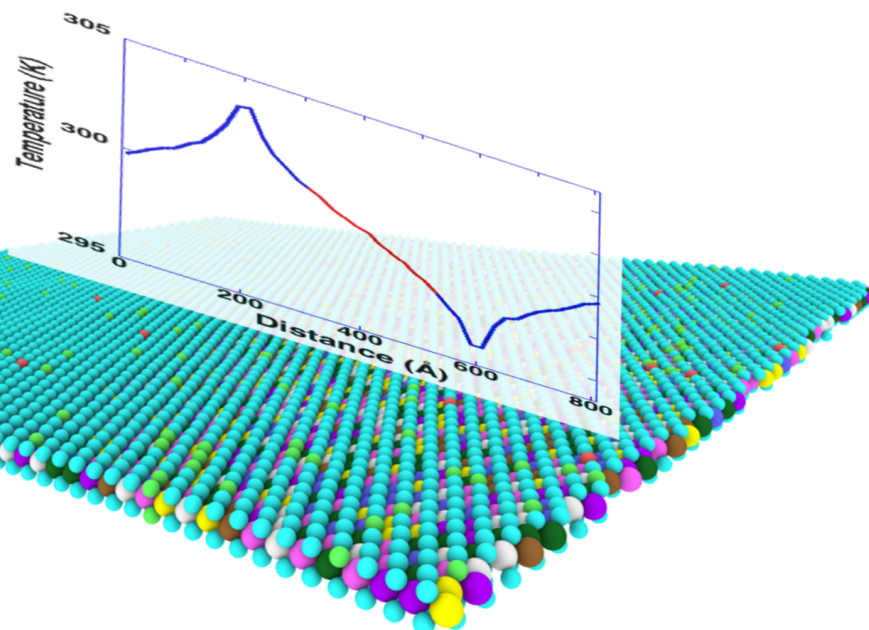
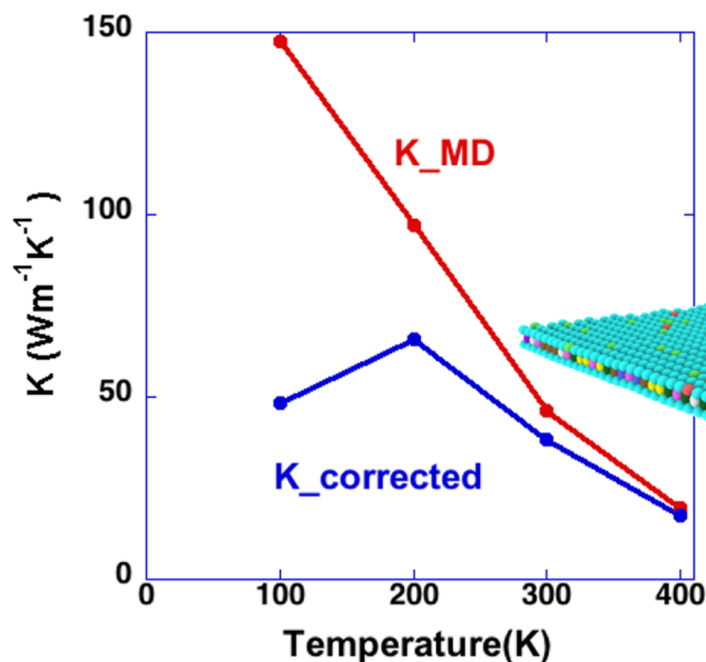


MAGICS Workshop

Thermal Conductivity Hands-On Session



This work was supported as part of the Computational Materials Sciences Program funded by the U.S. Department of Energy, Office of Science, Basic Energy Sciences, under Award Number *DE-SC00014607*

**The goal of this session is to calculate DOS,
 C_v and thermal conductivity of
2-D materials with quantum correction**

Outline

- Thermal conductivity calculation in MoS₂
 - Length scaling
 - Temperature scaling
- PDOS, DOS and C_v calculation
- Quantum correction for thermal conductivity

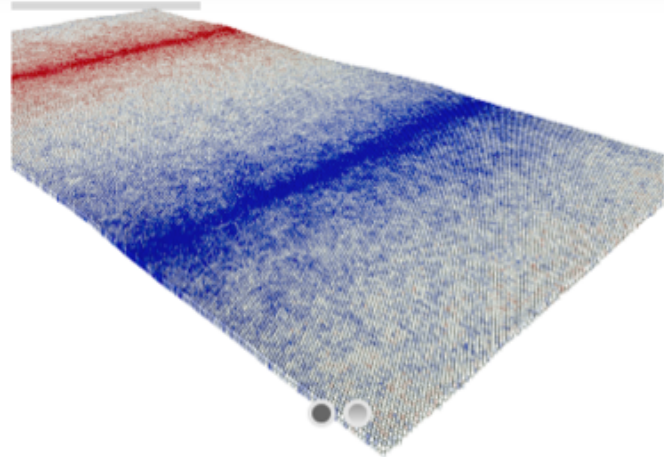
Software Download

LAMMPS Plugins for Thermal Conductivity Calculation

<https://magics.usc.edu/thermal-conductivity-plugin/>

Thermal Conductivity Plugins for LAMMPS

- Thermal conductivity with isotopes and quantum correction
- Velocity autocorrelation and Phonon Density of States using multiple initial conditions
- Specific heat from Phonon DOS as a function of temperature



Thermal conductivity tools is a series of plugins for thermal properties — velocity autocorrelation functions, phonon density of states, specific heat and thermal conductivity from MD using LAMMPS.

Software Download Links

SOURCE CODE




[Thermal Conductivity Calculation](#)



[Phonon DOS and Specific heat](#)

Documentation download

PDF of quick start guide can be downloaded [here](#) 

