Physics

发布 1.00

Yong He

2021 年 11 月 11 日

Contents

CHAPTER 1

Overview

This Blog aims at providing some useful installation tips and first-principles calculations details.

About Me
Yong He (贺勇)
Ph. D Student.
Department of Physics, Peking University
Beijing 100871, China.
E-mail: hey@stu.pku.edu.cn
Scholar Page: http://scholar.pku.edu.cn/yong_phys

CHAPTER 2

Installation

2.1 libbeef

Compile libbeef using Intel Parallel Studio Xe 2020_update4

2.1.1 Download and Install

Download

1. Download version master[1] of libbeef.

```
git clone https://github.com/vossjo/libbeef.git
```

Compile

1. Create configure file compile.sh

```
./configure CC=icc \
   --prefix=/opt/software/libbeef/build \
```

2. configure libbeef

```
bash compile.sh
```

3. compile and test libbeef

[1] https://github.com/vossjo/libbeef

```
make && make check && make install
```

2.2 libxc

Compile libxc-4.3.4 using Intel Parallel Studio Xe 2020_update4

2.2.1 Download and Install

Download

1. Download version 4.3.4[2] of libxc.

```
wget http://www.tddft.org/programs/libxc/down.php?file=4.3.4/libxc-4.3.4.tar.gz
```

Compile

1. Create configure file compile.sh

```
./configure --prefix=/opt/software/libxc-4.3.4/bin \
    AR=ar \
    FC=ifort \
    F77=ifort \
    F90=ifort \
    CC=icc
```

2. configure libxc

```
bash compile.sh
```

3. compile and test yambo

```
make && make check && make Install
```

2.3 hdf5

Compile hdf5-1.12.0 using Intel Parallel Studio Xe 2020_update4

[2] http://www.tddft.org/programs/libxc/down.php?file=4.3.4/libxc-4.3.4.tar.gz

2.3.1 Download and Install

Download

1. Download version 4.3.4[3] of libxc.

```
wget http://www.tddft.org/programs/libxc/down.php?file=4.3.4/libxc-4.3.4.tar.gz
```

Compile

1. Create configure file compile.sh

```
./configure --prefix=/opt/software/libxc-4.3.4/bin \
    AR=ar \
    FC=ifort \
    F77=ifort \
    F90=ifort \
    CC=icc
```

2. configure libxc

```
bash compile.sh
```

3. compile and test yambo

```
make && make check && make Install
```

2.4 Quantum-ESPRESSO

Compile Quantum ESPRESSO-6.7, including EPW and thermo_pw codes, using Intel Parallel Studio Xe 2020_update4 with libbeef, libxc-4.3.4 and HDF5-1.12.0 packsges

2.4.1 Download and Install

Download

1. Download version 6.7[4] of Quantum-ESPRESSO.

```
wget https://github.com/QEF/q-e/releases/download/qe-6.7.0/qe-6.7-ReleasePack.tgz
```

[3] http://www.tddft.org/programs/libxc/down.php?file=4.3.4/libxc-4.3.4.tar.gz
[4] https://github.com/QEF/q-e/releases/download/qe-6.7.0/qe-6.7-ReleasePack.tgz

小技巧: One should also download the thermo_pw version 1.4.1[5] and Wannier90 version 3.1.0[6] codes

Compile

小技巧: Before compile Quantum-ESPRESSO package, one should compile the libbeef, libxc_4.3.4 and hdf5_1.12.0

preparation

- Unpack qe-6.7-ReleasePack

```
tar zxvf qe-6.7-ReleasePack.tgz && cd qe-6.7/test-suite
```

- Edit the check_pseudo.sh file as follows

```
#!/bin/bash
#
# Copyright (C) 2001-2016 Quantum ESPRESSO group
#
# This program is free software; you can redistribute it and/or
# modify it under the terms of the GNU General Public License
# as published by the Free Software Foundation; either version 2
# of the License. See the file 'License' in the root directory
# of the present distribution.

if test "`which curl`" = "" ; then
   if test "`which wget`" = "" ; then
      echo "### wget or curl not found: will not be able to download missing PP ###"
   else
      DOWNLOADER="wget -O"
      echo "wget found"
   fi
else
   DOWNLOADER="curl -o"
   echo "curl found"
fi

inputs=`find $1* -type f -name "*.in" -not -name "test.*" -not -name "benchmark.*"`
pp_files=`for x in ${inputs}; do grep UPF ${x} | awk '{print $3}'; done`
```

(下页继续)

[5] http://people.sissa.it/~dalcorso/thermo_pw/thermo_pw.1.4.1.tar.gz
[6] https://github.com/wannier-developers/wannier90/archive/v3.1.0.tar.gz

