Objective: Predict viscosities with several different force fields for normal and branched alkanes at saturated liquid and compressed liquid conditions using equilibrium molecular dynamics and Green-Kubo analysis.

Deliverables: This code was used for two separate publications:

Messerly, R.A.; Anderson, M.C.; Razavi, S.M.; Elliott, J.R. “Improvements and limitations of Mie λ-6 potential for prediction of saturated and compressed liquid viscosity." *Fluid Phase Equilibria,* Volume 483, 2019, pg 101-115, <https://doi.org/10.1016/j.fluid.2018.11.002>

Messerly, R.A.; Anderson, M.C.; Razavi, S.M.; Elliott, J.R. “Mie 16-6 force field predicts viscosity with faster-than-exponential pressure dependence for 2,2,4-trimethylhexane.” *Fluid Phase Equilibria*, 2019. Under review.

Overview:

1. Force fields:
   1. TraPPE
   2. Potoff
   3. TAMie
   4. AUA4
2. State points:
   1. Saturated liquid – Uses REFPROP liquid densities for a given Tsat
   2. Compressed liquid – T = 293 K and either a prescribed density or prescribed pressure
3. Compounds (as named in directories):
   1. Ethane: C2H6
   2. Propane: C3H8
   3. N-Butane: C4H10
   4. N-octane: C8H18
   5. N-dodecane: C12H26
   6. N-Hexadecane: C16H34
   7. N-Docosane C22H46
   8. Isobutane: IC4H10
   9. Isopentane: IC5H12
   10. Isohexane: IC6H14
   11. Isooctane: IC8H18
   12. Neopentane: NEOC5H12
   13. 3-methylpentane: 3MPentane
   14. 2,3,-dimethylbutane: 23DMButane
   15. 2,4,4-trimethylhexane: 244TMHexane
   16. 2,2,5-trimethylhexane: 225TMHexane
   17. 2,2,4-trimethylhexane: 224TMHexane (IFPSC 10th challenge compound)

Method:

Can either run NPT first to determine box size for a given pressure, or skip the NPT stages by running NVT at predetermined box sizes

If running NPT to find box size:

Perform NPT equilibration with leapfrog integrator, 2 fs time-step, velocity-rescaling thermostat (time constant of 1), and Berendsen barostat (time constant of 1)

Perform NPT production with leapfrog integrator, 2 fs time-step, Nosé-Hoover thermostat (time constant of 1), and Parrinello-Rahman barostat (time constant of 5)

Perform NVT equilibration with velocity-verlet integrator, 2 fs time-step, and Nosé-Hoover thermostat (time constant of 1)

Perform NVT production with velocity-verlet integrator, 2 fs time-step, and Nosé-Hoover thermostat (time constant of 1)

If running with fixed bonds, uses LINCS constraint algorithm with lincs-order of 8

Necessary software:

GROMACS

Python

Directory architecture:

Scripts directory – Contains the shell and Python scripts for creating input files, running simulations, and post-processing the data. The two primary files that a user will interact with are:

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

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

Conditions directories:

Saturation\_Conditions – Contains the temperature and box size files, where box sizes are computed with REFPROP saturated liquid densities

Potoff\_Saturation\_Conditions – Contains the temperature and box size files, where box sizes correspond to Potoff force field literature values

TraPPE\_Saturation\_Conditions – Contains the temperature and box size files, where box sizes correspond to TraPPE force field literature values

TAMie\_Saturation\_Conditions – Contains the temperature and box size files, where box sizes correspond to TAMie force field literature values

T293highP\_Conditions – Contains the temperature (293 K), box size and/or pressure files

Gromacs (.mdp) input directories:

Tabulated – Contains .mdp files for tabulated potentials. Includes energy minimization, NPT equilibration, NPT production, NVT equilibration, and NVT production. Separate files for fixed bonds (LINCS) and flexible bonds (harmonic).

LennardJones- Contains .mdp files for Lennard-Jones potentials. Includes energy minimization, NPT equilibration, NPT production, NVT equilibration, and NVT production. Separate files for fixed bonds (LINCS) and flexible bonds (harmonic).

Gromacs (.gro and .top) input directories:

[Compound\_name]/Gromacs/Gromacs\_input – Contains a [Compound\_name].gro file with the molecular coordinates. Contains a [Compound\_name]\_temp.top file with placeholders for the force field parameters (some\_C6\_CH3, etc.) and a [Compound\_name] specific description of the atoms, bonds, angles, and dihedrals.