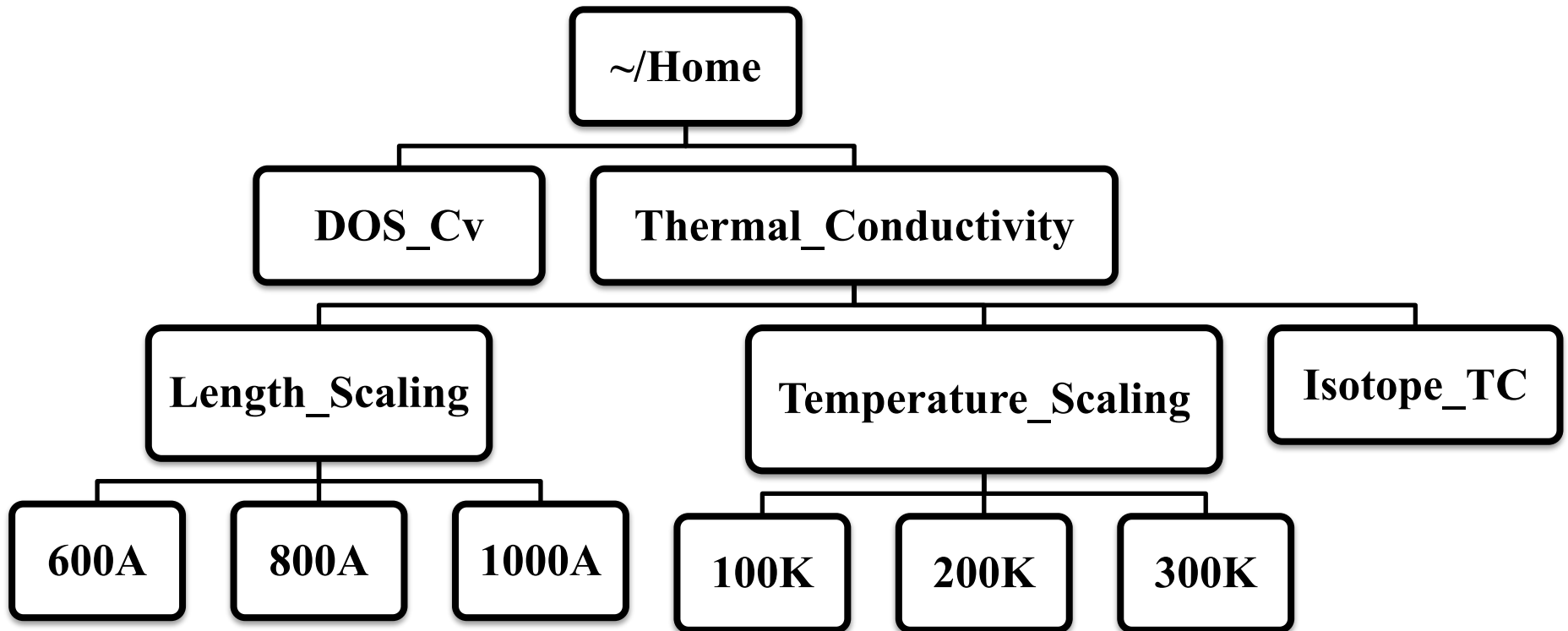
Frequently Questions

For frequently asked questions, please click [here](#) ?

Asked

LAMMPS Plugins for Thermal Conductivity

Contents of your Home Directory



LAMMPS Plugins for Thermal Conductivity

Each sub directory of Length_Scaling and Temperature_Scaling following files

1. **in.relax:** *LAMMPS script to create a relaxed system at a given Temperature*
2. **in.heatflux:** *LAMMPS script to for thermal conductivity calculation*
3. **in.variables:** *Variable definitions that are used in the LAMMPS Script*
4. **MoS₂.data:** *Input unit cell coordinate of MOS₂ monolayer*
5. **MoS₂.sw:** *SW Interaction potential for MOS₂*
6. **MoS₂.restart:** *Restart file to be used by LAMMPS.*
7. **job.pbs:** *PBS file used to submit jobs to the HPC*
8. **calthermal_conductivity.py:** *Python code to post-process LAMMPS data to compute thermal conductivity value*
9. **input.txt:** *Input parameters for calthermal_conductivity.py*

First Set of Job Submissions

STEP 1: Go to Thermal_Conductivity folder in your home directory

```
cd ~/Thermal_Conductivity
```



STEP 2: Navigate to Length_Scaling folder

```
cd ../Length_Scaling
```



STEP 3: Navigate to 1000A folder

```
cd ../1000A
```



STEP 4: Submit your first Job

```
qsub job.pbs
```

First Set of Job Submissions (Contd.)

STEP 1: Get back to Length_Scaling folder

```
cd ~/Thermal_Conductivity/Length_Scaling
```



STEP 2: Navigate to 800A folder

```
cd ./800A
```



STEP 3: Submit your second Job

```
qsub job.pbs
```


Steps for Thermal Conductivity Calculation

Step 1: Create a relaxed system at Temperature = TK

use LAMMPS script: in.relax



Step2: Add/remove heat to/from the system for 10-12ns

use LAMMPS script: in.heatflux



Step3: Compute thermal conductivity

post process LAMMPS output file using calthermal_conductivity.py

LAMMPS Walkthrough: Simulation Units

*Units used in LAMMPS here for Molecular dynamics Simulation – **Metal**.*

Example : Timestep is 1 Femtosecond = 0.001 (in LAMMPS Metal unit)

Parameter	Units
Distance	Angstroms (Å)
Time	Picoseconds (ps)
Energy	Electron Volts (eV)
Temperature	Kelvin (K)

Step 1: Create a relaxed system

```
units
atom_style
boundary
processors
read_data
replicate
group
group
neighbor
neigh_modify
pair_style
pair_coeff
thermo_style
thermo
dump
min_style
minimize
```

```
metal
atomic
p p p
${px} ${py} ${pz}
MoS2.data
${xnum} ${ynum} ${znum}
watom type 1
seatom type 2
2.0 bin
delay 0 every 1 check yes
sw
* * MoS2.sw Mo S
custom step temp pe ke etotal press vol
50
1 all atom 100 dump.min
cg
1.0e-8 1.0e-8 5000 10000
```