(续上页)

```
for pp_file in ${pp_files} ; do
if ! test -f ${ESPRESSO_PSEUDO}/${pp_file} ; then
    echo -n "Downloading ${pp_file} to ${ESPRESSO_PSEUDO} ... "
    ${DOWNLOADER} ${ESPRESSO_PSEUDO}/${pp_file} ${NETWORK_PSEUDO}/$
↪{pp_file} 2> /dev/null
    if test $? != 0 ; then
        echo "Download of" ${pp_file} "FAILED, do it manually -- Testing aborted!"
        exit -1
    else
        echo "done."
    fi
else
    echo "No need to download ${pp_file}."
fi
done
```

- Download the pseudopotentials

```
make pseudo
```

- Unpack the thermo_pw and move it into Quantum-ESPRESSO main direction

```
tar zxvf thermo_pw.1.4.1.tar.gz && mv thermo_pw qe-6.7
```

- Rename the Wannier90 code and move it into Quantum-ESPRESSO

```
mv wannier90-3.1.0.tar.gz v3.1.0 && mv v3.1.0 qe-6.7/archive/
```

Compile Quantum-ESPRESSO

1. Create configure file compile.sh

```
./configure MPIF90=mpiifort CC=mpiicc F90=ifort F77=mpiifort -enable-parallel \
        --with-libxc=yes \
        --with-libxc-prefix=/opt/software/libxc-4.3.4/bin \
        --with-libxc-include=/opt/software/libxc-4.3.4/bin/include \
        --with-scalapack=intel \
        --with-hdf5=yes \
        --with-hdf5-libs=/opt/software/hdf5-1.12.0_intelmpi/hdf5-1.12.0/hdf5/lib \
        --with-hdf5-include=/opt/software/hdf5-1.12.0_intelmpi/hdf5-1.12.0/hdf5/include \
        --with-libbeef=yes \
        --with-libbeef-prefix=/opt/software/libbeef/build/lib
```

2. configure Quantum-ESPRESSO

```
bash compile.sh
```

3. Edit the make.inc file

```
43 DFLAGS         =  -D__DFTI -D__LIBXC -D__MPI -D__SCALAPACK -D__
↪NOBEEF
109 FFLAGS         = -O3 -assume byterecl -g -traceback -xhost
124 LD_LIBS        = -L/opt/software/libxc-4.3.4/bin/lib -lxcf03 -lxc
131 BLAS_LIBS      =   -L${MKLROOT}/lib/intel64 -lmkl_scalapack_lp64 -lmkl_intel_
↪lp64 -lmkl_sequential -lmkl_core -lmkl_blacs_intelmpi_lp64 -lpthread -lm -ldl
137 LAPACK_LIBS    = -L${MKLROOT}/lib/intel64 -lmkl_scalapack_lp64 -lmkl_intel_
↪lp64 -lmkl_sequential -lmkl_core -lmkl_blacs_intelmpi_lp64 -lpthread -lm -ldl
140 SCALAPACK_LIBS = -L${MKLROOT}/lib/intel64 -lmkl_scalapack_lp64 -lmkl_intel_
↪lp64 -lmkl_sequential -lmkl_core -lmkl_blacs_intelmpi_lp64 -lpthread -lm -ldl
145 FFT_LIBS        = -L${MKLROOT}/lib/intel64 -lmkl_scalapack_lp64 -lmkl_intel_
↪lp64 -lmkl_sequential -lmkl_core -lmkl_blacs_intelmpi_lp64 -lpthread -lm -ldl
148 HDF5_LIBS = -L/opt/software/hdf5-1.12.0_intelmpi/hdf5-1.12.0/hdf5/lib -lhdf5
155 MPI_LIBS        = -L/opt/intel/impi/2019.9.304/intel64/lib -lmpi
163 BEEF_LIBS      = /opt/software/libbeef/build/lib/libbeef.a
164 BEEF_LIBS_SWITCH = internal
185 LIBXC_LIBS     = -L/opt/software/libxc-4.3.4/bin/lib -lxcf03 -lxc
```

4. compile Quantum-ESPRESSO

```
pp=4
make -j $pp pw && make -j $pp ph && make -j $pp hp && make -j $pp pwcond && make -
↪j $pp neb \
   && make -j $pp pp && make -j $pp cp && make -j $pp tddfpt && make -j $pp gwl \
   && make -j $pp ld1 && make -j $pp xspectra && make -j $pp couple && make epw

cd thermo_pw && make join_qe && cd ../ && make thermo_pw
```

5. test Quantum-ESPRESSO

```
cd test-suite && make run-tests-parallel
```

2.5 ABINIT

Compile abinit-9.4.2 using Intel Parallel Studio Xe 2020_update4

2.5.1 Download and Install

Download

1. Download version 9.4.2[7] of abinit.

```
wget https://www.abinit.org/sites/default/files/packages/abinit-9.4.2.tar.gz
```

[7] https://www.abinit.org/sites/default/files/packages/abinit-9.4.2.tar.gz

小技巧: One should also download the recommended ABINIT Fallbacks

1. Download Fallbacks[8] of abinit-9.4.2

Compile

Before compile abinit package, one should create a installation direction and copy the abinit and wannier90 to this direction. Next, one should build a tarballs by following command.

```
mkdir -p ~/.abinit/tarballs
```

Move the packages of above download (exclude wannier90) to tarballs.

