**Objective:**

Optimize Mie lambda-6 non-bonded parameters with vapor-liquid coexistence properties (saturated liquid density, rhol, saturated vapor density, rhov, saturated vapor pressure, Psat, enthalpy of vaporization, DeltaHv) by performing grand canonical Monte Carlo (GCMC) simulations with GPU Optimized Monte Carlo (GOMC) and reweighting with Multistate Bennett Acceptance Ratio (MBAR).

**Location:**

The code described in this manual is available at github/ramess101/MBAR\_GCMC and gitlab/ram9/ MBAR\_GCMC, although only the GitHub code will be actively maintained.

**Deliverables:**

This code was used for the following publication:

Messerly, R.A.; Barhaghi, M.S.; Potoff, J.J.; Shirts, M.R. “Histogram-free reweighting with grand canonical Monte Carlo: Post-simulation optimization of non-bonded potentials for phase equilibria." Journal of Chemical and Engineering Data, Under review.

**Overview:**

1. Force fields:
   1. MiPPE general (Potoff) – Mie Potentials for Phase Equilibria (note that the name MiPPE was not yet proposed until the last compound we simulated, cyclohexane)
   2. MiPPE short/long (Potoff\_SL) – Mie Potentials for Phase Equilibria
   3. TraPPE – Transferable Potentials for Phase Equilibria
   4. NERD – Nath Escobedo Revised de Pablo
2. State points:
   1. One bridge (near critical point)
   2. Two vapor phase
   3. 4-6 liquid phase (depending on compound)
3. Compounds (as named in directories):
   1. Cyclohexane: CYC6
   2. n-hexane: C6H14
   3. Isobutane: IC4H10
   4. Isooctane: IC8H18
   5. Neopentane: NEOC5H12
   6. 3-methyl-3-ethylpentane: 3M3EPentane
   7. 2,2-dimethylbutane: 22DMButane
   8. 2,3-dimethylbutane: 23DMButane
   9. 3,3-dimethylhexane: 33DMHexane
   10. 2,3,4-trimethylpentane: 234TMPentane

**High-level methodology:**

Runs series of GCMC simulations:

Equilibration simulation that does not output xyz coordinates (pdb file)

~1000 short production simulations that output xyz coordinates (pdb file)

Generate basis functions:

After each short production simulation:

Read in xyz coordinates

Recompute non-bonded energy for different Mie parameter sets (first set is same as actual simulation)

Store recomputed energies in his\_rr?\_basis\_function\_?

Remove pdb file (GOMC does not have a compressed file format)

After all ~1000 production simulations have completed:

Read in his\_rr”$iRun”\_basis\_function\_? files for “$iRun” (iRun is 0 to ~1000)

Compile his\_rr?\_basis\_function\_? files into single his”$iState”a.dat file in the histfiles/ directory

Perform matrix algebra to determine the repulsive and attractive basis functions (i.e., the contribution to the non-bonded energy without the epsilon\*sigma^6 or epsilon\*sigma^lambda scaling term)

After generating all of the basis function files:

**Necessary software:**

GPU Optimized Monte Carlo (development version with rings and rerun feature)

Python 2.7 (might not be compatible with 3.7)

**Directory architecture:**

**GOMC\_Scripts directory:** Contains the shell and Python scripts for creating input files, running simulations, and post-processing the data. The primary files are:

AlkanesViscosity.sh – The main script, basically the only file that you will execute directly.

GreenKubo\_analyze.py – Performs the Green-Kubo analysis.