Input script: *in.relax*

1. *Create a system*

2. *Do energy minimization*

 *Input Crystal structure*

 *Interaction potential*

 *Energy minimization*

*Input file for in.heatflux
for thermal conductivity*

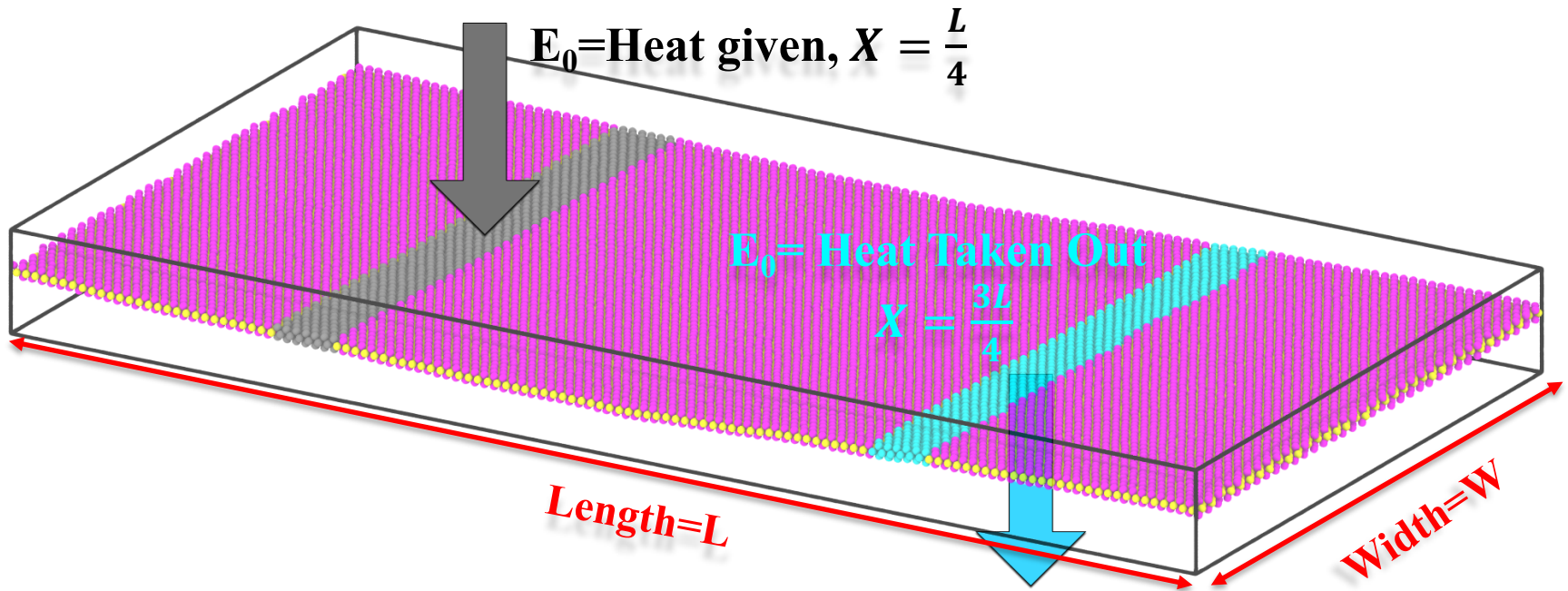
LAMMPS Walkthrough: Variables

File: *in.variables*

- *Contains variable definitions*
- *Here are some of the important ones*

```
variable px equal 20 # Number of processors in the x direction
variable py equal 4  # Number of processors in the y direction
variable pz equal 1  # Number of processors in the z direction
variable SimTimestep equal 0.001 # Timestep used. Here 1 femtosecond
variable Steps300K equal 300000 # Number of steps to run. Here 300000
variable HeatAdded equal 0.00431*100 # Amount of heat added. 0.431eV
variable HeatRemoved equal -0.00431*100 # Amount of heat removed
                                         0.431eV
```

Schematic of Thermal Conductivity Simulation Setup



- Define **Hot** and **Cold** region in the system at $X=L/4$ and $X=3L/4$ respectively
- E_0 heat is given to the hot region and E_0 heat is taken away from cold region.

Step 2: Add/Remove heat to/from system

Read relaxed structure coordinate

```
boundary      p p p
processors    ${px} ${py} ${pz}
read_restart  MoS2.restart
neighbor      2.0 bin
```

Input script: in.heatflux

- 1. Read relaxed structure coordinate*
- 2. Define two strips $L/2\text{\AA}$ apart*
- 3. Add/remove heat from these strip*
- 4. Run system for 4ns*
- 5. Take temperature average over 4ns*
- 6. Continue this process at least thrice*

Define strips and add/remove heat from these regions

```
region        hot  block  240 260  INF INF INF INF units box
region        cold block  740 760  INF INF INF INF units box
group         hot  region hot
group         cold region cold
fix           1 hot  heat  100 ${HeatAdded} region hot
fix           2 cold heat  100 ${HeatRemoved} region cold
```

Step 2: Add/Remove heat to/from system

Input script: *in.heatflux*

1. *Read relaxed structure coordinate*
2. *Define two strips $L/2\text{\AA}$ apart*
3. *Add/remove heat from these strip*
4. *Run system for 4ns*
5. *Take temperature average over 4ns*
6. *Continue this process for at least 3 time*

Take Temperature average for 4ns

```
compute      myKE all ke/atom
compute      Thot hot temp/region hot
compute      Tcold cold temp/region cold
variable     atemp atom c_myKE/(1.5*8.621738*0.00001)
timestep     ${SimTimestep}
fix          6b all ave/spatial 1 ${T_avg4ns} ${T_avg4ns} x lower
             20.0 v_atemp file Temperature.txt units box
run          ${T_avg4ns}
```


Hands on Calculations

Length Scaling:

1. Pre-relaxed system at 300K for 12ns is inside L_scaling folder:

a) $600\text{\AA} \times 100\text{\AA}$: MOS2_600L.restart

b) $800\text{\AA} \times 100\text{\AA}$: MOS2_800L.restart

c) $1000\text{\AA} \times 100\text{\AA}$: MOS2_1000L.restart

Already Submitted Earlier

2. Run in.heatflux for 300000 steps

4. Compute thermal conductivity using calthermal_conductivity.py

Job Submission: Job 3 (Length Scaling)

STEP 1: Navigate to Thermal_Conductivity folder in your home directory

```
cd ~/Thermal_Conductivity
```



STEP 2: Navigate to Length_Scaling folder

```
cd ./Length_Scaling
```



STEP 3: Navigate to 600A folder

```
cd ./600A
```



STEP 4: Submit your Third Job

```
qsub job.pbs
```

Hands on Calculations

Temperature Scaling:

1. Pre-relaxed system at 100K, 200K, 300K for 12ns is inside T-scaling folder:

a) $800\text{\AA} \times 100\text{\AA}$, 200K : MOS2_200T.restart

b) $800\text{\AA} \times 100\text{\AA}$, 100K : MOS2_100T.restart

c) $800\text{\AA} \times 100\text{\AA}$, 300K : MOS2_200T.restart



Already Submitted Earlier

2. Run in.heatflux for 300,000

3. Compute thermal conductivity using calthermal_conductivity.py

Job Submission: JOB1 (Temperature Scaling)

STEP 1: Get to Temperature_Scaling folder

```
cd ~/Thermal_Conductivity/Temperature_Scaling
```



STEP 2: Navigate to 100K folder

```
cd ./100K
```



STEP 3: Submit your fourth Job

```
qsub job.pbs
```

Step 3: Compute Thermal Conductivity

Compute thermal conductivity using *calthermal_conductivity.py*

➤ Input parameters for *calthermal_conductivity.py* is defined in **input.txt**

```
height      3.5      # height of your system.  
              # This value can be seen from lammps dump file  
width       197.45   # width of your system.  
              # This value can be seen from lammps dump file  
energy      0.2586   # energy input into the system in ev/ps.  
              # Should be equal to eheat/rheat of in.heatflux file  
frequency   1000     # how frequently heat is supplied/extracted from system.  
              # Should be equal to value of fix hot/cold in.heatflux file  
dtstep      0.001    # time step used in your md simulation in ps
```

Note: you can find these parameters in in.heatflux and lammps dump file

Sourcing Python

Before we start plotting anything we must source python. Follow these steps

STEP 1: Go to your home directory

```
cd ~/
```



STEP 2: Source Python

```
source /usr/usc/python/default/setup.sh
```

Step 3: Compute Thermal Conductivity

Compute thermal conductivity using *calthermal_conductivity.py*

```
$ python3.5 calthermal_conductivity.py Temperature.txt
```

