



Physics  
发布 1.00

Yong He

2021 年 11 月 27 日



---

## Contents

---

1	Overview	1
2	Code list	3
2.1	pydefect . . . . .	3
2.1.1	pydefect 程序: 非金属体系点缺陷计算 . . . . .	3
2.2	Nonrad . . . . .	6
2.2.1	Nonrad 程序: 计算载流子非辐射捕获系数 . . . . .	6
2.3	PyXtal . . . . .	8
2.3.1	PyXtal 程序: 晶体结构生成和对称性分析 . . . . .	8
2.4	Irvsp . . . . .	12
2.4.1	Irvsp 程序: 计算 VASP 中电子态的不可约表示 . . . . .	12
2.5	phonopy . . . . .	12
2.5.1	Phonopy: 声子性质处理软件 . . . . .	12
3	Installation	13
3.1	libbeef . . . . .	13
3.1.1	Download and Install . . . . .	13
	Download . . . . .	13
	Compile . . . . .	13
3.2	libxc . . . . .	14
3.2.1	Download and Install . . . . .	14
	Download . . . . .	14
	Compile . . . . .	14
3.3	hdf5 . . . . .	14
3.3.1	Download and Install . . . . .	15
	Download . . . . .	15
	Compile . . . . .	15
3.4	Quantum-ESPRESSO . . . . .	15
3.4.1	Download and Install . . . . .	15
	Download . . . . .	15
	Compile . . . . .	16
3.5	ABINIT . . . . .	19
3.5.1	Download and Install . . . . .	19

	Download . . . . .	19
	Compile . . . . .	19
3.6	Yambo . . . . .	21
3.6.1	Download and Install . . . . .	21
	Download . . . . .	21
	Compile . . . . .	22
3.7	BerkeleyGW . . . . .	22
3.7.1	Download and Install . . . . .	22
	Download . . . . .	22
	Compile . . . . .	23
4	Tutorials . . . . .	27
4.1	VASP . . . . .	27
4.1.1	Mobility Calculation . . . . .	27
	理论基础 . . . . .	28
	计算与数据处理工具 . . . . .	28
	二维 InSe 有效质量计算过程 . . . . .	31
4.1.2	Formation Energy Calculation . . . . .	35
4.2	BerkeleyGW . . . . .	36
4.2.1	GW 计算过程 . . . . .	36
	Quantum ESPRESSO 计算 . . . . .	37
	BerkeleyGW 计算 . . . . .	55
	数据处理 . . . . .	60
4.3	Quantum-ESPRESSO . . . . .	60
4.4	EPW . . . . .	60
4.4.1	NSCF Kpoints . . . . .	60
4.5	pymatgen . . . . .	61
4.5.1	Basic Module . . . . .	61
4.6	Gnuplot . . . . .	62
4.6.1	希腊字母表 . . . . .	62
4.7	Atomic Simulation Environment (ASE) . . . . .	62
4.7.1	Install . . . . .	62
5	database . . . . .	63
5.1	无机材料数据库 . . . . .	63
5.1.1	三维材料数据库 . . . . .	63
6	Indices and tables . . . . .	65

# 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](mailto:hey@stu.pku.edu.cn)

Scholar Page: [http://scholar.pku.edu.cn/yong\\_phys](http://scholar.pku.edu.cn/yong_phys)



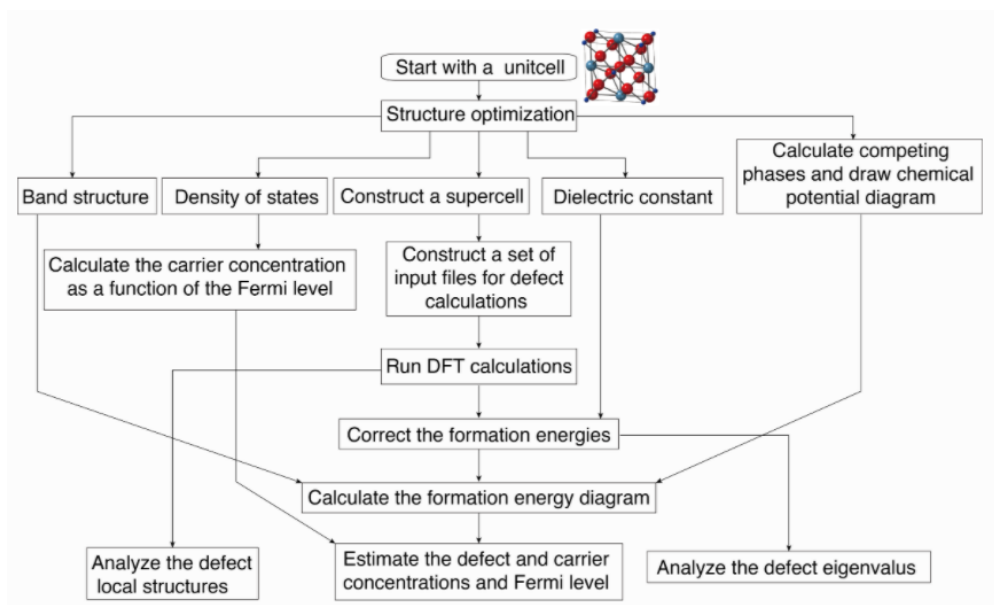
总结了常见的计算和数据处理软件，以备用时方便查找

## 2.1 pydefect

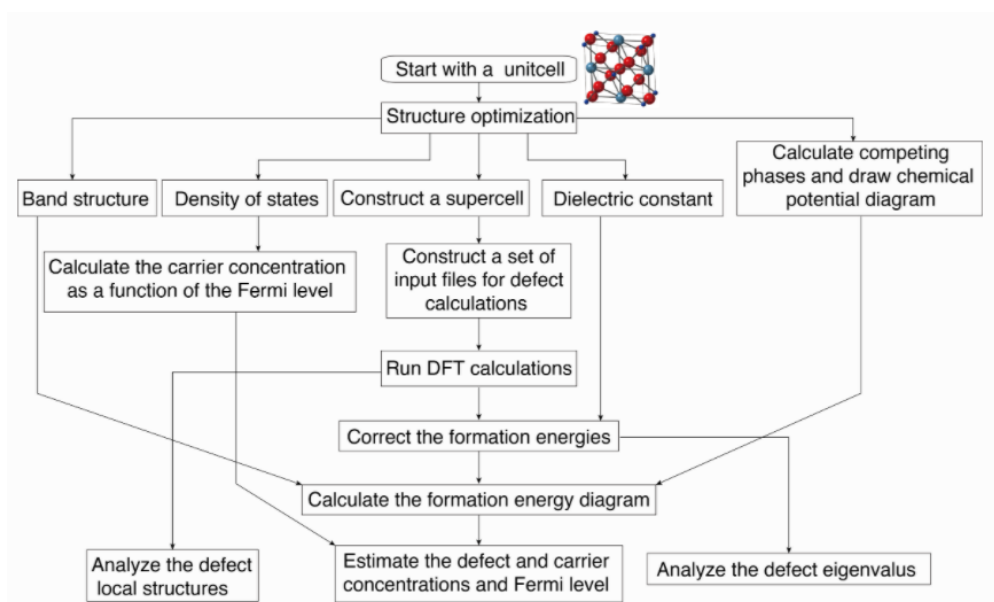
### 2.1.1 pydefect 程序: 非金属体系点缺陷计算

#### 介绍

半导体材料中的缺陷 (点缺陷、线缺陷、面缺陷等) 对于器件性能具有重要影响，如何构建这些缺陷模型，并且计算这些缺陷 (缺陷形成能等)，以及衡量缺陷对性能的影响至关重要。尤其是对于带电缺陷的处理较为复杂，涉及各种经验修正等等，如果没有合适的脚本借助工作量较大。pPydefect 是一个开源 dePython 库，用于基于 VASP 第一性原理计算的 非金属固体中的点缺陷处理。开发者是东京工业大学的 Yu Kumagai 教授课题组。该程序主要的逻辑框架如下所示：



- 思维框架图:



- pydefect 下载链接: [Github](#)<sup>1</sup>

注意: Pydefect 现在仅支持维也纳 ab-initio 模拟程序包 (VASP), 因此我们假设其输入和输出文件名 (例如 POSCAR, POTCAR, OUTCAR) 以及 VASP 中使用的计算技术 (例如周期性边界条件)