Compile wannier90

1. Unpack and configure wannier90. This can be done by copy/pasting the following lines:

```
tar zxvf wannier90-2.0.1.1.tar.gz && \
cd wannier90-2.0.1.1 && \
./configure FC=mpiifort CC=mpiicc \
    prefix=/opt/software/wannier90-2.0.1.1/build/ && \
make && \
make install
```

Compile abinit

1. Create configure file compile.sh

```
./configure FC=mpiifort CC=mpiicc \
    --prefix=/opt/software/abinit-9.4.2/build \
    --enable-mpi-io=yes \
    --with-mpi=/opt/intel/impi/2019.9.304/intel64 \
    --enable-mpi-inplace=yes \
    --enable-openmp=yes \
    --enable-bse-unpacked=yes \
    --enable-gw-dpc=yes \
    --with-wannier90=/opt/software/wannier90-2.0.1.1/build
```

2. configure abinit

[8] https://www.abinit.org/fallbacks

```
bash compile.sh
```

3. compile fallbacks by following commands

```
cd fallbacks && bash build-abinit-fallbacks.sh
```

4. edit the compile.sh

```
./configure FC=mpiifort CC=mpiicc \
--prefix=/opt/software/abinit-9.4.2/build \
--enable-mpi-io=yes \
--with-mpi=/opt/intel/impi/2019.9.304/intel64 \
--enable-mpi-inplace=yes \
--enable-openmp=yes \
--enable-bse-unpacked=yes \
--enable-gw-dpc=yes \
--with-wannier90=/opt/software/wannier90-2.0.1.1/build \
with_libxc=/opt/software/abinit-9.4.2/fallbacks/install_fb/intel/19.1/libxc/4.3.4 \
with_hdf5=/opt/software/abinit-9.4.2/fallbacks/install_fb/intel/19.1/hdf5/1.10.6 \
with_netcdf=/opt/software/abinit-9.4.2/fallbacks/install_fb/intel/19.1/netcdf4/4.6.3 \
with_netcdf_fortran=/opt/software/abinit-9.4.2/fallbacks/install_fb/intel/19.1/netcdf4_
↪fortran/4.5.2 \
with_xmlf90=/opt/software/abinit-9.4.2/fallbacks/install_fb/intel/19.1/xmlf90/1.5.3.1 \
with_libpsml=/opt/software/abinit-9.4.2/fallbacks/install_fb/intel/19.1/libpsml/1.1.7
```

5. configure abinit

```
bash compile.sh
```

6. compile abinit

```
make -j<n> && make install
```

7. test abinit

```
./runtests.py v1 -j4
```

The test results are as follows.

```
Suite   failed  passed  succeeded  skipped  disabled  run_etime  tot_etime
v1           0       1         73        0         0    5597.67    5604.19

Completed in 719.19 [s]. Average time for test=75.64 [s], stdev=66.36 [s]
Summary: failed=0, succeeded=73, passed=1, skipped=0, disabled=0

Execution completed.
Results in HTML format are available in Test_suite/suite_report.html
```

2.6 Yambo

Compile Yambo-5.0.2 using Intel Parallel Studio Xe 2020_update4

2.6.1 Download and Install

Download

1. Download version 5.0.2[9] of Yambo.

```
wget https://github.com/yambo-code/yambo/archive/5.0.2.tar.gz
```

2. One should also download the dependent packages of Yambo by following commands

```
tar zxvf yambo-5.0.2.tar.gz && cd yambo-5.0.2/lib/archive && \
   make -f Makefile.loc
```

重要: The yambo-libraries-0.0.2.tar.gz package should also be download

- Download version 0.0.2[10]
- Unpack and move the files to yambo code

```
tar zxvf yambo-libraries-0.0.2.tar.gz && \
   mv yambo-libraries-0.0.2/* yambo-5.0.2/lib/yambo
```

Compile

- Before compile the yambo, one should compile the hdf5_1.12.0 code

1. Create configure file compile.sh

```
./configure FC="ifort" MPIFC="mpiifort" F77="ifort" MPIF77="mpiifort" CC="icc"␣
↪MPICC="mpiicc" CPP="gcc -E -P" FPP="gfortran -E -P -cpp"   \
--enable-mpi \
--enable-open-mp \
--enable-dp \
--with-blas-libs="-lmkl_intel_lp64 -lmkl_intel_thread -lmkl_core -liomp5 -lpthread -lm" \
--with-lapack-libs="-lmkl_intel_lp64 -lmkl_intel_thread -lmkl_core -liomp5 -lpthread -lm" \
--with-fft-includedir=/opt/intel/compilers_and_libraries_2020.4.304/linux/mkl/include/fftw␣
↪\
--with-fft-libs="-lmkl_intel_lp64 -lmkl_intel_thread -lmkl_core -liomp5 -lpthread -lm" \
```