Note: Temperature.txt is the output file created by in.heatflux

Output:

a) K and 1/K value

```
('Length:', 200.0000325)
('1/Length', 0.004999999187500132)
('Mean Thermal Conductivity value and standard deviation', 57.151683472393458, 0.60627590718458368)
('1/(Thermal_Conductivity) value and standard deviation', 0.017499275150576817, 0.00018644671358329857)
```

Step 3: Compute Thermal Conductivity (4ns data)

Compute thermal conductivity using *calthermal_conductivity.py*

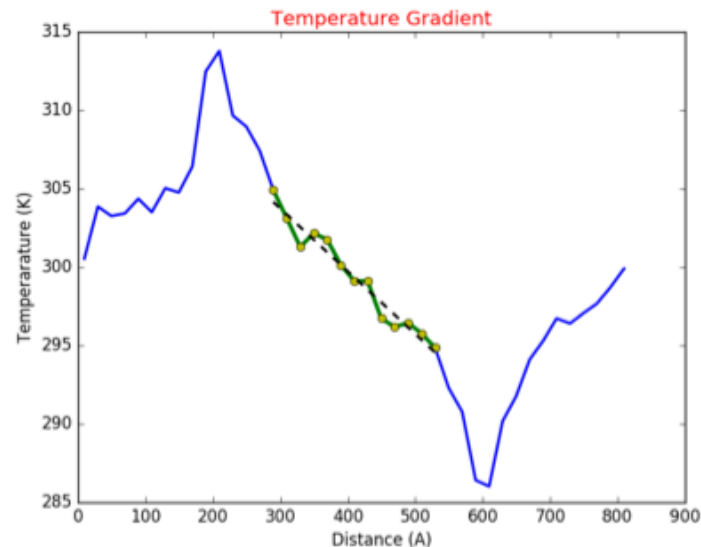
```
python3.5 calthermal_conductivity.py Temperature.txt
```

File: Temperature.txt

1. *Contains Temperature profile averaged over 300K Steps (0.3 ns)*
2. *Noisy*

Output : Length: 39.999983500000006
1/Length 0.02500001031250425
Mean Thermal Conductivity value and
standard deviation 249.719363936 0.0
1/(Thermal_Conductivity) value and
standard deviation 0.00400449522312
0.0

Also Outputs : 300K_steps.png



Step 3: Compute Thermal Conductivity (4ns data)

Compute thermal conductivity using *calthermal_conductivity.py*

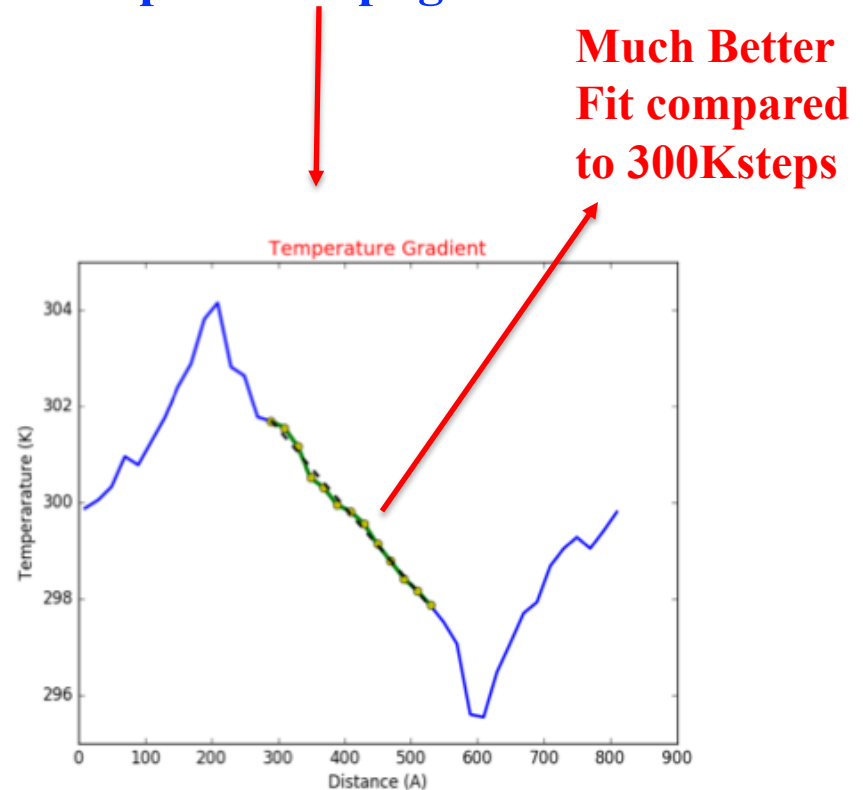
```
python3.5 calthermal_conductivity_4ns.py 20Temperature4.txt
```

File: 20Temperature4.txt

1. *Contains Temperature profile averaged over 4ns*
2. *Much Less Noise*

Output : Length: 39.999983500000006
1/Length 0.02500001031250425
Mean Thermal Conductivity value and
standard deviation 18.6176254622 0.0
1/(Thermal_Conductivity) value and
standard deviation 0.0537125425599
0.0

Also Outputs :4ns.png



Job Submission: JOB 2 Temperature Scaling

STEP 1: Navigate to Thermal_Conductivity folder in your home directory

```
cd ~/Thermal_Conductivity
```



STEP 2: Navigate to Temperature_Scaling folder

```
cd ./Temperature_Scaling
```



STEP 3: Navigate to 200K folder

```
cd ./200K
```



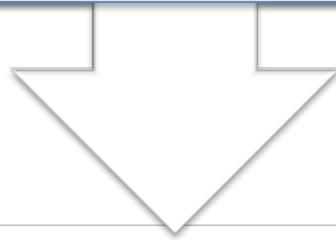
STEP 4: Submit your Fifth Job

```
qsub job.pbs
```

Job Submissions : Isotope (JOB 1)

STEP 1: Get to Isotope_TC directory

```
cd ~/Thermal_Conductivity/Isotope_TC
```