遵循 vasp 约定, 在 pydefect 中使用的单位是能量的 eV 和长度的埃  
仅假定非磁性主体材料

- pydefect 官方文档: [Document](#)<sup>2</sup>

<sup>1</sup> <https://github.com/kumagai-group/pydefect>

<sup>2</sup> <https://kumagai-group.github.io/pydefect/>

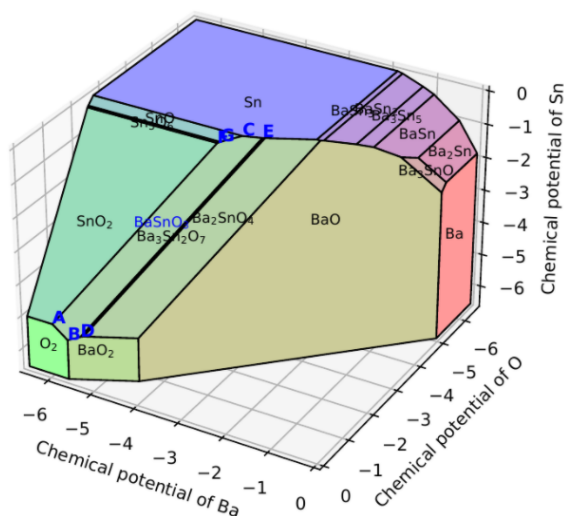


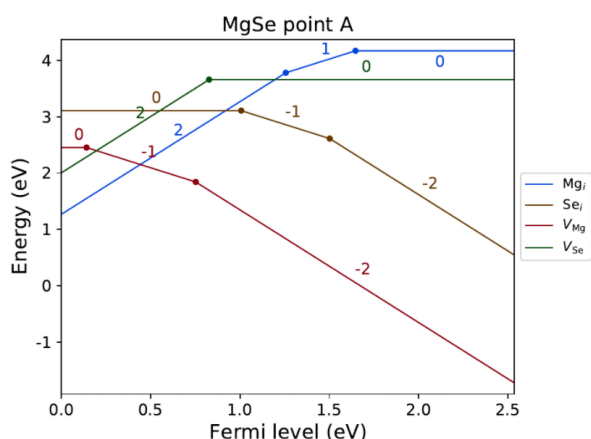
- 官方文档截图

## Welcome to pydefect documentation! 🔗

- Tutorial of pydefect
  - 1. Relaxation of the unit cell
  - 2. Calculation of band, DOS, and dielectric tensor
  - 3. Gathering unitcell information related to point-defect calculations
  - 4. Calculation of competing phases
  - 5. Construction of a supercell and defect initial setting file
  - 6. Decision of interstitial sites
  - 7. Creation of defect calculation directories
  - 8. Generation of defect\_entry.json
  - 9. Parsing supercell calculation results
  - 10. Corrections of defect formation energies in finite-size supercells
  - 11. Check defect eigenvalues and band-edge states in supercell calculations
  - 12. Plot defect formation energies
- Tips for first-principles calculations for point defects
  - 1. How to treat symmetry of point defects
  - 2. Tips for hybrid functional calculations
- Tutorial for calculation of vertical transition level
- Change log

- 实际例子:





## 2.2 Nonrad

### 2.2.1 Nonrad 程序: 计算载流子非辐射捕获系数

#### 介绍

半导体晶体中的点缺陷为载流子非辐射复合提供了一种手段，这种复合过程会影响器件的性能。Alkauskas 等人在 2014 年首先提出了基于量子力学方法来评估捕获过程 [Phys. Rev. B 90, 075202 (2014)]，详细来说他们提出了一种评估投影级加平面波中电子与声子耦合的方法。最近加州大学圣塔芭芭拉分校 Chris G. Van de Walle 教授课题组在 Alkauskas 等人的基础上开发了 Nonrad 程序用于评估载流子非辐射捕获系数。他们还表明，用高斯替换 Dirac 德尔塔函数的通用过程会在所得的捕获率中引入误差，并实现一种替代方案来适当考虑振动展宽。最后，他们通过与直接数值评估进行比较来评估对 Sommerfeld 参数使用解析近似的准确性。

- Nonrad 下载链接和官方文档：

[Github<sup>3</sup>](#)

[Document<sup>4</sup>](#)

- Nonrad 安装方法：

```
pip install nonrad
```

如果科研中使用了 Nonrad，请帮忙引用下面的文献：Ref<sup>5</sup>

- 文献截图

<sup>3</sup> <https://github.com/mturiansky/nonrad>

<sup>4</sup> <https://nonrad.readthedocs.io/en/latest/>

<sup>5</sup> <https://arxiv.org/pdf/2011.07433.pdf>

# Nonrad: Computing Nonradiative Capture Coefficients from First Principles

Mark E. Turiansky<sup>a,\*</sup>, Audrius Alkauskas<sup>b</sup>, Manuel Engel<sup>c</sup>, Georg Kresse<sup>c</sup>,  
Darshana Wickramaratne<sup>d</sup>, Jimmy-Xuan Shen<sup>a,e</sup>, Cyrus E. Dreyer<sup>f,g</sup>, Chris G.  
Van de Walle<sup>h</sup>

<sup>a</sup>Department of Physics, University of California, Santa Barbara, CA 93106-9530, U.S.A.

<sup>b</sup>Center for Physical Sciences and Technology (FTMC), Vilnius LT-10257, Lithuania

<sup>c</sup>University of Vienna, Faculty of Physics and Center for Computational Materials  
Sciences, Vienna A-1090, Austria

<sup>d</sup>Center for Computational Materials Science, US Naval Research Laboratory, Washington  
DC, 20375, U.S.A.

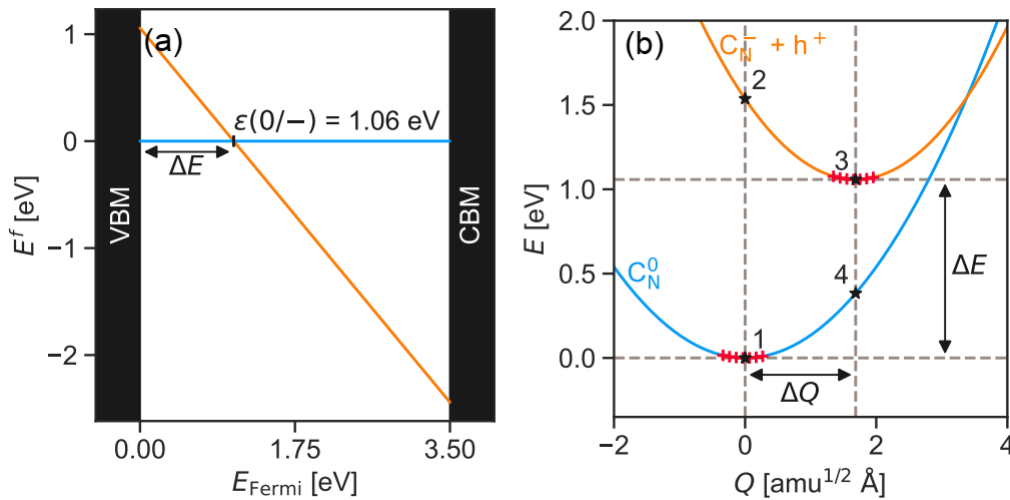
<sup>e</sup>Department of Materials Science and Engineering, University of California, Berkeley, CA  
94720-1760, U.S.A.

<sup>f</sup>Department of Physics and Astronomy, Stony Brook University, Stony Brook, New York  
11794-3800, U.S.A.

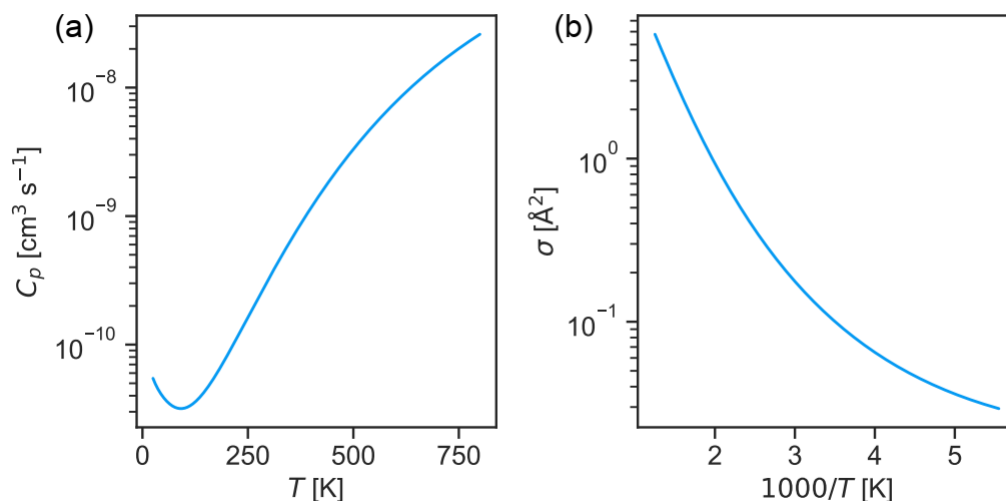
<sup>g</sup>Center for Computational Quantum Physics, Flatiron Institute, 162 5<sup>th</sup> Avenue, New  
York, New York 10010, U.S.A.