(下页继续)

[9] https://github.com/yambo-code/yambo/archive/5.0.2.tar.gz
[10] https://codeload.github.com/yambo-code/yambo-libraries/tar.gz/0.0.2

(续上页)

```
--with-blacs-libs="-lmkl_scalapack_lp64 -lmkl_blacs_intelmpi_lp64 -lmkl_core -liomp5 -
↪lpthread -lm" \
--with-scalapack-libs="-lmkl_scalapack_lp64 -lmkl_blacs_intelmpi_lp64 -lmkl_core -
↪liomp5 -lpthread -lm" \
--with-hdf5-libs=/opt/software/HDF5_intelmpi/hdf5-1.12.0/hdf5/lib/libhdf5.a \
--with-hdf5-path=/opt/software/HDF5_intelmpi/hdf5-1.12.0/hdf5 \
--with-hdf5-libdir=/opt/software/HDF5_intelmpi/hdf5-1.12.0/hdf5/lib \
--with-hdf5-includedir=-I/opt/software/HDF5_intelmpi/hdf5-1.12.0/hdf5/include \
```

2. configure yambo

```
bash compile.sh
```

3. compile yambo

```
make core
```

2.7 BerkeleyGW

Compile BerkeleyGW-3.0.1 using Intel Parallel Studio Xe 2020_update4

2.7.1 Download and Install

Download

- Download version 3.0.1[11] of BerkeleyGW.

Compile

- Before compile BerkeleyGW package, one should compile hdf5-1.12.0 code Download version 1.12.0[12] of HDF5.

[11] https://berkeley.box.com/s/1dgnhiemo47lhxczrn6si71bwxoxor8

[12] https://www.hdfgroup.org/package/hdf5-1-12-0-tar-gz/?wpdmdl=14582&refresh=618bc42bc76531636549675

2.1 Compile HDF5

```
./configure CC=mpiicc FC=mpiifort CXX=mpiicpc \
   --enable-fortran --enable-parallel --enable-shared \
   --prefix=/your_path
make

make install
```

2.2 Compile BerkeleyGW

- Create arch.mk file, one can modifiy the ./config/frontera.tacc.utexas.edu_impi.mk file as follows.

```
# arch.mk for BerkeleyGW codes
#
# suitable for Frontera at TACC, Cascade Lake (CLX) architecture
# Uses intel compilers + impi
#
# BERKELEYGW DOES NOT WORK WITH INTEL/18 DUE TO A COMPILER BUG!
# ------------------------------------------------------------
#
# You'll need to run:
# module load arpack impi intel/19.0.5 phdf5
#
# Use 'make -j 8' for parallel build on Frontera
#
# Zhenglu Li
# July 2020, Berkeley

COMPFLAG  = -DINTEL
PARAFLAG  = -DMPI -DOMP
MATHFLAG  = -DUSESCALAPACK -DUNPACKED -DUSEFFTW3 -DHDF5
# Only uncomment DEBUGFLAG if you need to develop/debug BerkeleyGW.
# The output will be much more verbose, and the code will slow down by ~20%.
#DEBUGFLAG = -DDEBUG

FCPP    = cpp -C -nostdinc
F90free = mpiifort -free -qopenmp -ip -no-ipo
LINK    = mpiifort -qopenmp -ip -no-ipo
# We need the -fp-model procise to pass the testsuite.
FOPTS   = -O3 -fp-model source
FNOOPTS = -O2 -fp-model source -no-ip
#FOPTS   = -g -O0 -check all -Warn all -traceback
#FNOOPTS = $(FOPTS)
MOD_OPT = -module
INCFLAG = -I
```

(下页继续)

(续上页)

```
C_PARAFLAG = -DPARA -DMPICH_IGNORE_CXX_SEEK
CC_COMP = mpiicpc
C_COMP  = mpiicc
C_LINK  = mpiicpc
C_OPTS  = -O3 -ip -no-ipo -qopenmp
C_DEBUGFLAG =

REMOVE  = /bin/rm -f

# Math Libraries
#
MKLPATH      = $(MKLROOT)/lib/intel64

FFTWLIB      =      -Wl,--start-group \
          $(MKLPATH)/libmkl_intel_lp64.a \
          $(MKLPATH)/libmkl_intel_thread.a \
          $(MKLPATH)/libmkl_core.a \
          -Wl,--end-group -liomp5 -lpthread -lm -ldl
FFTWINCLUDE  = $(MKLROOT)/include/fftw


LAPACKLIB    = -Wl,--start-group \
          $(MKLPATH)/libmkl_intel_lp64.a \
          $(MKLPATH)/libmkl_intel_thread.a \
          $(MKLPATH)/libmkl_core.a \
          $(MKLPATH)/libmkl_blacs_intelmpi_lp64.a \
          -Wl,--end-group -liomp5 -lpthread -lm -ldl
SCALAPACKLIB = $(MKLPATH)/libmkl_scalapack_lp64.a

HDF5PATH     =  /opt/software/BerkeleyGW-3.0.1/hdf5/lib
HDF5LIB      =      $(HDF5PATH)/libhdf5hl_fortran.a \
          $(HDF5PATH)/libhdf5_hl.a \
          $(HDF5PATH)/libhdf5_fortran.a \
          $(HDF5PATH)/libhdf5.a \
          -lz
HDF5INCLUDE  = $(HDF5PATH)/../include

TESTSCRIPT =
```

