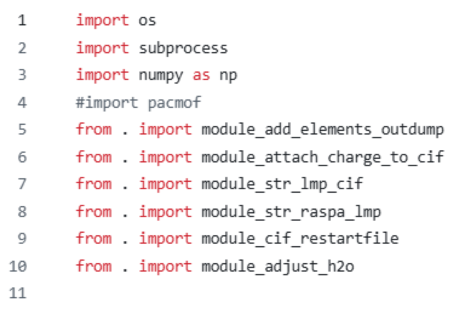
Defines the function "raspa_to_lmp(raspa_data, output, N_mof_O, mapping, hybrid_FF)": Convert Movies/System_0/result_XXX.data (the last snapshot in a GCMC simulation) into LAMMPS input format for the next SR iteration. For example:

```
raspa_to_lmp(' result_495000.data','conf.lmp', 32, type_to_element, False)
```

Convert the GCMC movie file 'result_495000.data' into 'conf.lmp'. The first '32' O atoms belongs to MOF structure. 'type_to_element' are element mapping defined in 'input' file. 'False' means not using hybrid FF (*I didn't write hybrid FF code, so, always set it to False in the 'input' file*).

- mofaff_main.py: The central driver script managing the full SR iteration workflow.

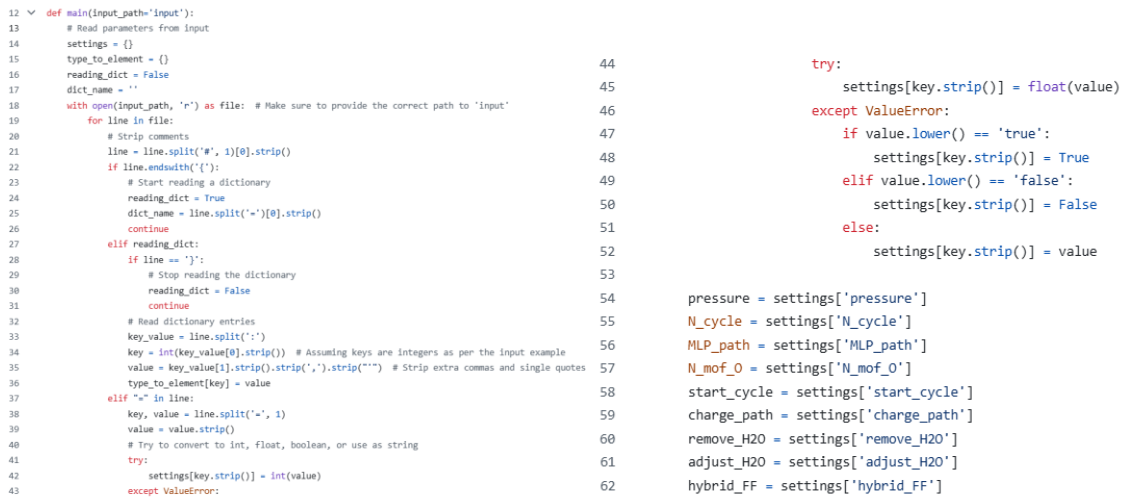
  Part 1. Import all the module files listed above.



```python
import os
import subprocess
import numpy as np
#import pacmof
from . import module_add_elements_outdump
from . import module_attach_charge_to_cif
from . import module_str_lmp_cif
from . import module_str_raspa_lmp
from . import module_cif_restartfile
from . import module_adjust_h2o