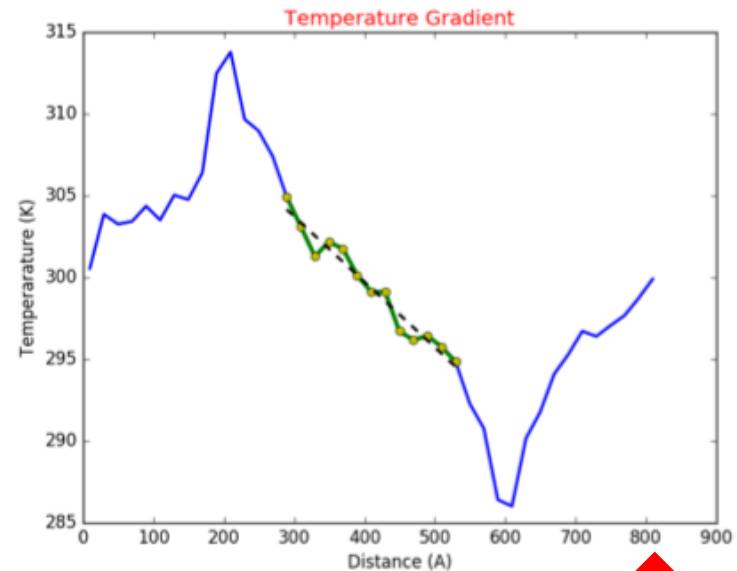
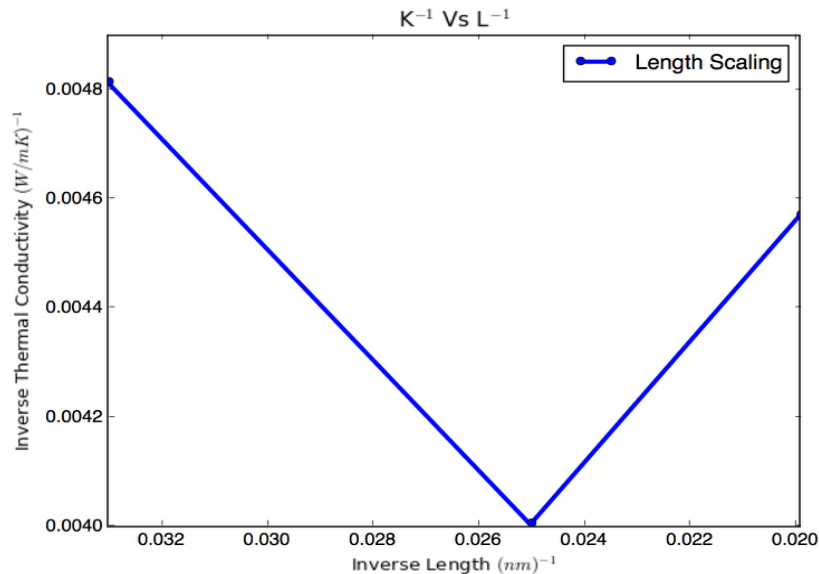


STEP 2: Submit job

```
qsub job.pbs
```

Step 4: Plot Length scaling

For $600\text{\AA} \times 100\text{\AA}$, $800\text{\AA} \times 100\text{\AA}$, $1000\text{\AA} \times 100\text{\AA}$
compute K value and plot K^{-1} vs L^{-1}

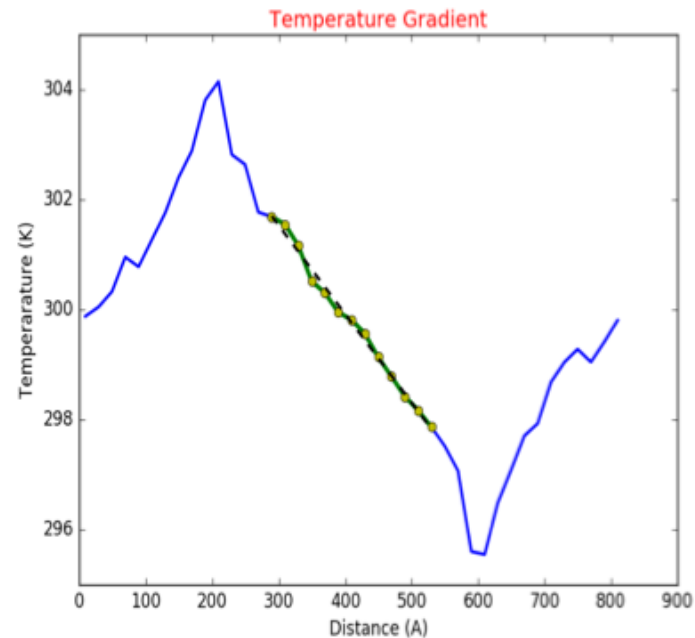
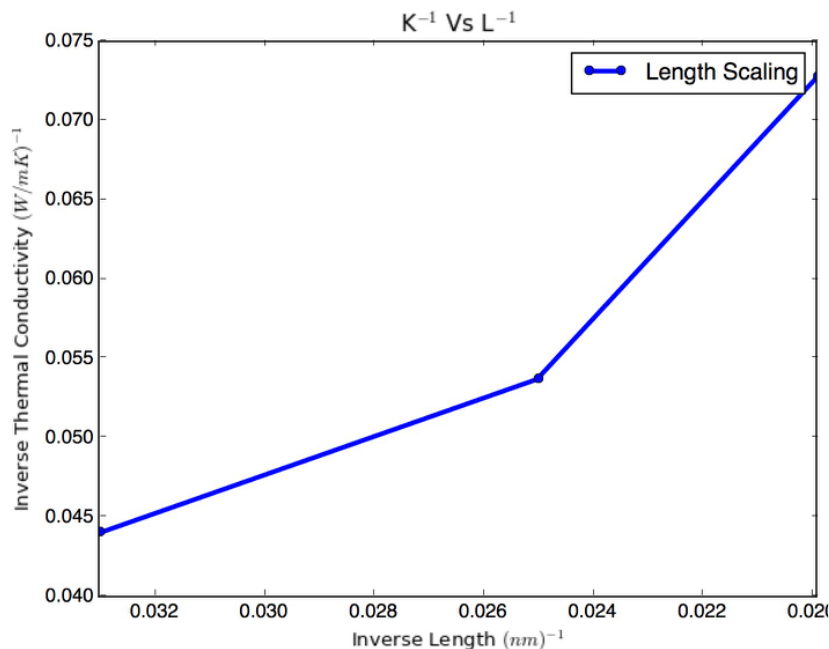


1. Poor Length Scaling because of insufficient time averaging
2. Extremely noisy temperature gradient

Step 4: Plot Length scaling (4ns System)

For $600\text{\AA} \times 100\text{\AA}$, $800\text{\AA} \times 100\text{\AA}$, $1000\text{\AA} \times 100\text{\AA}$ compute K value and plot K^{-1} vs L^{-1} using 20Temperature4.txt