- Compile

```
make -j N all-flavors (N is the number of cores)
```

CHAPTER 3

Tutorials

3.1 VASP

3.1.1 前言

载流子迁移率通常指半导体内部电子和空穴整体的运动快慢情况，是衡量半导体器件性能的重要物理量。2004 年，石墨烯的成功剥离引起了研究人员对于二维材料性质探索的浓厚兴趣。石墨烯、黑磷等二维材料展现出的高载流子迁移率是其中的一个重要研究课题，科研人员在理论计算方面已经做了大量的工作。由于电子在运动过程中不仅受到外电场力的作用，还会不断的与晶格、杂质、缺陷等发生无规则的碰撞，大大增加了理论计算的难度。

目前计算载流子迁移率比较常用的理论是形变势理论和玻尔兹曼输运理论，前者没有考虑电子和声子（晶格振动）以及电子与电子之间的相互作用等因素，计算结果存在一定的误差，但笔者的计算结果与实验值在数量级上是吻合的；玻尔兹曼输运理论的一种计算考虑了电子-声子的相互作用，基于第一性原理计算和最大局域化 wannier 函数插值方法，借助于 Quantum-ESPRESSO[13] 和 EPW[14] 软件可以完成载流子迁移率计算。缺点是计算量太大，一般的课题组很难承受起高昂的计算费用，另外 EPW 软件对于二维材料的计算存在部分问题，在其官方论坛也有讨论，计算过程在后续文章中会提到。

本文以形变势理论方法为基础，详细介绍了二维 InSe 的电子和空穴的有效质量与载流子迁移率的计算方法。

[13] http://www.quantum-espresso.org/
[14] https://epw-code.org/

3.1.2 理论基础

基于 Bardeen and Shockley[15] 提出的形变势理论，二维材料载流子迁移率可以根据下式计算：

$$\mu_{\mathrm{2D}} = \frac{e\hbar^3 C_{2D}}{k_B T m^* m_d E_1^2}$$

其中，m^* 是传输方向上的有效质量，T 是温度，k_B 是玻尔兹曼常数 E_1 表示沿着传输方向上位于价带顶（VBM）的空穴或聚于导带底（CBM）的电子的形变势常数，由公式 $E_1 = \Delta E/(\Delta l/l_o)$ 确定，ΔE 为在压缩或拉伸应变下 CBM 或 VBM 的能量变化，l_0 是传输方向上的晶格常数，Δl 是 l_0 的变形量。m_d 是载流子的平均有效质量，由公式 $m_d = \sqrt{m_x^* m_y^*}$ 定义。C_{2D} 是均匀变形晶体的弹性模量，对于 2D 材料，弹性模量可以通过公式 $C_{2D} = 2[\partial^2 E/\partial(\Delta l/l_0)^2]/S_0$ 来计算，其中 E 是总能量，S_0 是优化后的面积。

下面对公式中的单位（量纲）做一个简单换算，具体如下：

$m_d = \sqrt{m_x^* m_y^*}$ (Kg)
$E_1 = \Delta E/(\Delta l/l_o)$ (J) 在 VASP 中 ΔE 的单位是 eV，1 eV = 1.6×10^{-19} J
$C_{2D} = 2[\partial^2 E/\partial(\Delta l/l_0)^2]/S_0$ (J/m ^{2}) 其中 E 是总能（eV），S_0 表示面积（^{2} ）
e：1.6×10^{-19} C
h：6.626×10^{-34} $J \cdot s$
k_B：1.38×10^{-23} J/K
m^*：Kg

换算过程：

$$C \cdot J^3 \cdot s^3 \cdot \frac{J}{m^2} = \frac{C \cdot J \cdot s^3}{m^2 \cdot Kg^2} = \frac{C \cdot J}{\frac{m^2 \cdot Kg^2}{s^3} \cdot \frac{s}{s} \cdot \frac{m^2}{m^2}} = \frac{C \cdot J}{\frac{s}{m^2}\frac{m^4 \cdot Kg^2}{s^4}} = \frac{m^2}{s} \cdot \frac{C}{J} = m^2 \cdot s^{-1} \cdot V^{-1}$$

$$= 10^4 \times cm^2 \cdot s^{-1} \cdot V^{-1}$$

其中：$1V = 1J/C$ ，$1J = 1Kg \cdot m^2/s^2$

3.1.3 计算与数据处理工具

VASP.5.4.4 软件 (可以手动控制优化晶格方向)
OriginLab 软件
Excel
Materials Studio 软件
正格矢到倒格矢转化脚本，来源于 小木虫[16]

