Thermal Conductivity Plugin: Getting Started User Guide

Last Updated September 17, 2017

Part A: Thermal conductivity Calculation:

Prerequisite:

1) LAMMPS executable is necessary to do molecular dynamics simulation for thermal conductivity calculation. If you do not have LAMMPS installed on your computer, please refer to LAMMPS website
http://lammps.sandia.gov/download.html

2) Python 3 is required to post process the temperature profile file, generated by LAMMPS, to compute thermal conductivity value. There are several Python vendors you can find online. If you do not have Python version 3 please refer to their installation guide.

https://www.python.org/downloads/

https://www.anaconda.com/download/

Download:

First of all, go to the MAGICS website, click below icon to download *Thermal_Conductivity_Calculation* module:

Thermal Conductivity Calculation

Unzipping *Lammps_thermal_conductivity_input_file.zip* will create a directory *Lammps_thermal_conductivity_input_file* that contains files necessary to perform the thermal conductivity calculation.

Molecular dynamics simulation for thermal conductivity calculation:

Lammps_thermal_conductivity_input_file contains following input files:

readme.txt: README to describes the steps necessary for thermal conductivity calculation.

MoS2.sw: Stillinger-Weber potential file for MoS_2.

MoS2.data: Unit cell configuration of MoS_2.

in.relax: LAMMPS input script to create a thermalized system at desired temperature T.

in.heatflux: LAMMPS input script to perform thermal conductivity calculation at temperature T.

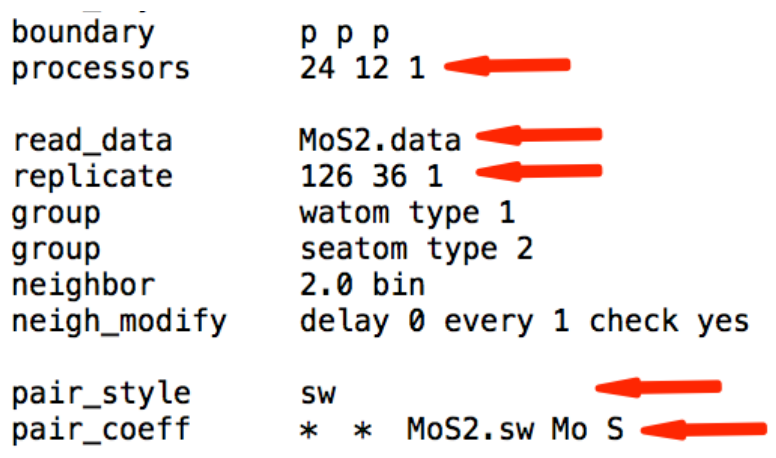
calthermal_conductivity.py: post-process the LAMMPS output file to compute thermal conductivity at temperature T and save images temperature profile of the system as a function of temperature.

Temperature.txt: A sample temperature profile of the system generated by LAMMPS. This is meant to be a quick test calculation of thermal conductivity value using calthermal_conductivity.py to make sure the Thermal Conductivity plugin works on your computing environment.

input.txt: contains input parameters for thermal conductivity calculation. These parameters are used by calthermal_conductivity.py.

NOTE: you need to put lammps_executable in this folder for LAMMPS calculation.

1) Create a relaxed configuration at temperature T by running LAMMPS and using in.relax as input file.
   The first a few lines of the code looks like this. The parameters that you would need to edit are highlighted by red arrows.



```
boundary        p p p
processors      24 12 1

read_data       MoS2.data
replicate       126 36 1
group           watom type 1
group           seatom type 2
neighbor        2.0 bin
neigh_modify    delay 0 every 1 check yes

pair_style      sw
pair_coeff      * *  MoS2.sw Mo S
```

   a) **processors**: define the number of processors that you want to use in each direction. Total number of processors(n_processors) used will be the multiple of these three values.
   b) **read_data**: contains the name of the input file of unit cell data. Here it is MoS2.data
   c) **replicate**: tells how many unit cells you want in each direction. Total system length in each direction will be the product of number of unit cell in that direction and the lattice constant in that direction. Lattice constant is defined inside unit cell configuration file (MoS2.data here).
   d) **pair_style**: the interaction potential type between atoms
   e) **parit_coeff**: contains the name of the file containing potential parameters (MoS2.sw) and mapping of different atom types in this file.