Step-by-step approach:

Create .gro file for compound of interest

Create template .top file for compound and force field of interest

Create output directory with name: “$Compound”/Gromacs/”$Conditions\_type”\_Viscosity/”$Model”\_N”$Nmol”\_”$BondType”

Specify parameters at top of AlkanesViscosity.sh, e.g., Compound, Model (force field), Conditions\_type, BondType, Temp, batches, NREPS, Nmol

Run AlkanesViscosity

Step-by-step description of AlkanesViscosity (see script for details):

System parameters – Specifies the compound, model, conditions, replicates, pinoffset, threads

Step size information – equil\_steps is usually set to 500000, prod\_steps of 500000 is sufficient for saturation viscosity but higher pressures/viscosities can require up to 48000000, output\_freq is typically 3 but can be increased (no greater than 18) for high pressures/viscosities

RDF information – Typically unused

NEMD information – Never fully developed

Path location – The path where scripts, conditions, experimental data, input, and output files are located. Important to verify that these paths exist. The user must create the output\_path directory.

jobfile – Creates a copy of the AlkanesViscosity.sh script that is submitted (to store the parameters) and appends this with other necessary files

Read in box size

Read in pressure if NPT = YES – Only need the pressures if running NPT to determine box size for NVT simulations

Read in temperatures

Equilibration time steps – Original (default) way of assigning equilibration times

Override steps - Uses the vector of equil\_steps and prod\_steps from “Step size information” to override the default values

Cut-off distance – Different non-bonded cut-off distance for Potoff model (1.0 nm), while other models use 1.4 nm cut-off

Assign force field parameters – Assigns CH3 bond length, epsilon, sigma, and lambda values for TraPPE, TraPPEAUA (TraPPE-2 for ethane), Potoff (which has both generalized, short and long), TAMie, and AUA4. Note that Potoff S/L only knows how to assign parameters for certain compounds.

Mdp path – If lambda is 12 you can use the native Lennard-Jones .mdp files (which run much faster) otherwise you must use the tabulated .mdp files

Copy experimental data – If performing saturation simulations copy TDE and REFROP saturation viscosities, otherwise copy REFPROP high pressure values

Create header for state point conditions which stores the simulated values of N, L, T in “$Conditions\_type”Settings.txt file

Loop through temperatures and densities – The variable “j” loops over the state points

Record state points – Write to “$Conditions\_type”Settings.txt file

Loop over number of batches – Variable “iRep” is for batches

Loop over number of replicates per batch – Variable “iMCMC” is antiquated. Really this is just a counter for the replicates in the iRep batch.

If using MCMC (Bayesian inference) non-bonded parameters – Read in epsilon and sigma values from predetermined distribution

If using MCMC for torsional is commented out because this has been moved into the force\_field\_params script

Make directories - MCMC\_”$iMCMC” keeps track of the replicate simulations, again iMCMC is a misnomer because we don’t always use MCMC for each replicate

Call force\_field\_params script – This file does the following:

Read in path locations

Read in the epsilon, sigma, and lambda values for CH3, CH2, CH, and C interactions

Read in bondlengths for CH3

Read in Nmol

Read in whether MCMC\_tors is YES or NO

Read in iMCMC to keep track of replicate count

For each CH3, CH2, CH, and C parameter set:

Call create\_tab.py script. This file does the following:

Creates a tabulated file (not necessary for LJ)

Converts epsilon, sigma, lambda to the GROMACS format (C6, Clambda)

Writes C6 and Clambda into files C6\_it and Clam\_it

Assign C6 and Clam variables by reading in C6\_it and Clam\_it files

For each pair of cross interactions:

For epsilon pairs:

Call comb\_rule.py script with -c geometric

Assign epsilon cross-interaction parameter by reading in comb\_rule file

For sigma pairs:

Call comb\_rule.py script with -c arithmetic

Assign sigma cross-interaction parameter by reading in comb\_rule file

The comb\_rule.py file does the following:

Reads in pair of epsilon or sigma

Applies geometric or arithmetic combining rules

Creates and writes to file called “comb\_rule”

For each cross-interaction pair:

Call create\_tab.py script (see description above)

Assign C6 and Clam variables by reading in C6\_it and Clam\_it files

Writes all non-bonded parameters into eps\_sig\_lam\_ref file

Writes bondlength\_CH3 into bondlength\_CH3 file

If not using MCMC sampling for torsional parameters:

Copy standard template .top file

If using MCMC sampling for torsional parameters:

Copy torsional template .top file

Read in MCMC parameter sets for four different torsions

Insert MCMC torsion parameter sets into .top file

Copy .gro file

Insert all C6 and Clam non-bonded parameters into .top file

Insert bondlength\_CH3 into .top file

Insert number of molecules into .top file

Initialize folders, copy files, insert variables:

Creates the Saturated folder (misnomer since simulations could also be non-saturation)

Creates rho”$j” folder

Creates “$nRep” folder – Note that this is deprecated because nRep will always be 1. But removing this would cause issues elsewhere

Insert molecules into box to create initial configuration – Utilizes GROMACS gmx insert-molecules command

Copy the minimization files – Copy .mdp files (found in mdp\_path) for performing steepest descent and L-BFGS energy minimization

Insert the non-bonded cutoff (rvdw) into the energy minimization .mdp files

Create NVT\_eq folder

Copy nvt\_eq.mdp from mdp\_path

Insert temperature, equilibration steps, cut-off into nvt\_eq.mdp

Create NVT\_prod folder – This is antiquated but needs to stay here for legacy purposes

Create NVT\_vis folder – This is the actual NVT “production” stage folder

Copy nvt\_vis.mdp from mdp\_path

Insert temperature, production steps, cut-off into nvt\_vis.mdp (ignore cos\_acceleration term, this refers to NEMD)

Run NPT steps to determine box sizes – If performing NPT first calls nAlkanesNPTsteps script, otherwise proceeds to run NVT steps (see below). Description of nAlkanesNPTsteps:

Reads in paths, system parameters, number of threads, replicates, steps, etc.

Loops over replicates (iMCMC)

Creates new folder for each replicate

Insert molecules into box to create initial configuration – Utilizes GROMACS gmx insert-molecules command

Creates NPT\_eq folder

Copies npt\_eq.mdp from mdp\_path

Inserts temperature, pressure, equilibration steps, cut-off into npt\_eq.mdp

Creates NPT\_prod folder

Copies npt\_prod.mdp from mdp\_path

Inserts temperature, pressure, production steps, cut-off into npt\_eq.mdp

First energy minimization – Performs steepest descent energy minimization

Loops over replicates (iMCMC)

Creates new folder for each replicate

Executes gromacs preprocessing command (gmx grompp) starting with configuration created by insert-molecules (“$Compound”\_box)

Executes gromacs mdrun on single thread (-nt 1) without GPUs (-nb cpu -pme cpu) in background

Assigns tasks based on pinoffset

Waits until all tasks are finished

Second energy minimization – Performs L-BFGS energy minimization

Loops over replicates (iMCMC)

Creates new folder for each replicate

Executes gromacs preprocessing command (gmx grompp) starting with configuration created by first energy minimization (em\_steep.gro)

Executes gromacs mdrun on single thread (-nt 1) without GPUs (-nb cpu -pme cpu) in background

Assigns tasks based on pinoffset

Waits until all tasks are finished

NPT equilibration period – Perform NPT equilibration

Loops over replicates (iMCMC)

Creates new folder for each replicate

Executes gromacs preprocessing command (gmx grompp) starting with configuration created by second energy minimization (../em\_l\_bfgs.gro)

Calls run\_single.sh script to execute gromacs mdrun on nt\_eq threads. The run\_single.sh code was written to handle the scenario where an equilibration simulation fails. This is important because a single failed simulation will halt all other replicates from proceeding to production. Description of run\_single.sh:

Read in necessary parameters

Perform gromacs mdrun

Pins jobs

Waits until job has finished

Exits if job was successful, otherwise proceeds to restart stage

Assigns variables based on whether NVT or NPT

Attempts up to 3 restarts

Each restart consists of:

Insert-molecules

First energy minimization

Second energy minimization

NVT/NPT equilibration

If no restart terminates appropriately, returns user warning

Waits until all run\_single.sh have finished

NPT production stage – Perform NPT production

Loops over replicates (iMCMC)

Creates new folder for each replicate

Executes gromacs preprocessing command (gmx grompp) starting with configuration created by equilibration (../../nvt\_eq.gro)

Executes gromacs mdrun on nt\_vis threads without GPUs (-nb cpu -pme cpu) in background

Assigns tasks based on pinoffset

Waits until all tasks are finished

Loops over replicates (iMCMC)

Computes average box length from NPT production log file and writes to Lbox\_NPT\_ave file

Reads in Lbox\_NPT from Lbox\_NPT\_ave

Insert molecules into box with length Lbox\_NPT to create initial configuration – Utilizes GROMACS gmx insert-molecules command

Run NVT steps

First energy minimization – Performs steepest descent energy minimization

Loops over replicates (iMCMC)

Creates new folder for each replicate

Executes gromacs preprocessing command (gmx grompp) starting with configuration created by insert-molecules (“$Compound”\_box)

Executes gromacs mdrun on single thread (-nt 1) without GPUs (-nb cpu -pme cpu) in background