<sup>h</sup>Materials Department, University of California, Santa Barbara, CA 93106-5050, U.S.A.

## • 实际例子



## • 实际结果



## 2.3 PyXtal

### 2.3.1 PyXtal 程序: 晶体结构生成和对称性分析

#### 介绍

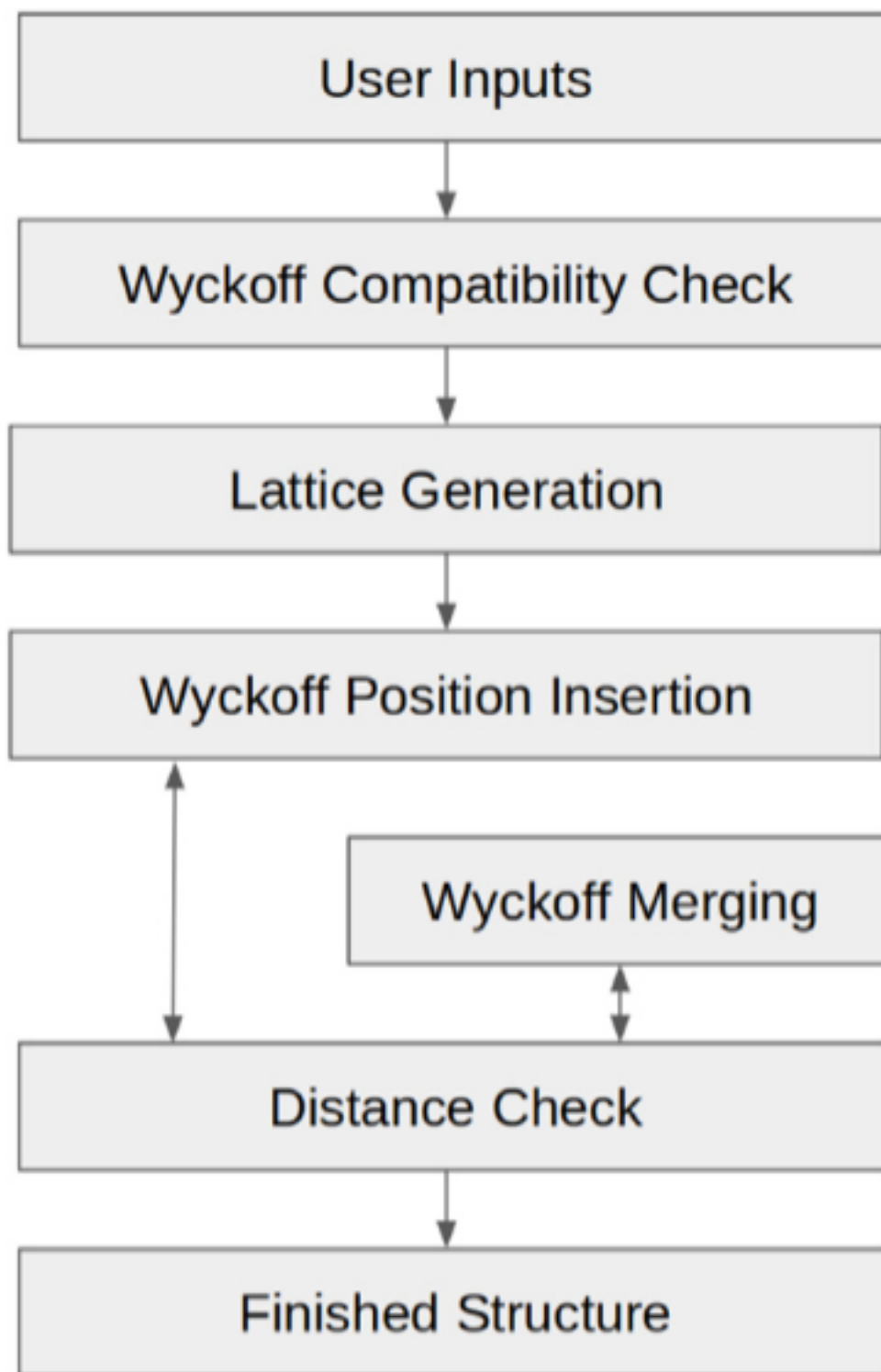
内华达大学物理与天文学系的助理教授 ZHu Qiang 课题组<sup>6</sup> 开发了 PyXtal, 这是一个基于 Python 编程语言的新软件包, 用于为原子和分子系统生成具有特定对称性和化学成分的结构。该软件为各种体系提供支持, 只需输入化学成分和对称基团信息, PyXtal 便可以通过逐步合并方案自动找到 Wyckoff 位置的合适组合。此外, 当给出分子几何结构时, PyXtal 可以生成具有不同维度的有机晶体, 且分子占据了一般和特殊的 Wyckoff 位置。PyXtal 也可以选择接受用户定义的参数 (例如, 晶胞参数、最小距离和 Wyckoff 位置)。通常, PyXtal 具有三个功能:

1. 生成自定义结构
2. 通过对称关系来调制结构
3. 连接需要生成随机对称结构的现有结构预测代码

此外, 我们提供了一些实用程序, 可简化结构分析, 包括对称性分析、几何优化和粉末 X 射线衍射 (XRD) 的模拟。

- PyXtal 内部原理框架:

<sup>6</sup> <https://qzhu2017.github.io>



- PyXtal 下载链接: Github<sup>7</sup>
- PyXtal 主页

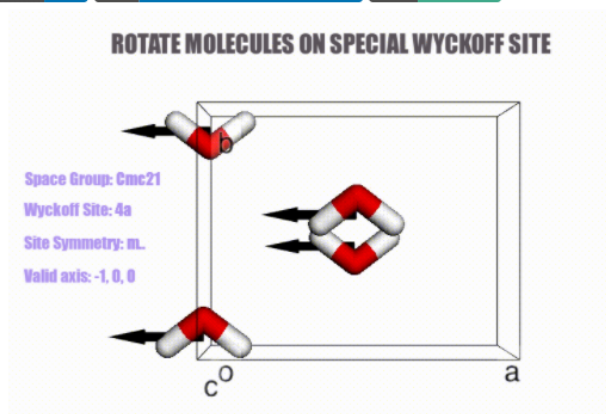
<sup>7</sup> <https://github.com/qzhu2017/PyXtal>



docs **passing** tests **falling** python 3 pypi v0.1.8 downloads 13k DOI 10.5281/zenodo.4048937 chat on gitter

## Content

- [Content](#)
- [Introduction](#)
- [Quick Start](#)
- [Current Features](#)
- [Installation](#)
- [Citation](#)
- [How to contribute?](#)
  - [user](#)
  - [developer](#)



- PyXtal 官方文档: Document<sup>8</sup>
- 官方文档

---

<sup>8</sup> <https://pyxtal.readthedocs.io/en/latest/>

PyXtal (pronounced `pie-crystal`) is an open source Python library for the ab-initio generation of random crystal structures. It has the following features:

- Generation of atomic structures for a given symmetry and stoichiometry (0-3D)
- Generation of molecular crystals (1-3D) with the support of special Wyckoff positions.
- Internal support of `cif` file and many other formats via `pymatgen` or `ASE`.
- Easy access to symmetry information (e.g., Wyckoff, site symmetry and international symbols).
- X-ray diffraction analysis and its [online application](#)
- Structural manipulation via symmetry constraint (group-subgroup relation)
- Geometry optimization from built-in and external optimization methods.