NOTE: Detailed description of above commands can be found on the LAMMPS website.

After defining all the above-mentioned parameters, run LAMMPS like below to create a thermalized system at temperature *T*.

```
> mpirun –np n_processors ./lammps_executable <in.relax
```

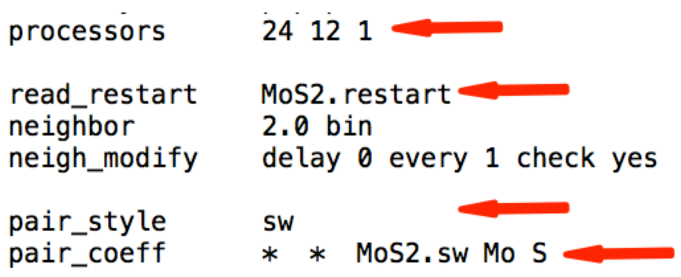
NOTE: The name of *lammps executable* depends upon how you have complied your LAMMPS.

2) After the thermalization step finishes, you will have following files.

   MoS2.restart: LAMMPS binary file for the thermal conductivity calculation.

   dump.300: contains position and velocity information of the relaxed system.

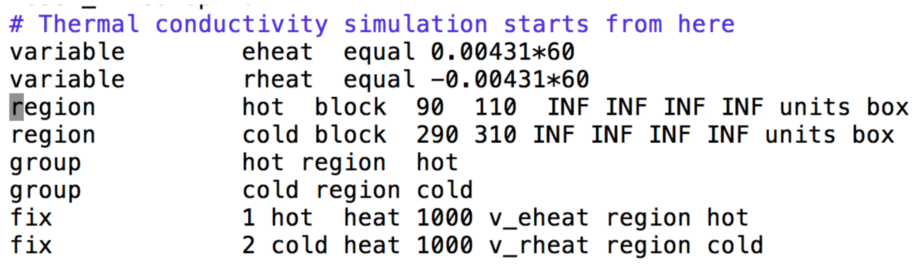
3) Next, we perform another MD simulation to induce the heat flux into the system using the LAMMP input file *in.heatflux*. Relevant parameters and lines in the LAMMPS input file are explained below.



```
processors      24 12 1

read_restart    MoS2.restart
neighbor        2.0 bin
neigh_modify    delay 0 every 1 check yes

pair_style      sw
pair_coeff      *  *  MoS2.sw Mo S
```



```
# Thermal conductivity simulation starts from here
variable          eheat  equal 0.00431*60
variable          rheat  equal -0.00431*60
region            hot  block  90  110  INF INF INF INF units box
region            cold block  290 310 INF INF INF INF units box
group             hot region  hot
group             cold region cold
fix               1 hot  heat 1000 v_eheat region hot
fix               2 cold heat 1000 v_rheat region cold
```

   a) The number of processors (**processors**), name of the input file (MoS2.restart), **pair_style** and **pair_coeff** depending upon your system. In general **processors**, **pair_style** and **pair_coeff** should remain same as in.relax.
   b) **eheat** and **rheat** are to control the amount to heat you wanted to put into the system. Note that the values of **eheat** and **rheat** should match to compensate the input and output heat.
   c) The value of **heat** (1000 here) in the **fix** command to control the frequency of how input heat into the system.
   d) **region hot** and **cold** define the location the system where you are putting/removing heat from the system.
   e) After defining all the above-mentioned parameters, run your simulation using the command given below

```
> mpirun –np n_processors ./lammps_executable <in.heatflux
```

NOTE: The name of *lammps executable* depends upon how you have complied your LAMMPS.