```

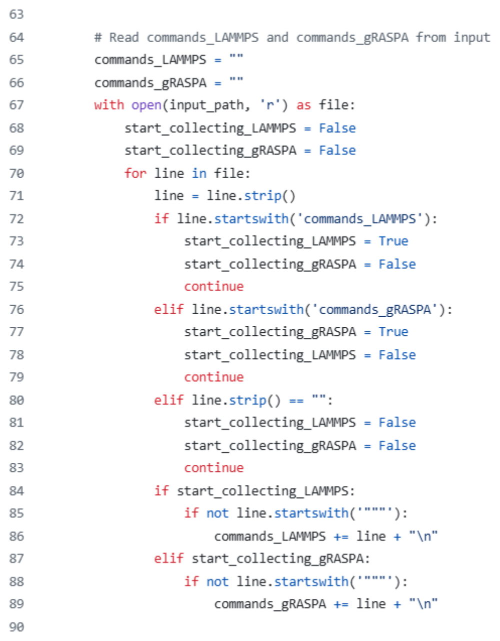
Part 2. Read SR parameter settings from the ‘input’ file, such as pressure, N_cyle, MLP_path, N_mof_O, start_cycle, charge_path, remove_H2O, adjust_H2O, hybrid_FF.



```python
def main(input_path='input'):
    # Read parameters from input
    settings = {}
    type_to_element = {}
    reading_dict = False
    dict_name = ''
    with open(input_path, 'r') as file:  # Make sure to provide the correct path to 'input'
        for line in file:
            # Strip comments
            line = line.split('#', 1)[0].strip()
            if line.endswith('{'):
                # Start reading a dictionary
                reading_dict = True
                dict_name = line.split('=')[0].strip()
                continue
            elif reading_dict:
                if line == '}':
                    # Stop reading the dictionary
                    reading_dict = False
                    continue
                # Read dictionary entries
                key_value = line.split(':')
                key = int(key_value[0].strip())  # Assuming keys are integers as per the input example
                value = key_value[1].strip().strip(',').strip("'")  # Strip extra commas and single quotes
                type_to_element[key] = value
            elif "=" in line:
                key, value = line.split('=', 1)
                value = value.strip()
                # Try to convert to int, float, boolean, or use as string
                try:
                    settings[key.strip()] = int(value)
                except ValueError:
                    try:
                        settings[key.strip()] = float(value)
                    except ValueError:
                        if value.lower() == 'true':
                            settings[key.strip()] = True
                        elif value.lower() == 'false':
                            settings[key.strip()] = False
                        else:
                            settings[key.strip()] = value

    pressure = settings['pressure']
    N_cycle = settings['N_cycle']
    MLP_path = settings['MLP_path']
    N_mof_O = settings['N_mof_O']
    start_cycle = settings['start_cycle']
    charge_path = settings['charge_path']
    remove_H2O = settings['remove_H2O']
    adjust_H2O = settings['adjust_H2O']
    hybrid_FF = settings['hybrid_FF']
```

Part 3. Read the LAMMPS and gRASPA environmental settings from the ‘input’ file.

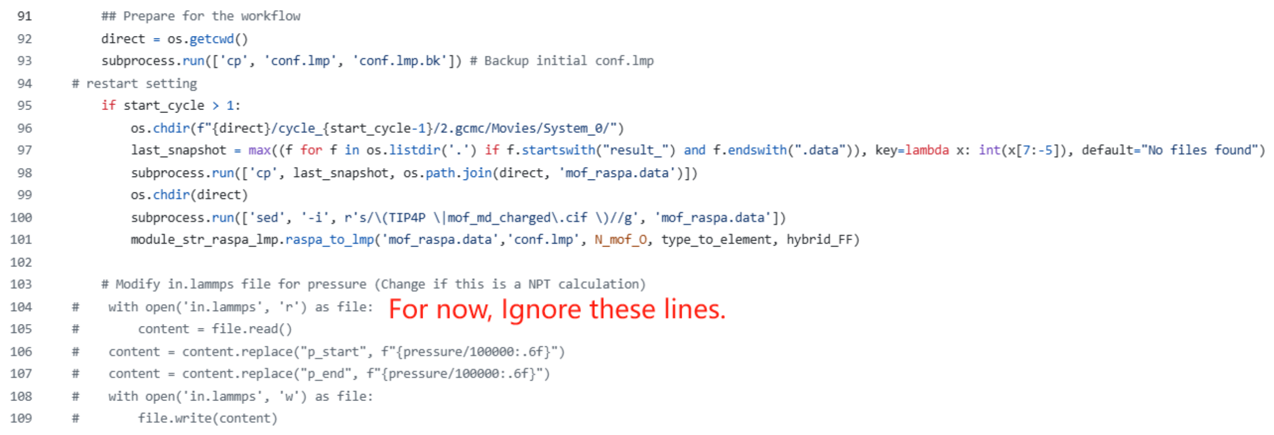


```python

    # Read commands_LAMMPS and commands_gRASPA from input
    commands_LAMMPS = ""
    commands_gRASPA = ""
    with open(input_path, 'r') as file:
        start_collecting_LAMMPS = False
        start_collecting_gRASPA = False
        for line in file:
            line = line.strip()
            if line.startswith('commands_LAMMPS'):
                start_collecting_LAMMPS = True
                start_collecting_gRASPA = False
                continue
            elif line.startswith('commands_gRASPA'):
                start_collecting_gRASPA = True
                start_collecting_LAMMPS = False
                continue
            elif line.strip() == "":
                start_collecting_LAMMPS = False
                start_collecting_gRASPA = False
                continue
            if start_collecting_LAMMPS:
                if not line.startswith('"""'):
                    commands_LAMMPS += line + "\n"
            elif start_collecting_gRASPA:
                if not line.startswith('"""'):
                    commands_gRASPA += line + "\n"