The current version is `0.1.8` at [GitHub](#). It is available for use under the MIT license. Expect updates upon request by [Qiang Zhu's group](#) at University of Nevada Las Vegas.



PyXtal is an open source project. You are welcome to contribute it directly via the [GitHub platform](#) or send your comments and suggestions to the [developer](#).

A basic tutorial is provided below for common functions. Additionally, documentation and source code are provided for individual modules.

For an experienced Python user who are familiar with Jupyter notebook, you are also encouraged to check out the following examples

- [Atomic crystal](#)
- [Molecular crystal](#)
- [XRD](#)

- 参考文献

 ELSEVIER	<p>Contents lists available at <a href="#">ScienceDirect</a></p> <p><b>Computer Physics Communications</b></p> <p>journal homepage: <a href="http://www.elsevier.com/locate/cpc">www.elsevier.com/locate/cpc</a></p>	
---	--	---

## PyXtal: A Python library for crystal structure generation and symmetry analysis☆☆☆



Scott Fredericks, Kevin Parrish, Dean Sayre, Qiang Zhu \*

*Department of Physics and Astronomy, University of Nevada Las Vegas, Las Vegas, NV 89154, USA*

## 2.4 Irvsp

### 2.4.1 Irvsp 程序：计算 VASP 中电子态的不可约表示

#### 介绍

最近 中国科学院物理研究所北京凝聚态物理国家重点实验室的 Wang Zhijun 教授课题组开发了一个开放源代码的程序 irvsp。它通过与 VASP 的接口为所有 230 个空间群计算电子状态的不可约表示。该代码由 VASP 软件包生成的基于平面波的波函数 (例如 WAVECAR) 和空间群算子 (列在 OUTCAR 中) 提供。该程序计算矩阵表示的迹线, 并确定三维布里渊区中所有能带和所有 k 点的对应不可约表示。它还适用于自旋轨道耦合 (SOC), 即用于双群。建立拓扑量子化学理论后, 分析能带, 能带的连接性和能带拓扑结构特别有用。因此, 开发了相关的库 irrep\_bcs.a, 可以很容易地通过其它的 ab-initio 软件包链接到该库。另外, 该程序已扩展到正交紧束缚 (TB) 哈密顿量, 例如电子或声子 TB 哈密顿量。

- irvsp 下载链接: Github<sup>9</sup>

## 2.5 phonopy

### 2.5.1 Phonopy: 声子性质处理软件

- phonopy<sup>10</sup>
- phono3py<sup>11</sup>
- 声子振动模式可视化 Phonon website<sup>12</sup>

---

<sup>9</sup> <https://github.com/zjwang11/irvsp>

<sup>10</sup> <https://phonopy.github.io/phonopy/>

<sup>11</sup> <https://phonopy.github.io/phono3py/>

<sup>12</sup> <http://henriquemiranda.github.io/phononwebsite/index.html>



## CHAPTER 3

---

### Installation

---

#### 3.1 libbeef

Compile libbeef using Intel Parallel Studio Xe 2020\_update4

##### 3.1.1 Download and Install

###### Download

1. Download version master<sup>13</sup> 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

---

<sup>13</sup> <https://github.com/vossjo/libbeef>

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

## 3.2 libxc

Compile libxc-4.3.4 using Intel Parallel Studio Xe 2020\_update4

### 3.2.1 Download and Install

#### Download

1. Download version 4.3.4<sup>14</sup> 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 libxc

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

## 3.3 hdf5

Compile hdf5-1.12.0 using Intel Parallel Studio Xe 2020\_update4

---

<sup>14</sup> <http://www.tddft.org/programs/libxc/down.php?file=4.3.4/libxc-4.3.4.tar.gz>

### 3.3.1 Download and Install

#### Download

1. Download version 1.12.0<sup>15</sup> of HDF5.

```
wget https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.12/hdf5-1.12.0/src/hdf5-1.12.0.  
→tar.gz
```

#### Compile

1. Create configure file configure.sh

```
#!/bin/bash  
  
./configure CC=mpiicc FC=mpiifort CXX=mpiicpc \  
--enable-fortran \  
--enable-parallel \  
--enable-shared \  
--prefix=/opt/software/hdf5-1.12.0 \  
--with-zlib=/opt/software/zlib-1.2.11
```

2. configure HDF5

```
bash configure.sh
```

3. compile and test HDF5

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

## 3.4 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 packages

### 3.4.1 Download and Install

#### Download

1. Download version 6.7<sup>16</sup> of Quantum-ESPRESSO.

<sup>15</sup> <https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.12/hdf5-1.12.0/src/hdf5-1.12.0.tar.gz>

<sup>16</sup> <https://github.com/QEF/q-e/releases/download/qe-6.7.0/qe-6.7-ReleasePack.tgz>

```
wget 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<sup>17</sup> and Wannier90 version 3.1.0<sup>18</sup> 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
```

(下页继续)

---

<sup>17</sup> [http://people.sissa.it/~dalcorsio/thermo\\_pw/thermo\\_pw.1.4.1.tar.gz](http://people.sissa.it/~dalcorsio/thermo_pw/thermo_pw.1.4.1.tar.gz)

<sup>18</sup> <https://github.com/wannier-developers/wannier90/archive/v3.1.0.tar.gz>

(续上页)

```
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`

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
```

## 3.5 ABINIT

Compile abinit-9.4.2 using Intel Parallel Studio Xe 2020\_update4

### 3.5.1 Download and Install

#### Download

1. Download version 9.4.2<sup>19</sup> of abinit.

```
wget 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<sup>20</sup> 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
```

---

<sup>19</sup> <https://www.abinit.org/sites/default/files/packages/abinit-9.4.2.tar.gz>

<sup>20</sup> <https://www.abinit.org/fallbacks>

## 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

```
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
```

## 3.6 Yambo

Compile Yambo-5.0.2 using Intel Parallel Studio Xe 2020\_update4

### 3.6.1 Download and Install

#### Download

1. Download version 5.0.2<sup>21</sup> 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<sup>22</sup>
- 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
```

---

<sup>21</sup> <https://github.com/yambo-code/yambo/archive/5.0.2.tar.gz>

<sup>22</sup> <https://codeload.github.com/yambo-code/yambo-libraries/tar.gz/0.0.2>

## 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" \
--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
```

## 3.7 BerkeleyGW

Compile BerkeleyGW-3.0.1 using Intel Parallel Studio Xe 2020\_update4

### 3.7.1 Download and Install

#### Download

- Download version 3.0.1<sup>23</sup> of BerkeleyGW.

---

<sup>23</sup> <https://berkeley.box.com/s/1dgnhiemo47lhxczrn6si71bwxoxor8>

## Compile

- Before compile BerkeleyGW package, one should compile hdf5-1.12.0 code Download version 1.12.0<sup>24</sup> of HDF5.

### 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 modify 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
```

(下页继续)

<sup>24</sup> <https://www.hdfgroup.org/package/hdf5-1-12-0-tar-gz/?wpdmdl=14582&refresh=618bc42bc76531636549675>

```

LINK    = mpiifort -qopenmp -ip -no-ipo
# We need the -fp-model precise 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

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)



### 4.1 VASP

#### 4.1.1 Mobility Calculation

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](http://www.quantum-espresso.org/)<sup>25</sup> 和 [EPW](https://epw-code.org/)<sup>26</sup> 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。

---

<sup>25</sup> <http://www.quantum-espresso.org/>

<sup>26</sup> <https://epw-code.org/>

## 理论基础

基于 Bardeen and Shockley<sup>27</sup> 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：

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

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  $E_1 = \Delta E / (\Delta l / l_0)$  确定， $\Delta E$  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， $\Delta l$  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：

$$m_d = \sqrt{m_x^* m_y^*} \text{ (Kg)}$$

$$E_1 = \Delta E / (\Delta l / l_0) \text{ (J) 在 VASP 中 } \Delta E \text{ 的单位是 eV, } 1 \text{ eV} = 1.6 \times 10^{-19} \text{ J}$$

$$C_{2D} = 2[\partial^2 E / \partial(\Delta l / l_0)^2] / S_0 \text{ (J/m}^2\text{)} \text{ 其中 } E \text{ 是总能量 (eV), } S_0 \text{ 表示面积 (}^2\text{)}$$

$$e : 1.6 \times 10^{-19} \text{ C}$$

$$\hbar : 6.626 \times 10^{-34} \text{ J} \cdot \text{s}$$

$$k_B : 1.38 \times 10^{-23} \text{ J/K}$$

$$m^* : \text{Kg}$$

换算过程：

$$\begin{aligned} C \cdot J^3 \cdot s^3 \cdot \frac{J}{m^2} &= \frac{C \cdot J \cdot s^3}{m^2 \cdot Kg^2} = \frac{C \cdot J}{\frac{m^2 \cdot Kg^2}{s^3} \cdot \frac{s}{s} \cdot \frac{m^2}{m^2}} = \frac{C \cdot J}{\frac{s}{m^2} \frac{m^4 \cdot Kg^2}{s^4}} = \frac{m^2}{s} \cdot \frac{C}{J} = m^2 \cdot s^{-1} \cdot V^{-1} \\ &= 10^4 \times cm^2 \cdot s^{-1} \cdot V^{-1} \end{aligned}$$

