1. 程序说明文档

AICON是一个旨在实现快速和准确的估计输运特性的程序，如导电率和热导率。

第一个版本的AICON是针对晶格导热率计算。它是基于德拜-卡拉威模型，但我

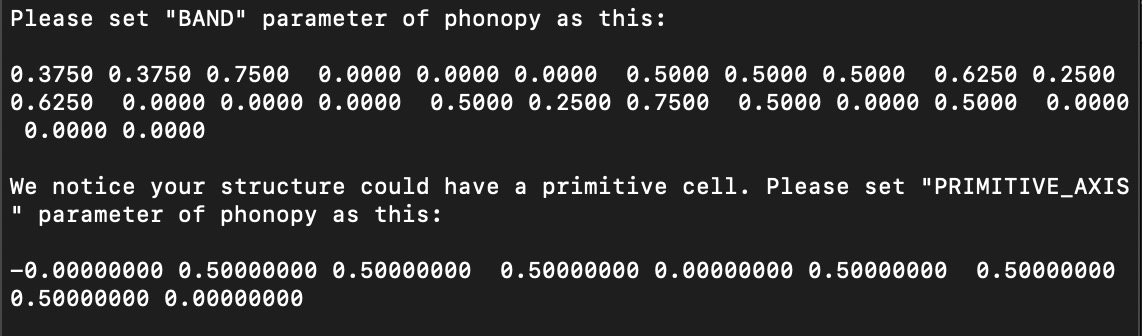
们更新了原始的公式，使其结果更加可靠。

该程序依赖于外部程序来生成它所需要的输入文件(POSCAR, band.yaml, gruneisen.yaml)。

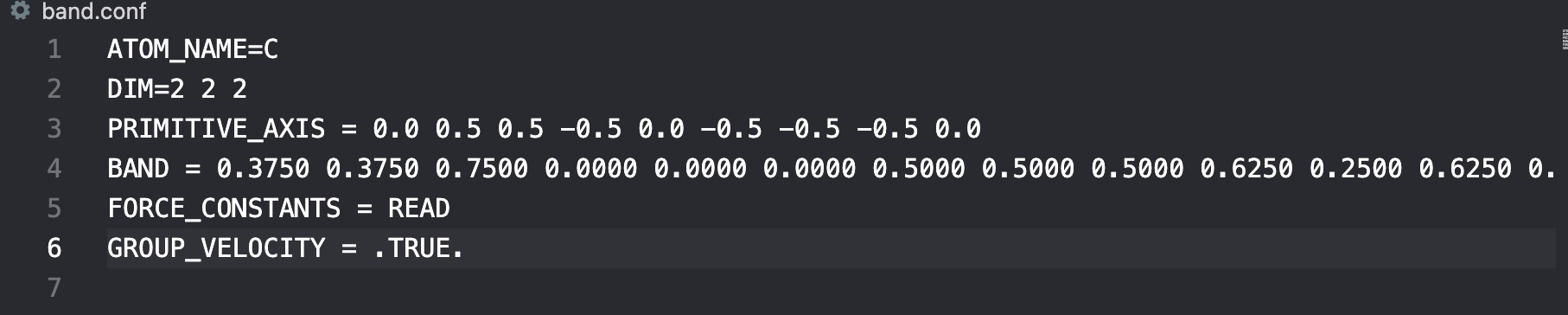
主程序在scripts-3.11中——AICON，依赖于mdcthermalc中的模块与函数，通过Phonopy计算一个在平衡体积，一个在稍小的体积，另一个在稍大的体积，格鲁尼森参数，并获取有一个名为POSCAR（VASP格式单元格）的结构文件，最后通过 AICON设置温度范围和取样频率列出温度，热导率值，比热比，和不同散射过程（正常、Umklapp和同位素缺陷）的分支声子弛豫时间