```

Part 4. Restart setup. If restart_cycle is set to x in the ‘input’ file, where x > 1, then the SR iteration will start from iteration x. The input structure for the LAMMPS part in iteration x is obtained from the final snapshot of GCMC from iteration x-1.

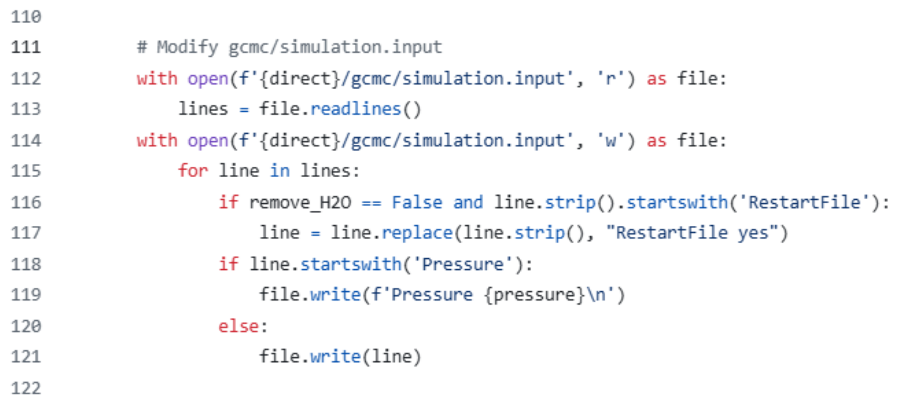


```
        ## Prepare for the workflow
        direct = os.getcwd()
        subprocess.run(['cp', 'conf.lmp', 'conf.lmp.bk']) # Backup initial conf.lmp
    # restart setting
        if start_cycle > 1:
            os.chdir(f"{direct}/cycle_{start_cycle-1}/2.gcmc/Movies/System_0/")
            last_snapshot = max((f for f in os.listdir('.') if f.startswith("result_") and f.endswith(".data")), key=lambda x: int(x[7:-5]), default="No files found")
            subprocess.run(['cp', last_snapshot, os.path.join(direct, 'mof_raspa.data')])
            os.chdir(direct)
            subprocess.run(['sed', '-i', r's/\(TIP4P \|mof_md_charged\.cif \)//g', 'mof_raspa.data'])
            module_str_raspa_lmp.raspa_to_lmp('mof_raspa.data','conf.lmp', N_mof_O, type_to_element, hybrid_FF)

        # Modify in.lammps file for pressure (Change if this is a NPT calculation)
    #    with open('in.lammps', 'r') as file:
    #        content = file.read()
    #    content = content.replace("p_start", f"{pressure/100000:.6f}")
    #    content = content.replace("p_end", f"{pressure/100000:.6f}")
    #    with open('in.lammps', 'w') as file:
    #        file.write(content)
```

For now, Ignore these lines.