Assigns tasks based on pinoffset

Waits until all tasks are finished

Second energy minimization – Performs L-BFGS energy minimization

Loops over replicates (iMCMC)

Creates new folder for each replicate

Executes gromacs preprocessing command (gmx grompp) starting with configuration created by first energy minimization (em\_steep.gro)

Executes gromacs mdrun on single thread (-nt 1) without GPUs (-nb cpu -pme cpu) in background

Assigns tasks based on pinoffset

Waits until all tasks are finished

Equilibration period – Performs NVT equilibration

Loops over replicates (iMCMC)

Creates new folder for each replicate

Executes gromacs preprocessing command (gmx grompp) starting with configuration created by second energy minimization (../em\_l\_bfgs.gro)

Assigns box length to Tempbox, either from liquid\_box if no NPT or from Lbox\_NPT\_ave

Calls run\_single.sh script to execute gromacs mdrun on nt\_eq threads. The run\_single.sh code was written to handle the scenario where an equilibration simulation fails. This is important because a single failed simulation will halt all other replicates from proceeding to production. Description of run\_single.sh:

Read in necessary parameters

Perform gromacs mdrun

Pins jobs

Waits until job has finished

Exits if job was successful, otherwise proceeds to restart stage

Assigns variables based on whether NVT or NPT

Attempts up to 3 restarts

Each restart consists of:

Insert-molecules

First energy minimization

Second energy minimization

NVT/NPT equilibration

If no restart terminates appropriately, returns user warning

Waits until all run\_single.sh have finished

Production/Viscosity period – Performs NVT production runs (a.k.a. viscosity runs)

Loops over replicates (iMCMC)

Creates new folder for each replicate

Executes gromacs preprocessing command (gmx grompp) starting with configuration created by equilibration (../../nvt\_eq.gro)

Executes gromacs mdrun on nt\_vis threads without GPUs (-nb cpu -pme cpu) in background

Assigns tasks based on pinoffset

Waits until all tasks are finished

Data analysis:

Optional uncomment to call separate recompile\_output script and exit AlkanesViscosity.sh. This is only necessary for high viscosity systems where prod\_steps is very large and, thus, the post-processing is extremely slow. Must modify recompile\_output so that it has the correct system parameters.

Skip NEMD section

Enter “Else” section

Loop over iMCMC replicates

Read in box length (Lbox) from either predefined liquid\_box or, if NPT = YES, from Lbox\_NPT\_ave

Compute box volume (Vbox) from Lbox

Analyze Green-Kubo and Einstein (really only use Green-Kubo method, but Einstein files are also created):

Utilize gromacs internal gmx energy command to generate vis\_out file that contains Green-Kubo integral values with respect to time

Wait until finished (depending on the system this can take minutes to weeks)

Skip RDF section

Remove large viscosity output files

Compile pressure output values from nvt\_vis.log:

If BondType is Harmonic log file includes extra harmonic energy term and so must read in different column entry

If Compound = Ethane log file is missing angle and dihedral average values and so must read in different column entry

If Compound = C3H8 log file is missing dihedral average values and so must read in different column entry

Concatenate pressure averages from all state points into press\_all\_log

If performed NPT simulations, compile the box lengths utilized in NVT stages into Lbox\_all file

Increment the NREP\_low and NREP\_high variables by the number of replicates (NREPS) that are performed in each batch

Loop back over iMCMC replicates

Loop back over iRep

All batches and replicates are finished

Call GreenKubo\_analyze.py, this code:

Reads in the vis\_out files from all replicates of a given state point

Averages the Green-Kubo integral

Fits integral to a double-exponential function

Estimates viscosity by evaluating the double-exponential fit in the infinite-time limit

Bootstraps uncertainty in viscosity by repeating fitting process hundreds of times with random subsets of replicates

Loop back over the state points (j)

All state points are finished

Call compare\_TDE\_REFPROP.py, this code:

Reads in the TDE/REFPROP experimental data at either saturation or high pressures

Reads in the simulation state points

Reads in the Green-Kubo viscosities and uncertainties

Plots comparison of simulation and experimental data

For saturation these plots should look like:

Top panel – Temperature vs Viscosity

Bottom panel – Temperature vs Percent Deviation from Experiment

For high pressure (T293highP) these plots should look like:

Top, left panel – Density vs Viscosity

Bottom, left panel – Density vs Percent Deviation from Experiment

Top, right panel – Pressure vs Viscosity

Bottom, right panel – Pressure vs Percent Deviation from Experiment

For saturation figure is saved as “compare\_TDE\_REFPROP\_sat.pdf”

For high pressure figure is saved as “compare\_REFPROP\_T293highP.pdf”