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

## 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<sup>28</sup>

<sup>27</sup> <https://doi.org/10.1103/PhysRev.80.72>

<sup>28</sup> <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: c recip.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())

```

## 二维 InSe 有效质量计算过程

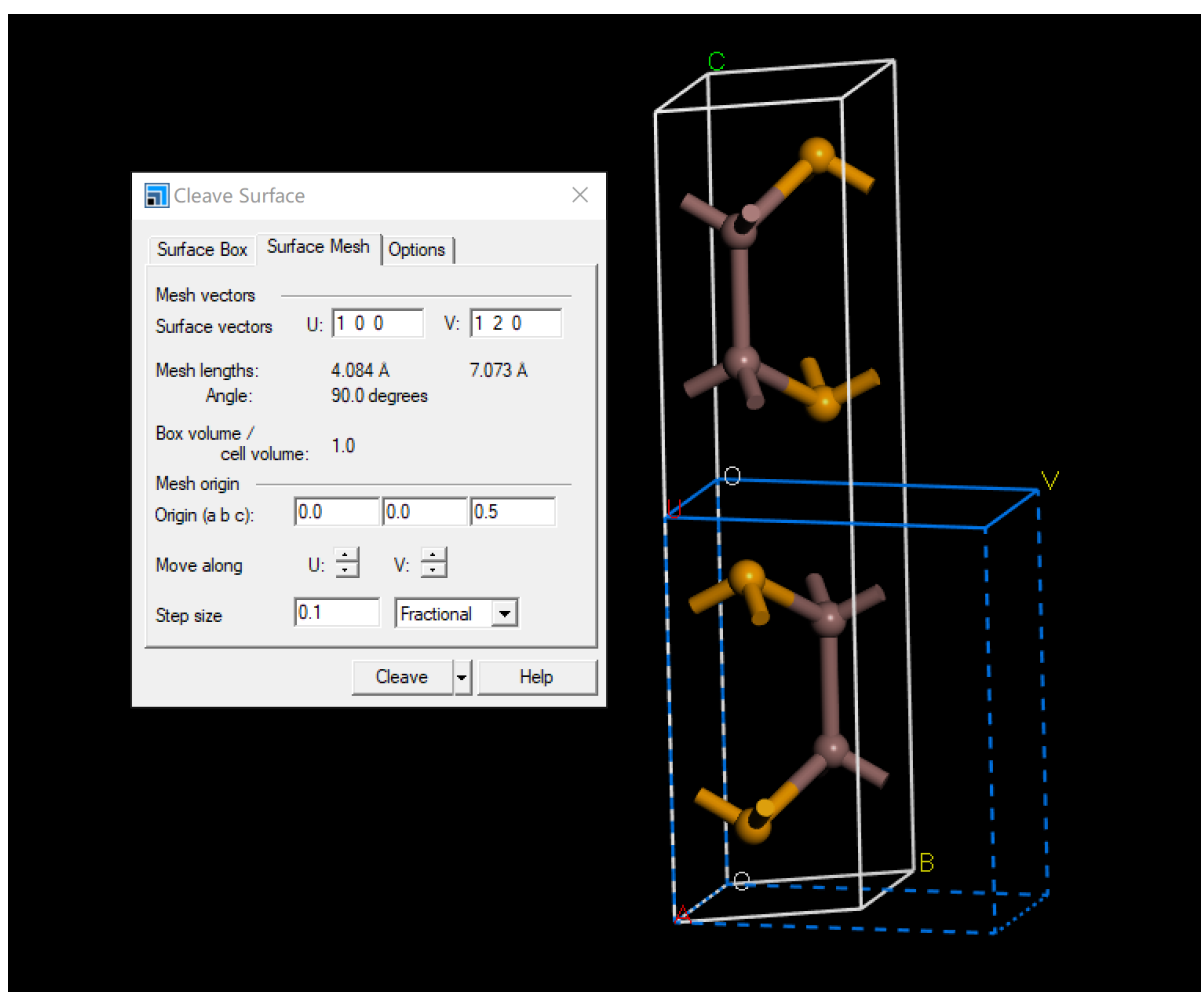
### 建模

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



```

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

### 4.1.2 Formation Energy Calculation

```
#!/usr/bin/env python3

"""
读取POSCAR与OUTCAR文件，自动计算形成能
"""

"""
以字典的形式定义每个原子的能量，需要从Materials Project库里查找补充
"""
Energy_atom = {'O': -4.9480, 'Mg': -1.6003, 'Ca': -1.9995, 'Zn': -1.2595, 'Cu': -4.0992,
               'Sr': -1.6895, 'Cd': -0.9062, 'Ba': -1.9190, 'Yb': -1.5396, 'Pt': -6.0709,
               'Pd': -3.7126
               }

"""
读取POSCAR文件，计算原子的总数和总能
"""
with open("POSCAR", 'r', encoding='utf-8') as poscar:
    list_poscar = poscar.readlines()
for i in range(0, len(list_poscar)):
    list_poscar[i] = list_poscar[i].rstrip('\n')
```

(下页继续)

(续上页)

```

list_element = ' '.join(list_poscar[5].strip().split('\t')).split()
# print("list_element = ", list_element)
list_num_element = ' '.join(list_poscar[6].strip().split('\t')).split()
# print("list_num_element = ", list_num_element)
total_atoms = 0
Toten_atoms = 0
for i in range(0, len(list_element)):
    total_atoms = total_atoms + int(list_num_element[i])
    Toten_atoms = Toten_atoms + Energy_atom[str(list_element[i])]*int(list_num_
↪element[i])
# print("total_atoms = ", total_atoms)
# print("Toten_atoms = ", Toten_atoms)

"""
读取OUTCAR文件获得体系总能
"""
with open("OUTCAR", 'r', encoding='utf-8') as f:
    lines = f.readlines()
    for line in lines:
        if ("TOTEN" in line):
            # print(line.rstrip('\n'))
            list_TOTEN = line
Toten_system = float(list_TOTEN.rstrip().split()[-2])

Ef = (Toten_system-Toten_atoms)/total_atoms
print("Ef = ", '%.3f' % Ef, "eV/atom")

```

## 4.2 BerkeleyGW

### 4.2.1 GW 计算过程

- 计算所需的文件结构如下：

```

5-epsilon
  bash
  epsilon.inp
6-sigma
  bash
  eqp.py
  sigma.inp
7-kernel
  bash
  kernel.inp
8-absorption
  absorption.inp
  bash

```

(下页继续)



(续上页)

```

9-inteqp
  bash
  inteqp.inp
ESPRESSO
0-kgrid
  script_0
  WFN__fi.in
  WFN__fiq.in
  WFN.in
  WFNq.in
1-scf
  bash
  B.SG15.PBE.UPF
  input
  N.SG15.PBE.UPF
2-wfn
  bash
  degeneracy_check.x
  input
  pp_in
3-wfnq
  bash
  input
  pp_in
5-wfn__fi
  bash
  input
  pp_in
6-wfn__fiq
  bash
  input
  pp_in
7-band
  bash
  input
  pp_in
script_1

```

## Quantum ESPRESSO 计算

### 0-kgrid 文件修改

- WFN.in

```

5 5 1      # k点网格
0.0 0.0 0.0
0.0 0.0 0.0

```

(下页继续)

(续上页)

```

2.5124282300    0.0000000000    0.0000000000 # 晶格常数
-1.2562141150    2.1758266724    0.0000000000 # 晶格常数
0.0000000000    0.0000000000    7.7072650000 # 晶格常数

4 # 原子总数
1    0.6666666700    0.3333333300    0.7500000000 # 第一个原子分数坐标
1    0.3333333300    0.6666666700    0.2500000000 # 第二个原子分数坐标
2    0.6666666700    0.3333333300    0.2500000000 # 第三个原子分数坐标
2    0.3333333300    0.6666666700    0.7500000000 # 第四个原子分数坐标
24 24 81 # 傅里叶变换网格
.false.