**GOMC\_Conditions directory:**

“$Compound”\_Temp\_”$Model” – Contains the temperatures for each state point to simulate for “$Compound” with force field “$Model”

“$Compound”\_ChemPot\_”$Model” – Contains the chemical potentials for each state point to simulate for “$Compound” with force field “$Model”

“$Compound”\_box\_length – Contains the box lengths for each state point to simulate for “$Compound”

**GOMC\_PDB\_PSF directory:**

“$Abbreviated\_Compound” – Subdirectories are named based on the abbreviated compound name that is used internally by GOMC

“$Model” – Subsubdirectories use the force field model name. The directories contain:

STEP3\_START\_liq\_BOX\_0.pdb – The pdb (coordinates) file for liquid phase

STEP3\_START\_vap\_BOX\_0.pdb – The pdb file for vapor phase

STEP3\_START\_reservoir\_BOX\_1.pdb – The pdb file for reservoir

STEP3\_START\_liq\_BOX\_0.psf – The psf (force field) file for liquid phase

STEP3\_START\_vap\_BOX\_0.psf – The psf (force field) file for vapor phase

STEP3\_START\_reservoir\_BOX\_1.psf – The psf (force field) file for reservoir

**Non-automated steps and user commands:**

Create PDB, PSF, Temp, ChemPot, and box\_length files for new compound of interest in corresponding directory

Create output directory with name: “$Compound”/GOMC/GCMC/”$Model”/

Edit input parameters at top of run\_all\_GCMC or run\_all\_GCMC\_rings

Run run\_all\_GCMC or run\_all\_GCMC\_rings

After runs have completed, compile histogram files and generate basis function files:

Run compile\_rerun\_rings\_basis\_function.py with corresponding flags, e.g.:

compile\_rerun\_rings\_basis\_function.py -c CYC6 -ff MiPPE\_basis -NS 7 -NH 1250 -NB 2

Create eps\_sig\_lam\_basis file where rows correspond to basis rerun parameters

Run compute\_rings\_basis\_function.py with corresponding flags

compute\_rings\_basis\_function.py -c CYC6 -ff MiPPE\_basis -NS 7 -NB 2

After basis function files are created, estimate VLE properties for range of epsilon, sigma, etc., for example, the 2x2 scan of epsilon and sigma for a given lambda of CYC6:

Run recompute\_with\_basis\_functions\_rings\_sig\_scan.py after modifying the input parameters towards the top of the script

Run run\_multiple\_cyclohexane\_MBAR after modifying input parameters at the top of the script. This function calls cyclohexane\_VLE\_from\_basis\_functions.py, which computes VLE values using epsilon\_scaling for a given sigma value from the sig\_scan done above

**Step-by-step description of run\_all\_GCMC\_rings (see script for details):**

Input parameters:

Compound: Name as found in file paths

Compound\_PDB: Name as found in PDB files for GOMC

Model: Force field name

pin0: The initial pin offset

Number of processors:

Vapor and liquid phase usually receive a different allotment. In some versions of this code, the number of processors is how the jobs distinguish between vapor and liquid. Now, however, a separate variable called “phase” should be used for this purpose.

Number of stages for liquid and vapor can be same or different

NBASIS is the number of basis function reruns minus 1 (i.e., NBASIS=1 for CYC6 MiPPE)

Specify the path location for files – Verify these paths exist. If problems arise due to use of “~”, just hard code “/home/ram9” for example

Copies backup of job

Read in the temperature conditions

Read in the box\_length conditions

Read in the ChemPot conditions

Loop over all states:

Create new directory for iState

Copy input files for starting, restarting, and rerunning in GOMC

Replace placeholders in input files with conditions, compound, and paths

Copy genBasisFunctions(\_rings) and make executable

Determine whether this state point is liquid or vapor (see three different examples)

Replace placeholder for phase in input start file with liquid or vapor

Call genBasisFunctions.sh, this file does the following:

Loops over the number of stages:

If the first stage:

Starts an equilibrium GCMC run, pins job, waits till job is finished

For all other stages:

Restarts a production GCMC run, pins job, waits till job is finished

Loops over the number of basis functions:

Reruns production GCMC run with different force field parameters, pins job, waits till job is finished

Copies histogram files from rerun analysis

Script is supposed to wait till all GCMC runs are finished, but this does not appear to be working properly. Once this is fixed, it would be possible to automate the post-processing python scripts