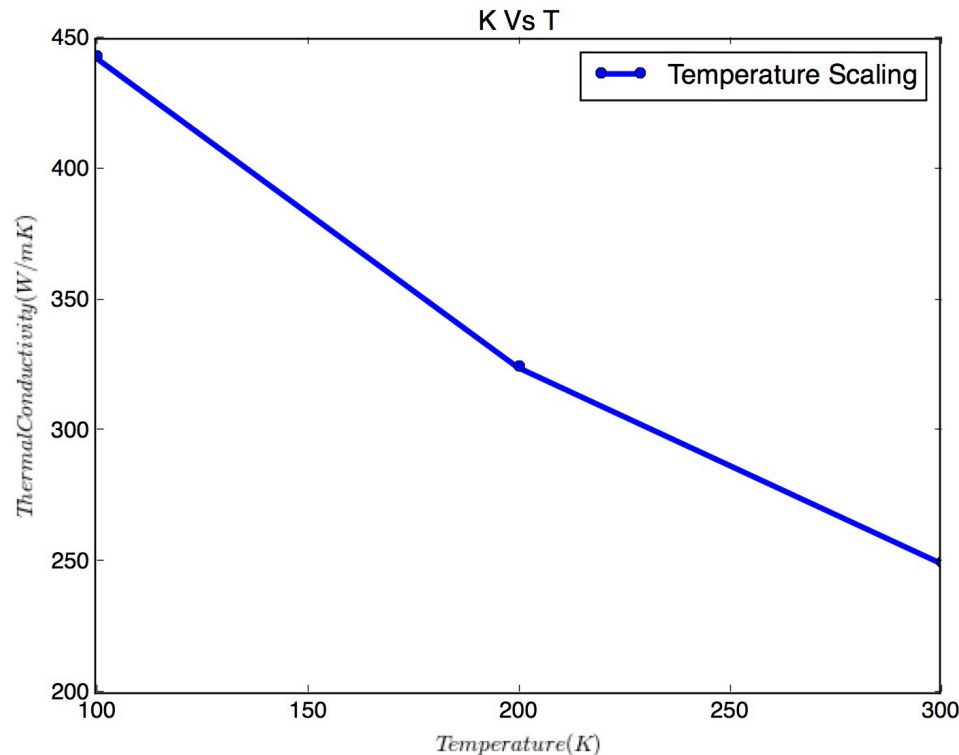
Note: Temperature profiles have been averaged over 4 ns instead of 0.3ns (300Ksteps)



Step 5: Plot Temperature scaling

For $800\text{\AA} \times 100\text{\AA}$ at temperature 100K, 200K and 300K
compute K value and plot K vs T

Note: For $800\text{\AA} \times 100\text{\AA}$ at 300K use the Thermal conductivity value from the Length Scaling System



PDOS, DOS and C_v Calculation

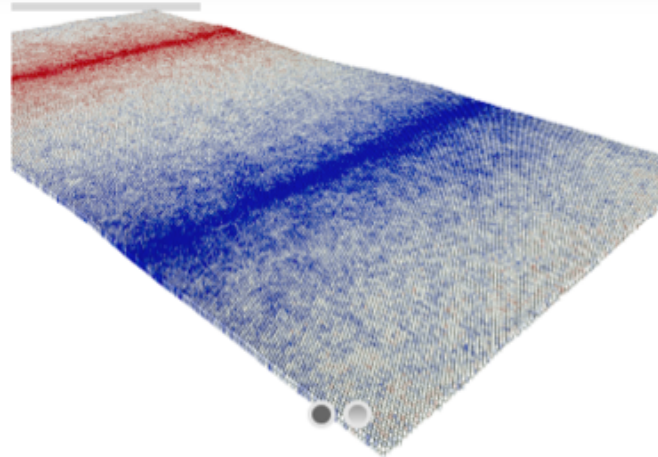
Software Download

LAMMPS Plugins for Thermal Conductivity Calculation

<https://magics.usc.edu/thermal-conductivity-plugin/>

Thermal Conductivity Plugins for LAMMPS

- Thermal conductivity with isotopes and quantum correction
- Velocity autocorrelation and Phonon Density of States using multiple initial conditions
- Specific heat from Phonon DOS as a function of temperature



Thermal conductivity tools is a series of plugins for thermal properties — velocity autocorrelation functions, phonon density of states, specific heat and thermal conductivity from MD using LAMMPS.

Software Download Links

SOURCE CODE



[Thermal Conductivity Calculation](#)



[Phonon DOS and Specific heat](#)

Documentation links

PDF quick start guide can be downloaded [here](#) 

download

Frequently Questions

For frequently asked questions, please click [here](#) ?

Asked

LAMMPS Plugins for Thermal Conductivity

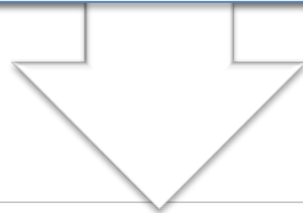
DOS_Cv: Contains following files

1. **in.dos:** *LAMMPS script to create a relaxed system at Temperature= T K*
2. **MOS₂.data:** *Input unit cell coordinate of MOS₂ monolayer*
3. **MoS₂.sw:** *SW Interaction potential for MOS₂*
4. **caldos.py:** *Python script to run dos.c and save images of velocity autocorrelation, PDOS, FDOS, C_v in the folder image.*
5. **dos.c:** *C program to compute velocity autocorrelation, PDOS, TDOS, C_v from LAMMPS dump file*
6. **dos.h:** *header file for dos.c*
7. **input.txt:** *contains input parameters for DOS calculation, used by dos.c*

Major steps involved in calculating DOS

Step 1: Create a relaxed system at a given Temperature

use LAMMPS script in.dos



Step2: Compute DOS from LAMMPS Dump file

use caldos.py

Step 1: Create a relaxed system

Input script: *in.dos*

1. *Create a system*
2. *Do energy minimization*
3. *Heat and relax to temperature T*
4. *Save coordinates for analysis*