```

- WFNq.in

```

5 5 1
0.0 0.0 0.0
0.001 0.0 0.0 # 第一个数变为0.001
2.5124282300    0.0000000000    0.0000000000
-1.2562141150    2.1758266724    0.0000000000
0.0000000000    0.0000000000    7.7072650000

4
1    0.6666666700    0.3333333300    0.7500000000
1    0.3333333300    0.6666666700    0.2500000000
2    0.6666666700    0.3333333300    0.2500000000
2    0.3333333300    0.6666666700    0.7500000000
24 24 81
.false.

```

- WFN\_fi.in

```

7 7 1 # k点网格变密，根据服务器计算能力修改
0.0 0.0 0.0
0.0 0.0 0.0
2.5124282300    0.0000000000    0.0000000000
-1.2562141150    2.1758266724    0.0000000000
0.0000000000    0.0000000000    7.7072650000

4
1    0.6666666700    0.3333333300    0.7500000000
1    0.3333333300    0.6666666700    0.2500000000
2    0.6666666700    0.3333333300    0.2500000000
2    0.3333333300    0.6666666700    0.7500000000
24 24 81
.false.

```

- WFN\_fiq.in

```

7 7 1

```

(下页继续)

(续上页)

```

0.0  0.0  0.0
0.001 0.001 0.0 # 第一个与第二个数变为0.001
      2.5124282300      0.0000000000      0.0000000000
      -1.2562141150      2.1758266724      0.0000000000
      0.0000000000      0.0000000000      7.7072650000

4
1      0.6666666700      0.3333333300      0.7500000000
1      0.3333333300      0.6666666700      0.2500000000
2      0.6666666700      0.3333333300      0.2500000000
2      0.3333333300      0.6666666700      0.7500000000
24 24 81
.false.

```

- script\_0

```

#!/bin/bash

#
KGRID="/home/gengzi/apps/BerkeleyGW-2.1/bin/kgrid.x" #修改BerkeleyGW路径

#
$KGRID ./WFN.in ./WFN.out ./WFN.log
$KGRID ./WFNq.in ./WFNq.out ./WFNq.log
#$KGRID ./WFN_co.in ./WFN_co.out ./WFN_co.log
$KGRID ./WFN_fi.in ./WFN_fi.out ./WFN_fi.log
$KGRID ./WFN_fiq.in ./WFN_fiq.out ./WFN_fiq.log

cd ..

```

- 生成输入文件

```
bash script_0
```

## 链接全局环境变量

```

#!/bin/bash
#
sys=BN # 体系名称
PO1=B.SG15.PBE.UPF # 势文件名称
PO2=N.SG15.PBE.UPF # 势文件名称

mkdir ./2-wfn/$sys.save
ln -s ../1-scf/$PO1 ./2-wfn # 连接势文件, 如果体系原子增加, 相应的势文件也要增加
ln -s ../1-scf/$PO2 ./2-wfn
#ln -s ../1-scf/vdW_kernel_table ./2-wfn

```

(下页继续)

(续上页)

```

ln -s ../../1-scf/$sys.save/data-file-schema.xml ./2-wfn/$sys.save
ln -s ../../1-scf/$sys.save/charge-density.dat ./2-wfn/$sys.save
#
mkdir ./3-wfnq/$sys.save
ln -s ../1-scf/$PO1 ./3-wfnq
ln -s ../1-scf/$PO2 ./3-wfnq
#ln -s ../1-scf/vdW_kernel_table ./3-wfnq
ln -s ../../1-scf/$sys.save/data-file-schema.xml ./3-wfnq/$sys.save
ln -s ../../1-scf/$sys.save/charge-density.dat ./3-wfnq/$sys.save
#
#
mkdir ./5-wfn_fi/$sys.save
ln -s ../1-scf/$PO1 ./5-wfn_fi
ln -s ../1-scf/$PO2 ./5-wfn_fi
#ln -s ../1-scf/vdW_kernel_table ./5-wfn_fi
ln -s ../../1-scf/$sys.save/data-file-schema.xml ./5-wfn_fi/$sys.save
ln -s ../../1-scf/$sys.save/charge-density.dat ./5-wfn_fi/$sys.save

mkdir ./6-wfn_fiq/$sys.save
ln -s ../1-scf/$PO1 ./6-wfn_fiq
ln -s ../1-scf/$PO2 ./6-wfn_fiq
#ln -s ../1-scf/vdW_kernel_table ./6-wfn_fiq
ln -s ../../1-scf/$sys.save/data-file-schema.xml ./6-wfn_fiq/$sys.save
ln -s ../../1-scf/$sys.save/charge-density.dat ./6-wfn_fiq/$sys.save

mkdir ./7-band/$sys.save
ln -s ../1-scf/$PO1 ./7-band
ln -s ../1-scf/$PO2 ./7-band
#ln -s ../1-scf/vdW_kernel_table ./6-wfn_fiq
ln -s ../../1-scf/$sys.save/data-file-schema.xml ./7-band/$sys.save
ln -s ../../1-scf/$sys.save/charge-density.dat ./7-band/$sys.save

ln -s ../ESPRESSO/2-wfn/wfn.complex ../5-epsilon/WFN
ln -s ../ESPRESSO/3-wfnq/wfn.complex ../5-epsilon/WFNq
#
ln -s ../ESPRESSO/2-wfn/vxc.dat ../6-sigma/vxc.dat
ln -s ../ESPRESSO/2-wfn/rho.cplx ../6-sigma/RHO
ln -s ../ESPRESSO/2-wfn/wfn.complex ../6-sigma/WFN_inner
ln -s ../5-epsilon/eps0mat ../6-sigma
ln -s ../5-epsilon/epsmat ../6-sigma
#
ln -s ../ESPRESSO/2-wfn/wfn.complex ../7-kernel/WFN_co
ln -s ../5-epsilon/eps0mat ../7-kernel
ln -s ../5-epsilon/epsmat ../7-kernel
#
ln -s ../ESPRESSO/2-wfn/wfn.complex ../8-absorption/WFN_co
ln -s ../ESPRESSO/5-wfn_fi/wfn.complex ../8-absorption/WFN_fi
ln -s ../ESPRESSO/6-wfn_fiq/wfn.complex ../8-absorption/WFNq_fi
ln -s ../5-epsilon/eps0mat ../8-absorption

```

(下页继续)

(续上页)

```
ln -s ../5-epsilon/epsmat ../8-absorption
ln -s ../7-kernel/bsedmat ../8-absorption
ln -s ../7-kernel/bsexmat ../8-absorption
#
ln -s ../ESPRESSO/2-wfn/wfn.complex ../9-inteqp/WFN_co
ln -s ../ESPRESSO/7-band/wfn.complex ../9-inteqp/WFN_fi
ln -s ../5-epsilon/eps0mat ../9-inteqp
ln -s ../5-epsilon/epsmat ../9-inteqp
#
ln -s ../8-absorption/eqp_co.dat ../9-inteqp
```

- 运行脚本

```
bash script_1
```

1-scf

- input 文件

```
&CONTROL
calculation = 'scf'
etot_conv_thr = 4.0000000000d-05
forc_conv_thr = 1.0000000000d-04
outdir = './'
prefix = 'BN'
pseudo_dir = './'
wfcdir = './'
tprnfor = .true.
tstress = .true.
verbosity = 'high'
wf_collect = .false.
/
&SYSTEM
ecutwfc = 4.0000000000d+01
ibrav = 0
nat = 4
ntyp = 2
/
&ELECTRONS
conv_thr = 1.0000000000d-7
electron_maxstep = 80
mixing_beta = 4.0000000000d-01
mixing_mode = 'plain'
diagonalization = 'david'
diago_david_ndim = 8
diago_full_acc = .true.
/
```

(下页继续)