1. 程序的实验文档  
   实验过程：（以钻石为例）

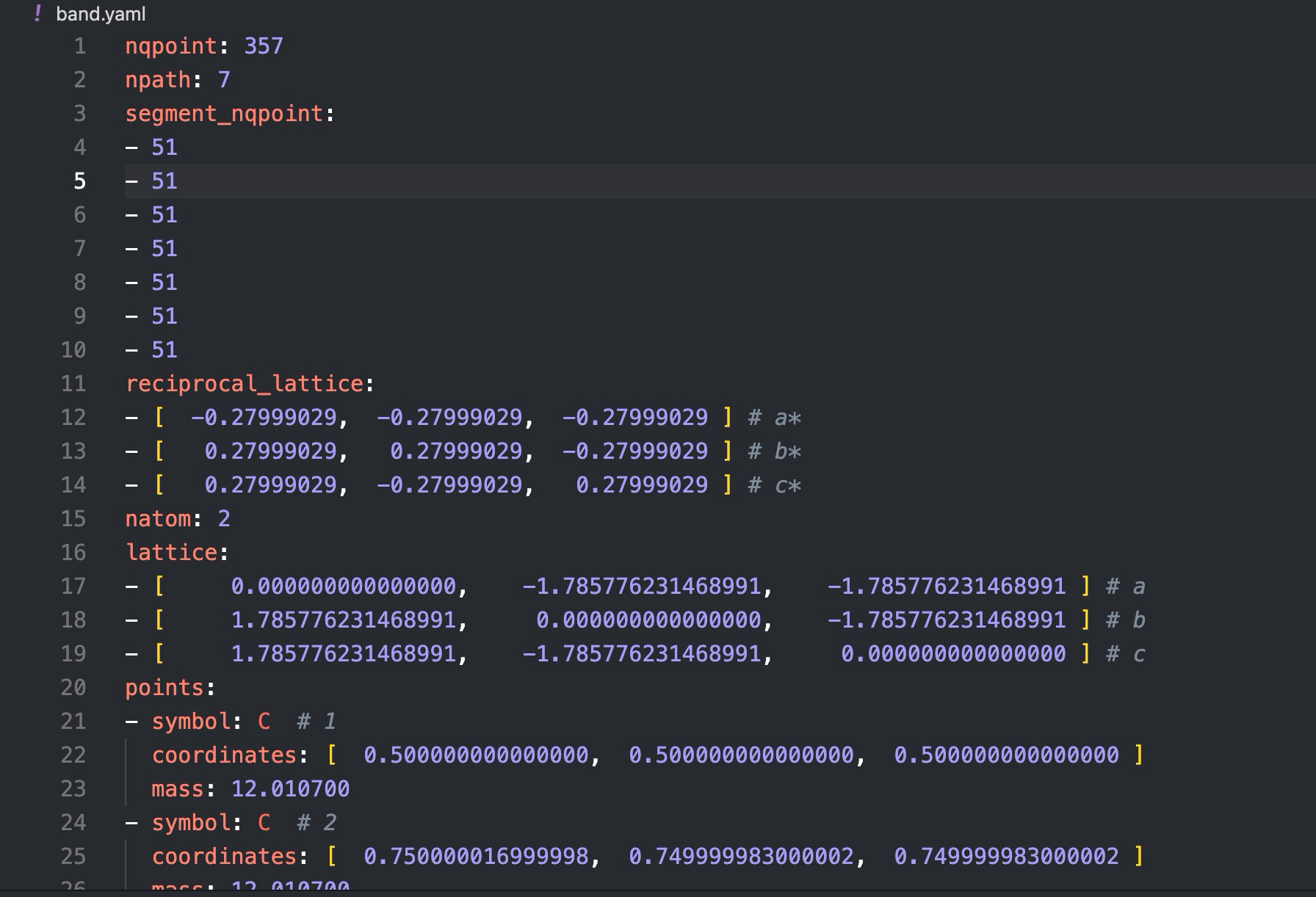
通过Phonopy获取一个名为POSCAR（VASP格式单元格）的结构文件，将其放在与 AICON 同目录下，运行python AICON -p，得到信息：