Part 5. Automatically update the Pressure value in gcmc/simulation.input according to the pressure setting in the ‘input’ file.

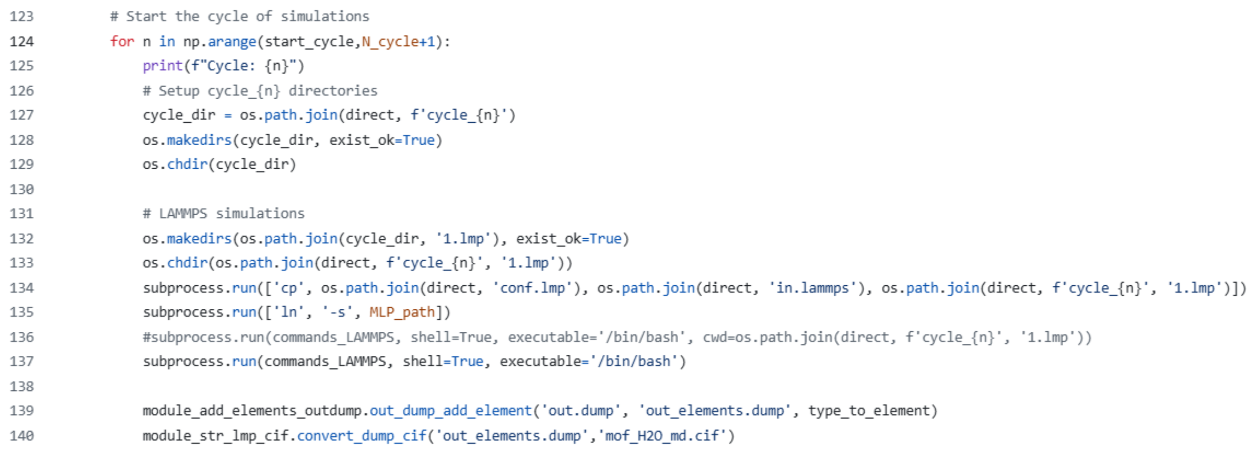


```

        # Modify gcmc/simulation.input
        with open(f'{direct}/gcmc/simulation.input', 'r') as file:
            lines = file.readlines()
        with open(f'{direct}/gcmc/simulation.input', 'w') as file:
            for line in lines:
                if remove_H2O == False and line.strip().startswith('RestartFile'):
                    line = line.replace(line.strip(), "RestartFile yes")
                if line.startswith('Pressure'):
                    file.write(f'Pressure {pressure}\n')
                else:
                    file.write(line)

```

Part 6. Run LAMMPS simulations using SR iterations (*in the code, these are referred to as SR “cycles,” though our recent manuscript refers to them as SR “iterations”*). After completing the LAMMPS simulations, add element information into the LAMMPS output file ‘out.dump’, creating a new ‘out_elements.dump’ file, and then convert it into a cif file ‘mof_H2O_md.cif’.