(续上页)

```

ATOMIC_SPECIES
B      10.811 B.SG15.PBE.UPF
N      14.0067 N.SG15.PBE.UPF
ATOMIC_POSITIONS crystal
B      0.6666666700      0.3333333300      0.7500000000 ! 顺序与0-kgrid中设置相同
B      0.3333333300      0.6666666700      0.2500000000
N      0.6666666700      0.3333333300      0.2500000000
N      0.3333333300      0.6666666700      0.7500000000
K_POINTS automatic
15 15 5 0 0 0
CELL_PARAMETERS angstrom
      2.5124282300      0.0000000000      0.0000000000
      -1.2562141150      2.1758266724      0.0000000000
      0.0000000000      0.0000000000      7.7072650000

```

- 提交任务脚本

```

#!/bin/sh
#PBS -N hey
#PBS -l nodes=1:ppn=40,walltime=500:00:00
##PBS -q workq

cd $PBS_O_WORKDIR
source /home/gengzi/apps/intel/19/parallel_studio_xe_2019/psxevars.sh

date "+01 Today's date is: %D. The time execution %T" >> time.info
mpirun -np 40 /home/gengzi/apps/qe-6.5/bin/pw.x -nk 4 -in ./input &> ./out
rm ./*.wfc*
date "+02 Today's date is: %D. The time finish %T" >> time.info

```

2-wfn

- input

```

&CONTROL
      title = 'BN' ,
      calculation = 'bands' ,
      restart_mode = 'from_scratch' ,
      wf_collect = .false. ,
      outdir = './' ,
      wfedir = './' ,
      pseudo_dir = './' ,
      prefix = 'BN' ,
      tstress = .false. ,
      tprnfor = .false. ,
/

```

(下页继续)

(续上页)

```

&SYSTEM
    ibrav = 0,
    nat = 4,
    ntyp = 2,
    ecutwfc = 40,
    nbnd = 80, !要求能带是占据态的10倍

/
&ELECTRONS
    electron_maxstep = 100
    conv_thr = 1.0d-7
    mixing_mode = 'plain'
    mixing_beta = 0.4
    diagonalization = 'david'
    diago_david_ndim = 8
    diago_full_acc = .true.

/
ATOMIC_SPECIES
B    10.811 B.SG15.PBE.UPF
N    14.0067 N.SG15.PBE.UPF
ATOMIC_POSITIONS crystal
B    0.6666666700    0.3333333300    0.7500000000
B    0.3333333300    0.6666666700    0.2500000000
N    0.6666666700    0.3333333300    0.2500000000
N    0.3333333300    0.6666666700    0.7500000000
CELL_PARAMETERS angstrom
    2.5124282300    0.0000000000    0.0000000000
    -1.2562141150    2.1758266724    0.0000000000
    0.0000000000    0.0000000000    7.7072650000
K_POINTS crystal ! 来源于0-kgrid/WFN.out

```

- pp\_in

```

&input_pw2bgw
    prefix = 'BN'          ! 体系名称
    real_or_complex = 2
    wfng_flag = .true.
    wfng_file = 'wfn.complex'
    wfng_kgrid = .true.
    wfng_nk1 = 5           ! k点网格, 与WFN.in设置相同
    wfng_nk2 = 5           ! k点网格, 与WFN.in设置相同
    wfng_nk3 = 1           ! k点网格, 与WFN.in设置相同
    wfng_dk1 = 0.0
    wfng_dk2 = 0.0
    wfng_dk3 = 0.0
    rhog_flag = .true.
    rhog_file = 'rho.cplx'
    vxcg_flag = .false.

```

(下页继续)

(续上页)

```

vxcg_file = 'vxc.cplx'
vxc_flag = .true.
vxc_file = 'vxc.dat'
vxc_diag_nmin = 4
vxc_diag_nmax = 13
vxc_offdiag_nmin = 0
vxc_offdiag_nmax = 0
/

```

- 提交任务脚本

```

#!/bin/sh
#PBS -N hey
#PBS -l nodes=1:ppn=40,walltime=500:00:00
##PBS -q workq

cd $PBS_O_WORKDIR
source /home/gengzi/apps/intel/19/parallel_studio_xe_2019/psxevars.sh

date "+01 Today's date is: %D. The time execution %T" >> time.info
mpirun -np 40 /home/gengzi/apps/qe-6.5/bin/pw.x -nk 1 -in ./input &> ./out
mpirun -np 40 /home/gengzi/apps/qe-6.5/bin/pw2bgw.x -in ./pp_in &> ./pp_out
#rm ./*.wfc*
#rm ./*.igk*
date "+02 Today's date is: %D. The time finish %T" >> time.info

```

- degeneracy\_check.x 程序后处理

```
degeneracy_check.x complex >> aa
```

- aa 文件分析

```

Reading eigenvalues from file wfn.complex
Number of spins:      1
Number of bands:      80
Number of k-points:   25

== Degeneracy-allowed numbers of bands (for epsilon and sigma) ==
  1
  2
  3
  4 ! 取为vxc_diag_nmin的值
  6
  8
  9
 10
 11
 13

```

(下页继续)



(续上页)

15 ! 取为 vxc\_diag\_nmax 的值

16

17

18

19

20

21

22

24

26

27

29

30

32

34

36

37

38

39

40

41

43

45

47

48

50

52

53

54

55

57

58

59

61

63

64

66

68

69

70

72

74

75

76

77

78

79 ! 取为 5-epsilon/epsilon.inp 中 number\_bands 的值, 通常比总能带数小 1

Note: cannot assess whether or not highest band 80 is degenerate.

(下页继续)

(续上页)

```
== Degeneracy-allowed numbers of valence bands (for inteqp, kernel, and absorption) ==
2
4
5
6
7
8

== Degeneracy-allowed numbers of conduction bands (for inteqp, kernel, and absorption) ==
1
2
3
5
7
8
9
10
11
12
13
14
16
18
19
21
22
24
26
28
29
30
31
32
33
35
37
39
40
42
44
45
46
47
49
50
51
53
55
56
```

(下页继续)

(续上页)

```

58
60
61
62
64
66
67
68
69
70
71

```

Note: cannot assess whether or not highest conduction band 72 is degenerate.

- 修改 pp\_in 文件

```

&input_pw2bgw
  prefix = 'BN'
  real_or_complex = 2
  wfng_flag = .true.
  wfng_file = 'wfn.complex'
  wfng_kgrid = .true.
  wfng_nk1 = 5
  wfng_nk2 = 5
  wfng_nk3 = 1
  wfng_dk1 = 0.0
  wfng_dk2 = 0.0
  wfng_dk3 = 0.0
  rhog_flag = .true.
  rhog_file = 'rho.cplx'
  vxcg_flag = .false.
  vxcg_file = 'vxc.cplx'
  vxc_flag = .true.
  vxc_file = 'vxc.dat'
  vxc_diag_nmin = 4      ! 见Note说明
  vxc_diag_nmax = 13    ! 见Note说明
  vxc_offdiag_nmin = 0
  vxc_offdiag_nmax = 0
/

```

注解: vxc\_diag\_nmin 取价带顶下约 5-10 条带, 其中取的最下面的一条带要求与下面的带连续, 例如取的最下面的带是 4, 则下一条必须是 3

vxc\_diag\_nmax 取导带顶上约 5-10 条带, 其中取的最上面的一条带要求与上面的带连续, 例如取的最上面的带是 13, 则再上一条带必须是 14

- 提交任务重新计算

```
#!/bin/sh
#PBS -N hey
#PBS -l nodes=1:ppn=40,walltime=500:00:00
##PBS -q workq

cd $PBS_O_WORKDIR
source /home/gengzi/apps/intel/19/parallel_studio_xe_2019/psxevars.sh

date "+01 Today's date is: %D. The time execution %T" >> time.info
# mpirun -np 40 /home/gengzi/apps/qe-6.5/bin/pw.x -nk 1 -in ./input &> ./out
mpirun -np 40 /home/gengzi/apps/qe-6.5/bin/pw2bgw.x -in ./pp_in &> ./pp_out
#rm ./*.wfc*
#rm ./*.igk*
date "+02 Today's date is: %D. The time finish %T" >> time.info
```

3-wfnq

- input

```
&CONTROL
    title = 'BN' ,
    calculation = 'bands' ,
    restart_mode = 'from_scratch' ,
    wf_collect = .false. ,
    outdir = './' ,
    wfedir = './' ,
    pseudo_dir = './' ,
    prefix = 'BN' ,
    tstress = .false. ,
    tprnfor = .false. ,
/
&SYSTEM
    ibrav = 0,
    nat = 4,
    ntyp = 2,
    ecutwfc = 40,
    nbnd = 8,    ! 设置为占据态数目
/
&ELECTRONS
    electron_maxstep = 100
    conv_thr = 1.0d-7
    mixing_mode = 'plain'
    mixing_beta = 0.4
    diagonalization = 'david'
    diago_david_ndim = 8
    diago_full_acc = .true.
```