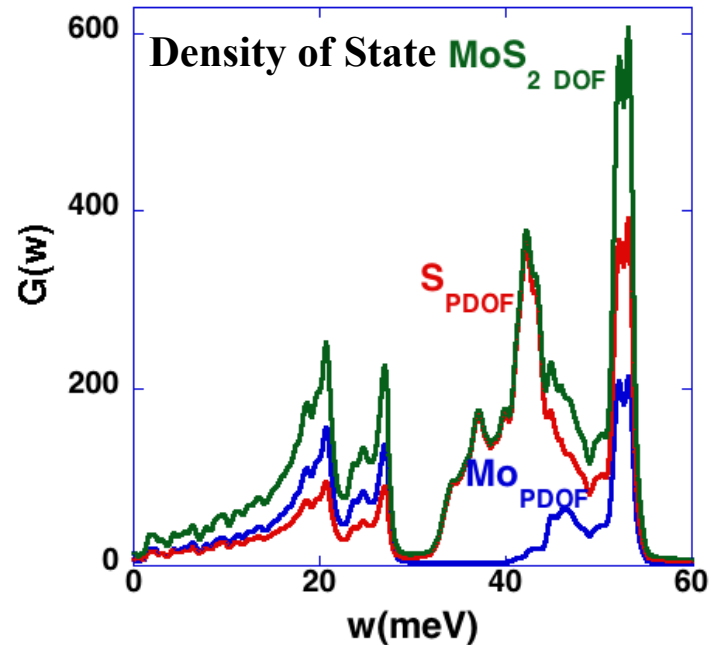
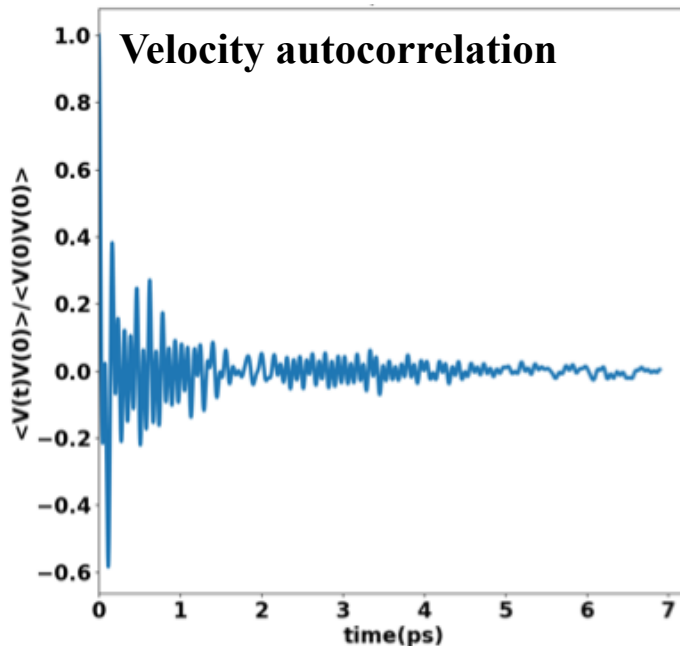
```
reset_timestep 0
fix 5 all nve
fix 6a all momentum 1000 linear 1 1 1
dump 3 all custom 1000 dump100relax.nve id type x y z vx vy vz
variable tt0 equal step
run 200000
undump 3
unfix 6a
#save coordinates of thermalized system for analysis
dump 4 all custom 10 dumpdos.nve id type x y z vx vy vz
dump_modify 4 sort id
run 10500
unfix 5
write restart mos2.restart
```



Used to calculate Velocity autocorrelation,
PDOS, DOS and Cv by caldos.py

Hands on Calculations

Compute Velocity autocorrelation, Density of states using LAMMPS dump file



- Compute Velocity autocorrelation (Z_α) for Mo & using 50 initial condition
- Partial Density of State:

$$G_\alpha(\omega) = \frac{6N_\alpha}{\pi} \int_0^\infty Z_\alpha(t) \cos(\omega t) dt$$

- Total Density of State :

$$G(\omega) = \sum_\alpha G_\alpha(\omega)$$

Hands on Calculations

Velocity autocorrelation, Density of states, C_v calculation:

1. Dump file for a relaxed system is inside DOS_cv_plugins folder:

➤ `dumpdos.nve`

2. Compute velocity autocorrelation, density of states and C_v

`python3.5 caldos.py dumpdos.nve`

*Note: **caldos.py** run a C program called **dos.c** which does this calculations. **dos.h** is header file for **dos.c**(no need to edit this file). Various parameters required for the calculation is defined inside **input.txt** file(you will need to change parameters here).*

Output : images and values of DOS, Velocity autocorrelations of each element and Specific Heat.

Input parameters for dos.c

Input parameter file: input.txt

```
Ninitial      10      #Total number of initial condition
Corlength     7000     #Correlation length for each initial condition
Ngap          100      #Gap between two initial condition
TFREQ         10       #Timestep between two consecutive saved frame
dT            0.001    #timestep in ps
massW         95.940   #mass of atom type 1
massSe        32.065   #mass of atom type 2
```

from in.dos file:

```
#save coordinates of thermalized system for analysis
dump          4 all custom 10 dumpdos.nve id type x y z vx vy vz
dump_modify   4 sort id
run           10500
unfix         5
write restart mos2.restart
```

TFREQ in input.txt

NFRAME

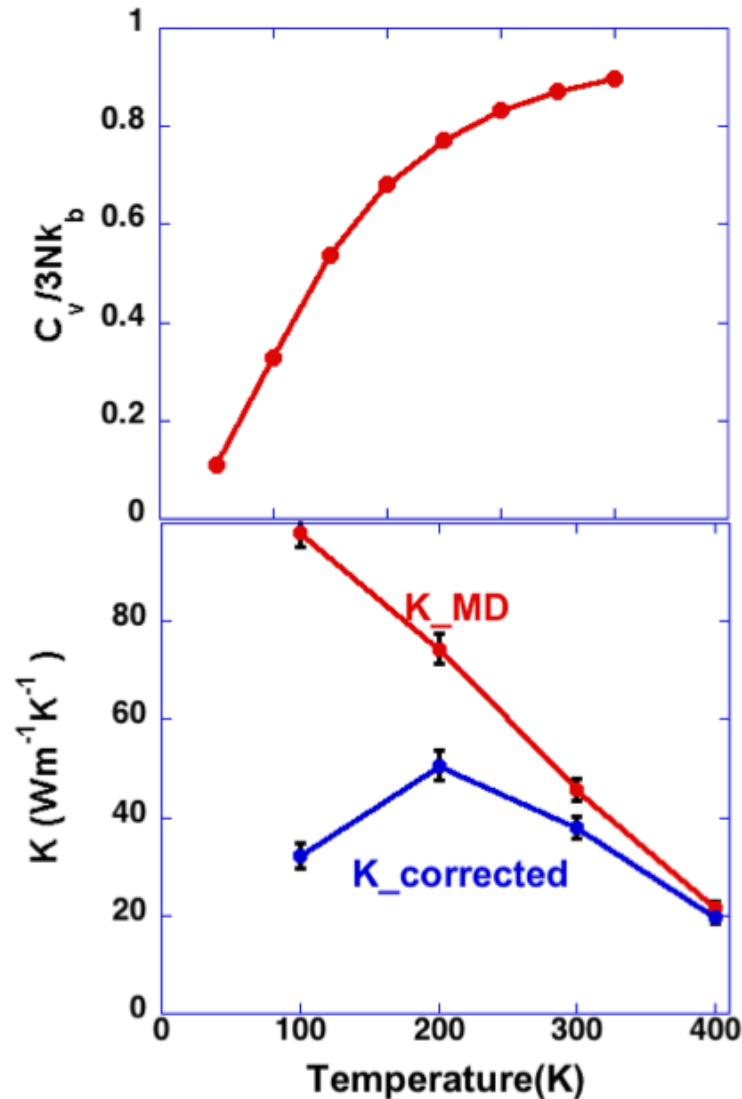
Note: $N_{\text{initial}} \times N_{\text{gap}} + \text{Corlength} \geq N_{\text{FRAME}}$

If this condition is not satisfied you will get error message

Hands on Calculations

1. Calculate Velocity autocorrelations, PDOS, DOS and C_v with one initial condition ($N_{\text{initial}}=1$) and with 10 initial condition ($N_{\text{initial}}=10$)
2. Compare the results