再设置 Phonopy 输入文件（如波段）



运行命令phonopy -c POSCAR-unitcell band.conf，将会得到band.yaml文件：



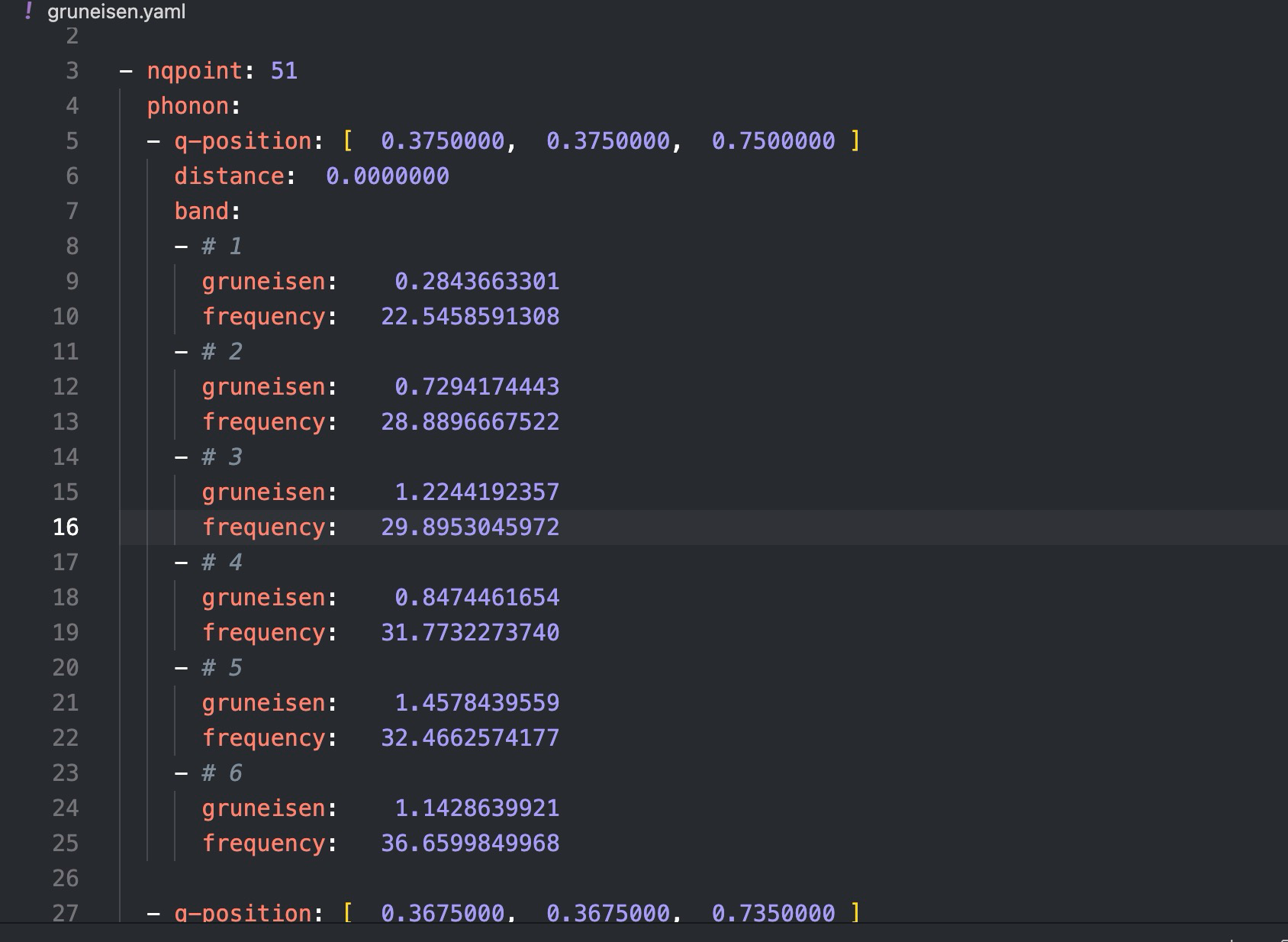
然后运行phonopy-gruneisen orig plus minus --dim="2 2 2" --pa="0 1/2 1/2 -1/2 0 -1/2 -1/2 -1/2