[15] https://doi.org/10.1103/PhysRev.80.72
[16] http://muchong.com/bbs/viewthread.php?tid=7149817&fpage=1

```
#! /usr/bin/python
# This program reads in base vectors from a given file, calculates reciprocal vectors
# then writes to outfile in different units
# LinuxUsage: crecip.py infile outfile
# Note:  the infile must be in the form below:
#   inunit  ang/bohr
#   _begin_vectors
#     46.300000000   0.000000000   0.000000000
#      0.000000000  40.500000000   0.000000000
#      0.000000000   0.000000000  10.000000000
#   _end_vectors
#
# Note: LATTICE VECTORS ARE SPECIFIED IN ROWS !
def GetInUnit( incontent ):
    inunit = ""
    for line in incontent:
        if line.find("inunit") == 0:
            inunit = line.split()[1]
            break
    return inunit
def GetVectors( incontent ):
    indstart = 0
    indend = 0
    for s in incontent:
        if s.find("_begin_vectors") != -1:
            indstart = incontent.index(s)
        else:
            if s.find("_end_vectors") != -1:
                indend = incontent.index(s)
    result = []
    for i in range( indstart + 1, indend ):
        line = incontent[i].split()
        result.append( [ float(line[0]), float(line[1]), float(line[2]) ] )
    return result
def Ang2Bohr( LattVecAng ):
    LattVecBohr = LattVecAng
    for i in range(0,3):
        for j in range(0,3):
            LattVecBohr[i][j] = LattVecAng[i][j] *  1.8897261246
    return LattVecBohr
def DotProduct( v1, v2 ):
    dotproduct = 0.0
    for i in range(0,3):
        dotproduct = dotproduct + v1[i] * v2[i]
    return dotproduct
def CrossProduct( v1, v2 ):
    # v3 = v1 WILL LEAD TO WRONG RESULT
    v3 = []
    v3.append( v1[1] * v2[2] - v1[2] * v2[1] )
    v3.append( v1[2] * v2[0] - v1[0] * v2[2] )
```

(下页继续)

(续上页)

```
    v3.append( v1[0] * v2[1] - v1[1] * v2[0] )
    return v3
def CalcRecVectors( lattvec ):
    pi = 3.141592653589793
    a1 = lattvec[0]
    a2 = lattvec[1]
    a3 = lattvec[2]
    b1 = CrossProduct( a2, a3 )
    b2 = CrossProduct( a3, a1 )
    b3 = CrossProduct( a1, a2 )
    volume = DotProduct( a1, CrossProduct( a2, a3 ) )
    RecVec = [ b1, b2, b3 ]
    # it follows the definition for b_j: a_i * b_j = 2pi * delta(i,j)
    for i in range(0,3):
        for j in range(0,3):
            RecVec[i][j] = RecVec[i][j] * 2 * pi / volume
    return RecVec
def main(argv = None):
    argv = sys.argv
    infilename  = argv[1]
    outfilename = argv[2]
    pi = 3.141592653589793
    bohr2ang = 0.5291772109253
    ang2bohr = 1.889726124546
    infile  = open(infilename,"r")
    incontent = infile.readlines()
    infile.close()
    inunit = GetInUnit( incontent )
    LattVectors = GetVectors( incontent )
    # convert units from ang to bohr
    if inunit == "ang":
        LattVectors = Ang2Bohr( LattVectors )
    # calculate reciprocal vectors in 1/bohr
    RecVectors = CalcRecVectors( LattVectors )
    # open outfile for output
    ofile = open(outfilename,"w")
    # output lattice vectors in bohr
    ofile.write("lattice vectors in bohr:\n")
    for vi in LattVectors:
        ofile.write("%14.9f%14.9f%14.9f\n" % (vi[0], vi[1], vi[2]))
    ofile.write("\n")
    # output lattice vectors in ang
    convfac = bohr2ang
    ofile.write("lattice vectors in ang:\n")
    for vi in LattVectors:
        ofile.write("%14.9f%14.9f%14.9f\n" % (vi[0]*convfac, vi[1]*convfac, vi[2]*convfac))
    ofile.write("\n")
    # output reciprocal vectors in 1/bohr
    ofile.write("reciprocal vectors in 1/bohr:\n")
```

(下页继续)