Quantum Corrected Thermal Conductivity



$$\frac{C_v}{3Nk_b} = \frac{\int_0^\infty \frac{u^2 e^u}{(e^u - 1)^2} G(\omega) d\omega}{\int_0^\infty G(\omega) d\omega}, \quad u = \frac{\hbar\omega}{k_B t}$$

$$K_{\text{corrected}} = \left(\frac{C_v}{3Nk_b} \right) \times K_{\text{MD}}$$

Hands-on Calculation:

1. Take the value of C_v at 100K, 200K and 300K from *Specific_heat.txt* file
2. Take the value of K computed at 100K, 200K and 300K for $800\text{\AA} \times 100\text{\AA}$ system
3. Multiply these two number to get quantum corrected K value

Hands on Calculations : Isotope Effect

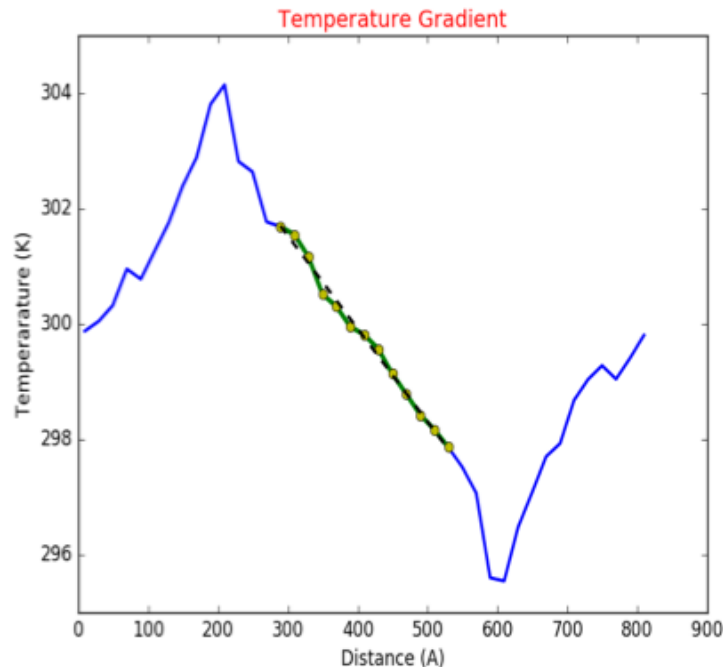
- Compute thermal conductivity value of $800\text{\AA} \times 100\text{\AA}$ at 100K with isotopes

python2.7 calthermal_conductivity.py Temperature.txt

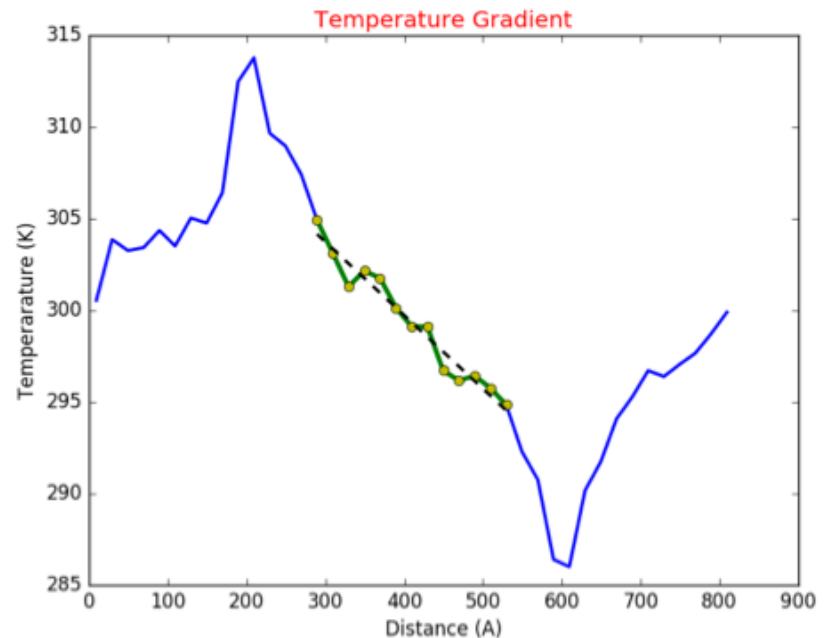
- Compare the thermal conductivity value of $800\text{\AA} \times 100\text{\AA}$ at 100K with and without isotopes

Stuff to keep in mind -1

Average temperature over longer time



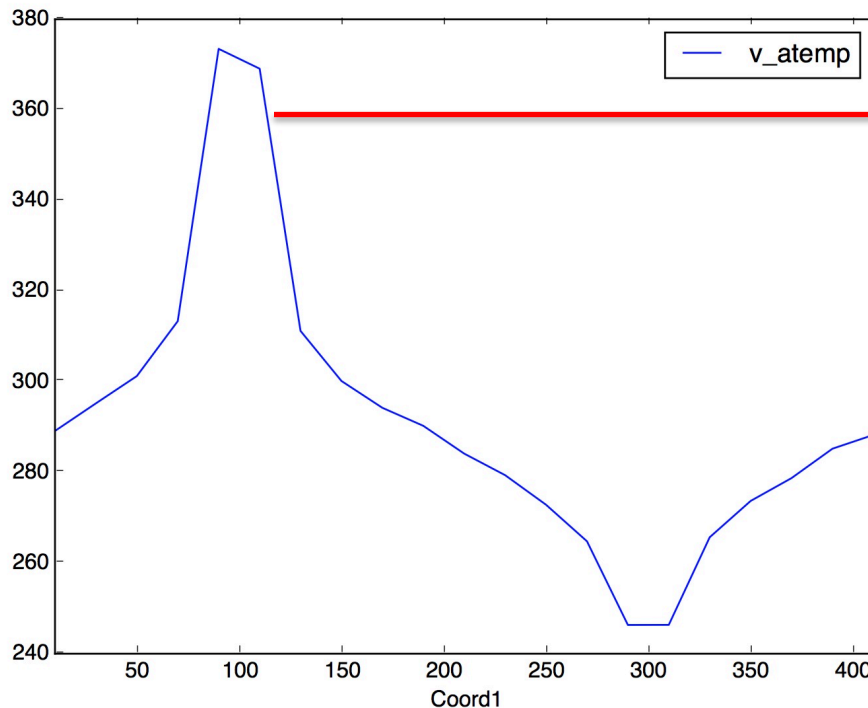
Averaging for longer gives a very clean gradient and almost a linear gradient



Extremely Noisy Temperature gradient. Gives erroneous slope values

Stuffs to keep in mind -2

Heating rate is important : Very high heating rate can give anomalous result



Note the weird bump at heat source. This can be avoided using a reasonable heating rate.
Also Note the rather flat heat sink.