0" --band="0.3750 0.3750 0.7500 0.0000 0.0000 0.0000 0.5000 0.5000 0.5000

0.6250 0.2500 0.6250 0.0000 0.0000 0.0000 0.5000 0.2500 0.7500 0.5000

0.0000 0.5000 0.0000 0.0000 0.0000" --readfc -c POSCAR-unitcell

得到gruneisen.yaml文件：



这里已经获得了AICON计算热导率所需的所有输入文件，将其放在同意目录下，

运行命令：

AICON -t 200:50:1000

注“-t200：50：1000”表示温度范围为200~1000K，每50K取样一次。您可以指定要通过类似语法计算的任何温度或温度范围。

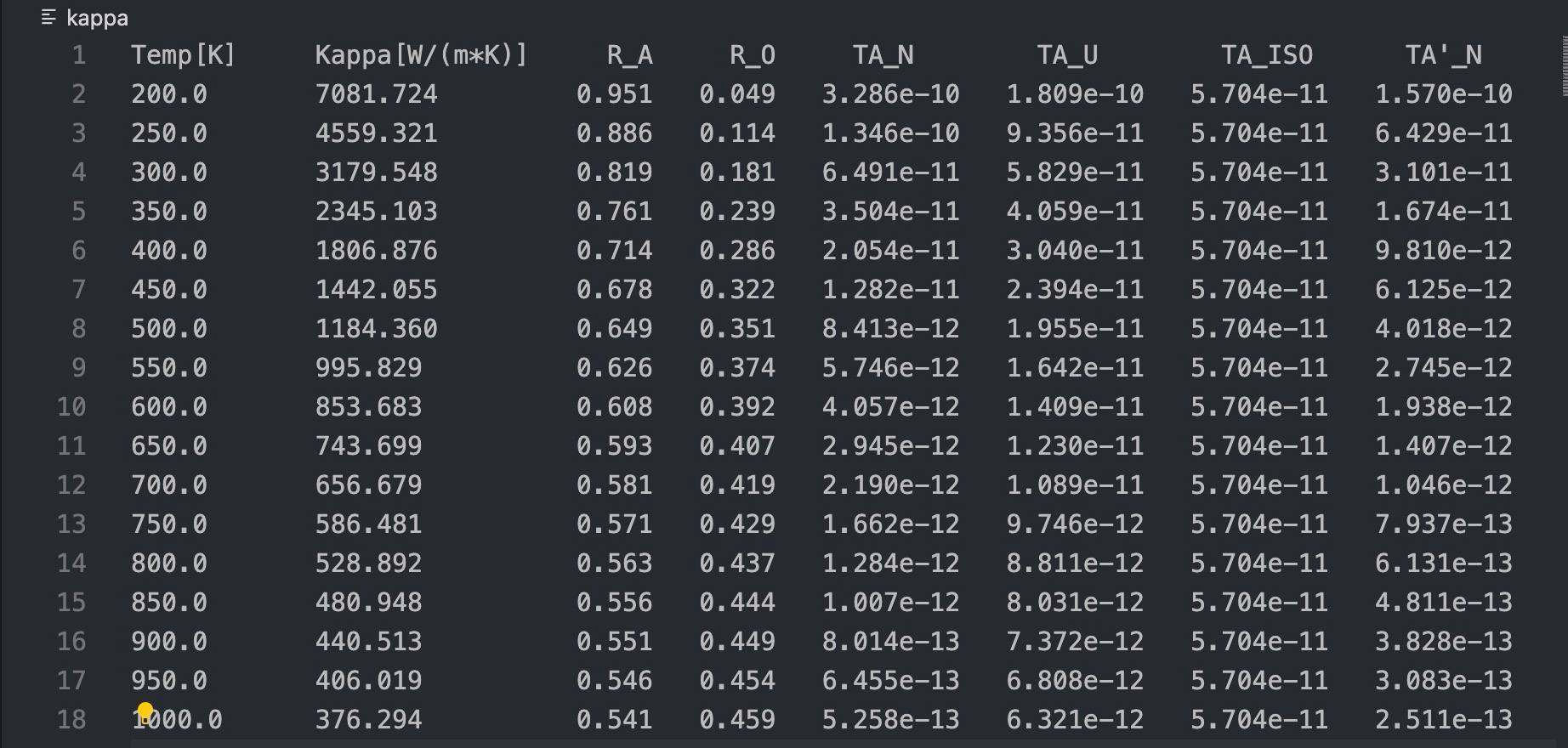
注意！由于所使用的方法，不能保证在极低的温度（<100K）下的导热

系数的结果。根据测试，超过200K的结果应该是可靠的。

结果被写入到名为“kappa”的文件中，并且这个文件非常简单。第一列为温度

，第二列为热导率值，第三和第四列为本文[1]定义的比热比，其余列为不同散

射过程（正常、Umklapp和同位素缺陷）的分支声子弛豫时间。



4、程序代码  
import sys

sys.path.append('/Applications/Python/AICON/build/lib.macosx-10.9-universal2-cpython-311')

import os

import numpy as np

from mdcthermalc.kappa import Kappa

from optparse import OptionParser

from mdcthermalc.get\_highsympath import get\_highsympath

#we need first get the structure file name and path. Then we need analysis which mode is executed

#-p means get the high symmetry path and transform coordinate. -t means calculate kappa

def optional\_arg(arg\_default):

def func(option, opt\_str, value, parser):

if parser.rargs and not parser.rargs[0].startswith('-'):

val = parser.rargs[0]

parser.rargs.pop(0)

else:

val = arg\_default

setattr(parser.values, option.dest, val)

return func

usage = "usage: %prog OPTIONS"

parser = OptionParser(usage=usage)

parser.add\_option("-p", "--highsym\_path", dest="spgpath", action="store\_true",

help="Obtain high-symmetry path", metavar="HPATH")