(续上页)

```
    for vi in RecVectors:
        ofile.write("%14.9f%14.9f%14.9f\n" % (vi[0], vi[1], vi[2]))
    ofile.write("\n")
    # output reciprocal vectors in 1/ang
    convfac = ang2bohr
    ofile.write("reciprocal vectors in 1/ang:\n")
    for vi in RecVectors:
        ofile.write("%14.9f%14.9f%14.9f\n" % (vi[0]*convfac, vi[1]*convfac, vi[2]*convfac))
    ofile.write("\n")
    # output reciprocal vectors in 2pi/bohr
    convfac = 1.0/(2.0*pi)
    ofile.write("reciprocal vectors in 2pi/bohr:\n")
    for vi in RecVectors:
        ofile.write("%14.9f%14.9f%14.9f\n" % (vi[0]*convfac, vi[1]*convfac, vi[2]*convfac))
    ofile.write("\n")
    # output reciprocal vectors in 2pi/ang
    convfac = ang2bohr/(2.0*pi)
    ofile.write("reciprocal vectors in 2pi/ang:\n")
    for vi in RecVectors:
        ofile.write("%14.9f%14.9f%14.9f\n" % (vi[0]*convfac, vi[1]*convfac, vi[2]*convfac))
    # close
    ofile.close()
    return 0
if __name__ == "__main__":
    import sys
    sys.exit(main())
```

3.1.4 二维 InSe 有效质量计算过程

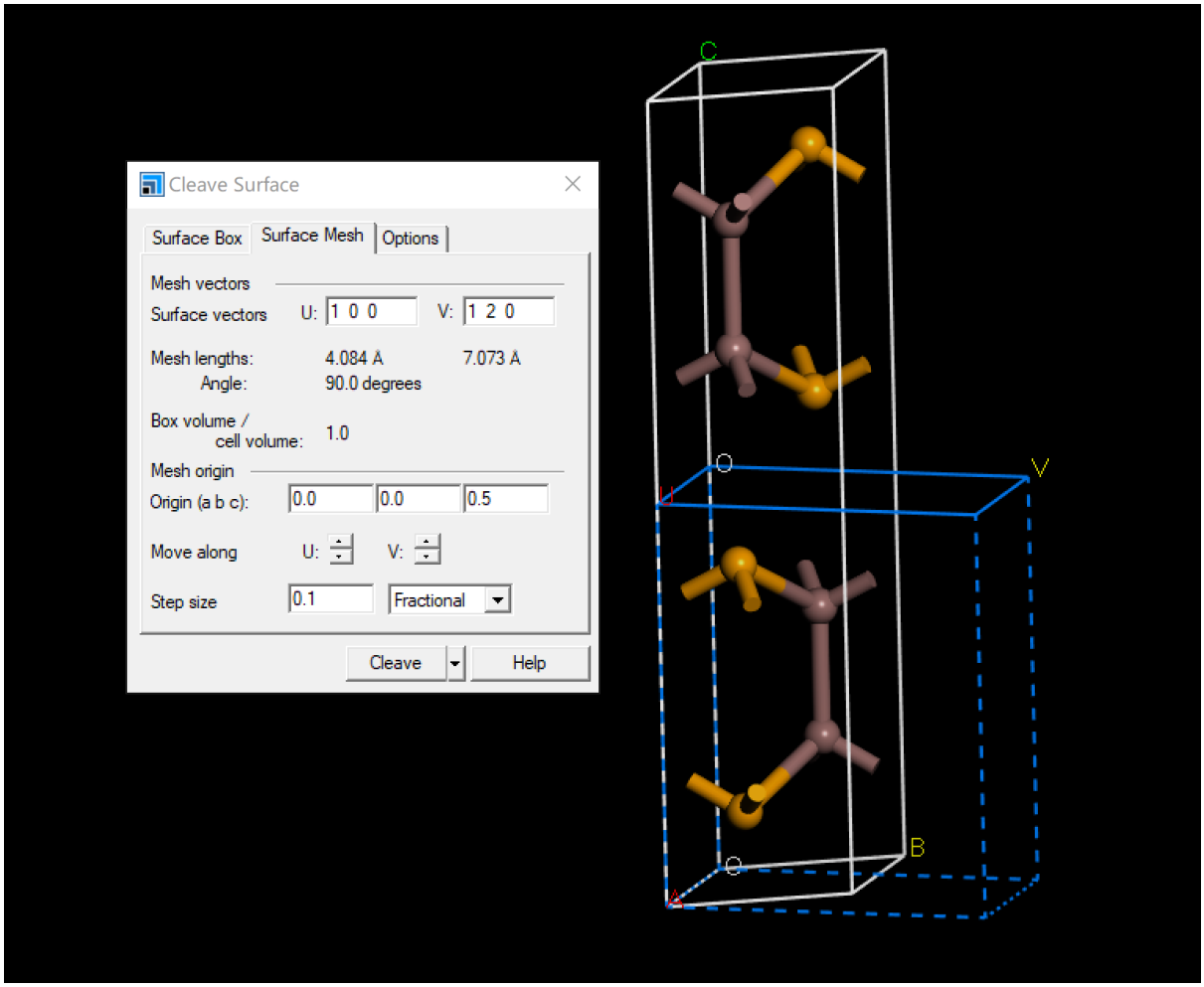
建模

由于计算过程中需要对二维 InSe 施加应变，但二维 InSe 原胞是六角结构，不容易施加应变。但是侯柱峰老师讲了对石墨烯原胞施加应变的方法，笔者认为虽然可行，但过于繁琐，故不采用此法。我们可以利用根号建模的方法讲六角结构 InSe 原胞变为方形结构的 InSe 超胞，然后施加应变可大大提高操作效率，但计算量的增加在可接受范围之内。下面给出关键的建模步骤，更多的根号建模部分可参考我的往期博客文章。