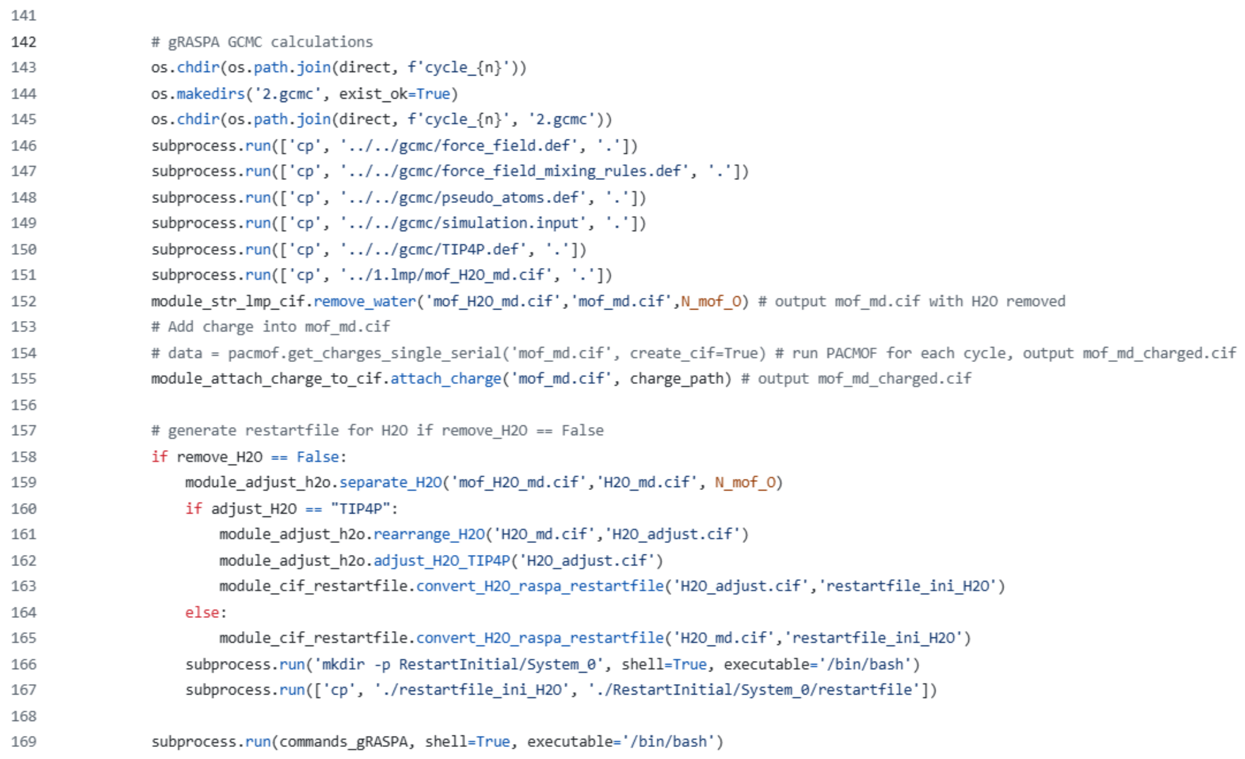


```
    # Start the cycle of simulations
    for n in np.arange(start_cycle,N_cycle+1):
        print(f"Cycle: {n}")
        # Setup cycle_{n} directories
        cycle_dir = os.path.join(direct, f'cycle_{n}')
        os.makedirs(cycle_dir, exist_ok=True)
        os.chdir(cycle_dir)

        # LAMMPS simulations
        os.makedirs(os.path.join(cycle_dir, '1.lmp'), exist_ok=True)
        os.chdir(os.path.join(direct, f'cycle_{n}', '1.lmp'))
        subprocess.run(['cp', os.path.join(direct, 'conf.lmp'), os.path.join(direct, 'in.lammps'), os.path.join(direct, f'cycle_{n}', '1.lmp')])
        subprocess.run(['ln', '-s', MLP_path])
        #subprocess.run(commands_LAMMPS, shell=True, executable='/bin/bash', cwd=os.path.join(direct, f'cycle_{n}', '1.lmp'))
        subprocess.run(commands_LAMMPS, shell=True, executable='/bin/bash')

        module_add_elements_outdump.out_dump_add_element('out.dump', 'out_elements.dump', type_to_element)
        module_str_lmp_cif.convert_dump_cif('out_elements.dump','mof_H2O_md.cif')
```