parser.add\_option("-t", "--temperature", dest="temp", action='callback', callback=optional\_arg('300'),

help="Specific temperature range to be calculated, if no value, the default is 300K", metavar="TEMP")

parser.add\_option("-v", "--version", dest="version", action="store\_true",

help="Obtain current version of the software", metavar="VERS")

(options, args) = parser.parse\_args()

if not options.spgpath and not options.temp and not options.version:

parser.error('Please specify -p, -t or -v')

if options.version:

Version = '2.0.1'

print('\n mDCThermalC %s\n' % (Version))

sys.exit(0)

if options.spgpath:

#1 get current directory and obtain POSCAR name and temperature

file\_path = os.getcwd()

if os.path.exists(file\_path + '/POSCAR'):

#2 analysis the structure file

filename = file\_path + '/POSCAR'

get\_highsympath(filename)

else:

print('No structure file found in current directory')

sys.exit(0)

if options.temp:

#1 get current directory and obtain POSCAR name

file\_path = os.getcwd()

sTemp = options.temp

sTemp = sTemp.strip()

sTemp\_list = np.double(sTemp.split(":"))

if len(sTemp\_list) == 1:

Temp = np.array(sTemp\_list)

if len(sTemp\_list) == 3:

Temp = np.arange(sTemp\_list[0],sTemp\_list[2]+sTemp\_list[1],sTemp\_list[1])

#2 calculate the thermal conductivity

if os.path.exists(file\_path + '/POSCAR') and os.path.exists(file\_path + '/gruneisen.yaml') and os.path.exists(file\_path + '/band.yaml'):

Kappa(file\_path,Temp)

else:

print('Not enough input files, please check if POSCAR, gruneisen.yaml and band.yaml are all exist')

sys.exit(0)