能带计算

结构优化

INCAR


Cleave Surface
Surface Box
Surface Mesh
Options
Mesh vectors
Surface vectors
U: 1 0 0
V: 1 2 0
Mesh lengths:
4.084 Å
7.073 Å
Angle:
90.0 degrees
Box volume / cell volume:
1.0
Mesh origin
Origin (a b c):
0.0
0.0
0.5
Move along
U:
V:
Step size
0.1
Fractional
Cleave
Help
C
O
V
U
B
O
A

```
SYSTEM = InSe
ISTART = 0
NWRITE = 2
PREC   = Accurate
ENCUT  = 500
GGA    = PE
NSW    = 200
ISIF   = 3
ISYM   = 2
IBRION = 2
NELM   = 80
EDIFF  = 1E-05
EDIFFG = -0.01
ALGO   = Normal
LDIAG  = .TRUE.
LREAL  = .FALSE.
ISMEAR = 0
SIGMA  = 0.05
ICHARG = 2
LWAVE  =  .FALSE.
LCHARG = .FALSE.
NPAR   = 4
```

KPOINTS

```
Monkhorst Pack
0
Gamma
 11   7   1
.0   .0   .0
```

POSCAR

```
  Se   In
  1.000
       4.083622259999999      -0.000000000000001       0.000000000000000
       0.000000000000000       7.073041233239241       0.000000000000000
       0.000000000000000       0.000000000000000      25.377516849029359
  Se   In
  4   4
Direct
  0.5000005000000000   0.1666665000000000   0.5271404971815050   !Se
  0.0000004999999997   0.6666665000000004   0.5271404971815050   !Se
```

(下页继续)

(续上页)

```
 0.5000005000000000  0.1666665000000000  0.3152396685456632  !Se
 0.0000004999999997  0.6666665000000004  0.3152396685456632  !Se
 0.4999995000000003  0.8333335000000002  0.4767849697227853  !In
-0.0000005000000000  0.3333335000000000  0.4767849697227853  !In
 0.4999995000000003  0.8333335000000002  0.3655951960043828  !In
-0.0000005000000000  0.3333335000000000  0.3655951960043828  !In
```

POTCAR

```
cat Se/POTCAR In_d/POTCAR > POTCAR
```

OPTCELL

```
100
010
000
```

静态自洽

INCAR

```
SYSTEM = InSe
ISTART = 0
NWRITE = 2
PREC   = Accurate
ENCUT  = 500
GGA    = PE
NSW    = 0
ISIF   = 2
ISYM   = 2
IBRION = -1
NELM   = 80
EDIFF  = 1E-05
EDIFFG = -0.01
ALGO   = Normal
LDIAG  = .TRUE.
LREAL  = .FALSE.
ISMEAR = 0
SIGMA  = 0.05
```

(下页继续)

(续上页)

```
ICHARG = 2
NPAR   = 4
```

KPOINTS

```
Monkhorst Pack
0
Gamma
 21   13   1
.0   .0   .0
```

POSCAR

```
cp CONTCAR scf/POSCAR
```

能带计算

3.2 Quantum-ESPRESSO

3.3 EPW

3.4 pymatgen

3.4.1 Basic Module

1. Lattice, Structure, Molecule Module

```
from pymatgen.core import Lattice, Structure, Molecule
# Read a POSCAR and write to a CIF.
structure = Structure.from_file("POSCAR")
structure.to(filename="CsCl.cif")

# Read an xyz file and write to a Gaussian Input file.
methane = Molecule.from_file("methane.xyz")
methane.to(filename="methane.gjf")
```

2. VASP and cif Module

```
from pymatgen.io.cif import CifParser
# read structure from cif file
parser = CifParser("mycif.cif")
structure = parser.get_structures()[0]
# read POSCAR file and write it to cif structure
from pymatgen.io.vasp import Poscar
from pymatgen.io.cif import CifWriter
p = Poscar.from_file('POSCAR')
w = CifWriter(p.structure)
w.write_file('mystructure.cif')
```

3.5 Gnuplot

3.5.1 希腊字母表

- 用法：{/symbol De} 结果：Δ

表 1: 希腊字母符号表

ALPHA-BET	SYM-BOL	ALPHA-BET	SYM-BOL	alpha-bet	sym-bol	alpha-bet	sym-bol
A	Alpha	N	Nu	a	alpha	n	nu
B	Beta	O	Omicron	b	beta	o	omi-cron

CHAPTER 4

Indices and tables

- genindex
- modindex
- search