Part 7. Separate H2O and MOF atoms in ‘mof_H2O_md.cif’ into two files: ‘mof_md.cif’ and ‘H2O_md.cif’. Next, add charge (the charge file path is defined in the ‘input’ file) for each atom in ‘mof_md.cif’, creating ‘mof_md_charged.cif’. If you set adjust_H2O = True in the ‘input’ file, the H2O molecules in ‘H2O_md.cif’ will be adjusted into TIP4P in ‘H2O_adjust.cif’, which will then be converted into GCMC restart file and be put in /RestartInitial/System_0/restartfile. Such that GCMC simulations can utilize the water molecules as the initial guess.



```

        # gRASPA GCMC calculations
        os.chdir(os.path.join(direct, f'cycle_{n}'))
        os.makedirs('2.gcmc', exist_ok=True)
        os.chdir(os.path.join(direct, f'cycle_{n}', '2.gcmc'))
        subprocess.run(['cp', '../../gcmc/force_field.def', '.'])
        subprocess.run(['cp', '../../gcmc/force_field_mixing_rules.def', '.'])
        subprocess.run(['cp', '../../gcmc/pseudo_atoms.def', '.'])
        subprocess.run(['cp', '../../gcmc/simulation.input', '.'])
        subprocess.run(['cp', '../../gcmc/TIP4P.def', '.'])
        subprocess.run(['cp', '../1.lmp/mof_H2O_md.cif', '.'])
        module_str_lmp_cif.remove_water('mof_H2O_md.cif','mof_md.cif',N_mof_O) # output mof_md.cif with H2O removed
        # Add charge into mof_md.cif
        # data = pacmof.get_charges_single_serial('mof_md.cif', create_cif=True) # run PACMOF for each cycle, output mof_md_charged.cif
        module_attach_charge_to_cif.attach_charge('mof_md.cif', charge_path) # output mof_md_charged.cif

        # generate restartfile for H2O if remove_H2O == False
        if remove_H2O == False:
            module_adjust_h2o.separate_H2O('mof_H2O_md.cif','H2O_md.cif', N_mof_O)
            if adjust_H2O == "TIP4P":
                module_adjust_h2o.rearrange_H2O('H2O_md.cif','H2O_adjust.cif')
                module_adjust_h2o.adjust_H2O_TIP4P('H2O_adjust.cif')
                module_cif_restartfile.convert_H2O_raspa_restartfile('H2O_adjust.cif','restartfile_ini_H2O')
            else:
                module_cif_restartfile.convert_H2O_raspa_restartfile('H2O_md.cif','restartfile_ini_H2O')
            subprocess.run('mkdir -p RestartInitial/System_0', shell=True, executable='/bin/bash')
            subprocess.run(['cp', './restartfile_ini_H2O', './RestartInitial/System_0/restartfile'])

        subprocess.run(commands_gRASPA, shell=True, executable='/bin/bash')
```

Part 8. Output the H_2O loading results from GCMC to the output file ‘workflow.out’. Then, covert the final snapshot of GCMC to the LAMMPS input file ‘conf.lmp’ for use in the next SR iteration. Continue this SR iteration until the specified N_cycle value set in the ‘input’ file is reached.

```
        # Output pressure and H2O loading
        subprocess.run('cat Output/System_0_*.data >> result', shell=True, executable='/bin/bash')
        H2O_loading = float(subprocess.getoutput("grep -A 20 'LOADING: mol/kg' result | grep 'Overall' | tail -1 | awk '{print $3}' | sed 's/,//g'"))
        with open(os.path.join(direct, 'workflow.out'), 'a') as f:
            f.write(f"{n} {H2O_loading}\n")

        # Copy the last snapshot of GCMC to direct/conf.lmp
        os.chdir(f"{direct}/cycle_{n}/2.gcmc/Movies/System_0/")
        last_snapshot = max((f for f in os.listdir('.') if f.startswith("result_") and f.endswith(".data")), key=lambda x: int(x[7:-5]), default="No files found")
        subprocess.run(['cp', last_snapshot, os.path.join(direct, 'mof_raspa.data')])
        os.chdir(direct)
        subprocess.run(['sed', '-i', r's/\(TIP4P \|mof_md_charged\.cif \)//g', 'mof_raspa.data'])
        module_str_raspa_lmp.raspa_to_lmp('mof_raspa.data','conf.lmp', N_mof_O, type_to_element, hybrid_FF)

        # Check for convergence
#    if n > 10:
#        with open(os.path.join(direct, 'workflow.out'), 'r') as f:
#            lines = f.readlines()
#            if abs(float(lines[-1].split()[1])-float(lines[-2].split()[1])) < 0.01:
#                break

# After finishing all cycles
print("Workflow complete!")


```