4) After the step to insert heal flux is done, you will have text files containing temperature profile information of the system as a function of system length. Name of these files will be *Temperatue10.txt*, *Temepature20.txt* and so on.

5) Run *calthermal_conductivity.py* to compute thermal conductivity based on the temperature profile files you obtained in the previous step. The Python script also plots the temperature profile of the system and save them into image. The various parameters required for thermal conductivity calculations are defined in the file input.txt. Beside that *calthermal_conductivity.py* takes name of the temperature profile file as command line argument. A sample temperature profile, **Temperature.txt**, is included as a quick test calculation as shown below

```
> python calthermal_conductivity.py Temperature.txt
```

NOTE:

input.txt: defines several parameters that are necessary for thermal conductivity calculation. These are the system dimension (height and width) used in thermal conductivity calculation. These values are written in LAMMPS dump file. Other parameters of input.txt are time step, amount of heat given/extracted from system and the frequency of head addition/ extracted from the system. You can find these parameters in your in.heatflux script. Change these parameters as per your simulation done using LAMMPS

Part B: PDOS, TDOS and C_V calculation:

Prerequisite:

1) LAMMPS executable is necessary to run and obtain LAMMPS dump file for DOS calculation. If you do not have LAMMPS installed on your computer, please refer to LAMMPS website
http://lammps.sandia.gov/download.html
2) Python 3 and GCC is required to calculate DOS and Cv from LAMMPS dump file.

Download:

Go to the MAGICS website, click below icon to download *Phonon_DOS_and_specific_heat* module:
https://magics.usc.edu/thermal-conductivity-plugin/

Phonon DOS and Specific heat

Unzipping *Phonon_DOS_and_specific_heat.zip* will create a directory *Phonon_DOS_and_specific_heat* that contains files necessary to perform the DOS and Cv calculations.

Density of States (DOS) and Specific heat (Cv) calculations

1) DOS_Cv_plugins contains following files

   readme.txt: steps required for DOS and C_v calculation

   MoS2.sw: Stillinger-weber potential file for MoS_2.

   MoS2.data: Unit cell configuration of MoS_2.

   in.dos: LAMMPS input script to create a thermalized system at desired temperature T.

   dos.c: C program to compute velocity autocorrelation, PDOS, TDOS, C_v from LAMMPS dump file

   dos.h: header file for the dos.c

   caldos.py: Python workflow to run dos.c and save images of velocity autocorrelation, PDOS, FDOS, C_v in the folder called image.

   input.txt: contains input parameters for DOS calculation. These parameters are used by dos.c

   dumpdos.nve: A sample LAMMPS dump file for quick calculation of DOS and C_v from caldos.py.

2) First step is to create a relaxed configuration at temperature T. Run LAMMPS using the input file *in.dos*.

   ```
   > mpirun –np n_processors ./lammps_executable < in.dos
   ```

   NOTE:

   a) in.dos defines the parameters **processors**, **read_data**, **replicate**, **pair_style** and **pair_coeff**.

   b) The name of *lammps executable* depends upon how you have complied your LAMMPS.

3) After the relaxation step using *in.dos* finished you will have an output file *dumpdos.nve*. *dumpdos.nve* contains the position and velocity information of the atoms that is used by dos.c to compute PDOS, FDOS and C_v.

4) Use *caldos.py* to compute partial DOS (PDOS), DOS and Cv. *caldos.py* will generate several text files and plots for PDOS, DOS and Cv in *image* folder. *caldos.py* also takes name of the LAMMPS dump file as input argument. A sample LAMMPS dump, *dosdump.nve*, is included in this module for you to run a quick test calculation like shown below.

```
> python caldos.py dumpdos.nve
```

NOTE:

input.txt defines several parameters that are necessary for DOS and C_v calculation, for example the number of initial condition, correlation length, difference between two initial condition and time step. You may want to tune the parameters for your system.