(下页继续)

(续上页)

```

/
ATOMIC_SPECIES
B      10.811 B.SG15.PBE.UPF
N      14.0067 N.SG15.PBE.UPF
ATOMIC_POSITIONS crystal
B      0.6666666700      0.3333333300      0.7500000000
B      0.3333333300      0.6666666700      0.2500000000
N      0.6666666700      0.3333333300      0.2500000000
N      0.3333333300      0.6666666700      0.7500000000
CELL_PARAMETERS angstrom
      2.5124282300      0.0000000000      0.0000000000
      -1.2562141150      2.1758266724      0.0000000000
      0.0000000000      0.0000000000      7.7072650000
K_POINTS crystal ! 来源于0-kgrid/WFNq.out中的k点

```

- pp\_in

```

&input_pw2bgw
  prefix = 'BN'           ! 体系名称
  real_or_complex = 2
  wfng_flag = .true.
  wfng_file = 'wfn.complex'
  wfng_kgrid = .true.
  wfng_nk1 = 5             ! 与WFNq.in设置相同
  wfng_nk2 = 5             ! 与WFNq.in设置相同
  wfng_nk3 = 1             ! 与WFNq.in设置相同
  wfng_dk1 = 0.005         ! k点乘以0.001
  wfng_dk2 = 0.0
  wfng_dk3 = 0.0

```

- 提交任务脚本

```

#!/bin/sh
#PBS -N hey
#PBS -l nodes=1:ppn=40,walltime=500:00:00
##PBS -q workq

cd $PBS_O_WORKDIR
source /home/gengzi/apps/intel/19/parallel_studio_xe_2019/psxevars.sh

date "+01 Today's date is: %D. The time execution %T" >> time.info
mpirun -np 40 /home/gengzi/apps/qe-6.5/bin/pw.x -nk 1 -in ./input &> ./out
mpirun -np 40 /home/gengzi/apps/qe-6.5/bin/pw2bgw.x -in ./pp_in &> ./pp_out
rm ./*.wfc*
rm ./*.igk*
date "+02 Today's date is: %D. The time finish %T" >> time.info

```

5-wfn\_fi

- input

```
&CONTROL
    title = 'BN' ,
    calculation = 'bands' ,
    restart_mode = 'from_scratch' ,
    wf_collect = .false. ,
    outdir = './' ,
    wfcdir = './' ,
    pseudo_dir = './' ,
    prefix = 'BN' ,
    tstress = .false. ,
    tprnfor = .false. ,
/
&SYSTEM
    ibrav = 0,
    nat = 4,
    ntyp = 2,
    ecutwfc = 40,
    nbnd = 80,      !占据态乘以10
/
&ELECTRONS
    electron_maxstep = 100
    conv_thr = 1.0d-7
    mixing_mode = 'plain'
    mixing_beta = 0.4
    diagonalization = 'david'
    diago_david_ndim = 8
    diago_full_acc = .true.
/
ATOMIC_SPECIES
B      10.811 B.SG15.PBE.UPF
N      14.0067 N.SG15.PBE.UPF
ATOMIC_POSITIONS crystal
B      0.6666666700    0.3333333300    0.7500000000
B      0.3333333300    0.6666666700    0.2500000000
N      0.6666666700    0.3333333300    0.2500000000
N      0.3333333300    0.6666666700    0.7500000000
CELL_PARAMETERS angstrom
    2.5124282300    0.0000000000    0.0000000000
    -1.2562141150    2.1758266724    0.0000000000
    0.0000000000    0.0000000000    7.7072650000
K_POINTS crystal ! k点来源于0-kgrid/WFN_fi.out
```

- pp\_in

```
&input_pw2bgw
  prefix = 'BN'
  real_or_complex = 2
  wfng_flag = .true.
  wfng_file = 'wfn.complex'
  wfng_kgrid = .true.
  wfng_nk1 = 7      ! 与WFN_fi.in设置相同
  wfng_nk2 = 7      ! 与WFN_fi.in设置相同
  wfng_nk3 = 1      ! 与WFN_fi.in设置相同
  wfng_dk1 = 0.0
  wfng_dk2 = 0.0
  wfng_dk3 = 0.0
/
```

- 提交任务脚本

```
#!/bin/sh
#PBS -N hey
#PBS -l nodes=1:ppn=40,walltime=500:00:00
##PBS -q workq

cd $PBS_O_WORKDIR
source /home/gengzi/apps/intel/19/parallel_studio_xe_2019/psxevars.sh

date "+01 Today's date is: %D. The time execution %T" >> time.info
mpirun -np 40 /home/gengzi/apps/qe-6.5/bin/pw.x -nk 1 -in ./input &> ./out
mpirun -np 40 /home/gengzi/apps/qe-6.5/bin/pw2bgw.x -in ./pp_in &> ./pp_out
rm ./*.wfc*
rm ./*.igk*
date "+02 Today's date is: %D. The time finish %T" >> time.info
```

6-wfn\_fiq

- input

```
&CONTROL
  title = 'BN' ,
  calculation = 'bands' ,
  restart_mode = 'from_scratch' ,
  wf_collect = .false. ,
  outdir = './' ,
  wfedir = './' ,
  pseudo_dir = './' ,
  prefix = 'BN' ,
  tstress = .false. ,
  tprnfor = .false. ,
/
```

(下页继续)

(续上页)

```

&SYSTEM
    ibrav = 0,
    nat = 4,
    ntyp = 2,
    ecutwfc = 40,
    nbnd = 8,    ! 设置为占据态

/
&ELECTRONS
    electron_maxstep = 100
    conv_thr = 1.0d-7
    mixing_mode = 'plain'
    mixing_beta = 0.4
    diagonalization = 'david'
    diago_david_ndim = 8
    diago_full_acc = .true.

/
ATOMIC_SPECIES
B    10.811 B.SG15.PBE.UPF
N    14.0067 N.SG15.PBE.UPF
ATOMIC_POSITIONS crystal
B    0.6666666700    0.3333333300    0.7500000000
B    0.3333333300    0.6666666700    0.2500000000
N    0.6666666700    0.3333333300    0.2500000000
N    0.3333333300    0.6666666700    0.7500000000
CELL_PARAMETERS angstrom
    2.5124282300    0.0000000000    0.0000000000
    -1.2562141150    2.1758266724    0.0000000000
    0.0000000000    0.0000000000    7.7072650000
K_POINTS crystal ! k点来源于0-kgrid/WFN_fiq.out

```

- pp\_in

```

&input_pw2bgw
    prefix = 'BN'
    real_or_complex = 2
    wfng_flag = .true.
    wfng_file = 'wfn.complex'
    wfng_kgrid = .true.
    wfng_nk1 = 7    ! 与WFN_fiq.in设置相同
    wfng_nk2 = 7    ! 与WFN_fiq.in设置相同
    wfng_nk3 = 1    ! 与WFN_fiq.in设置相同
    wfng_dk1 = 0.007 ! k点乘以0.001
    wfng_dk2 = 0.007 ! k点乘以0.001
    wfng_dk3 = 0.0

/

```

- 提交任务的脚本



```

#!/bin/sh
#PBS -N hey
#PBS -l nodes=1:ppn=40,walltime=500:00:00
##PBS -q workq

cd $PBS_O_WORKDIR
source /home/gengzi/apps/intel/19/parallel_studio_xe_2019/psxevars.sh

date "+01 Today's date is: %D. The time execution %T" >> time.info
mpirun -np 40 /home/gengzi/apps/qe-6.5/bin/pw.x -nk 1 -in ./input &> ./out
mpirun -np 40 /home/gengzi/apps/qe-6.5/bin/pw2bgw.x -in ./pp_in &> ./pp_out
rm ./*.wfc*
rm ./*.igk*
date "+02 Today's date is: %D. The time finish %T" >> time.info

```

## 7-band

- input

```

&CONTROL
  calculation = 'bands'
  etot_conv_thr = 4.0000000000d-05
  forc_conv_thr = 1.0000000000d-04
  outdir = './'
  prefix = 'BN'
  pseudo_dir = './'
  wfcdir = './'
  tprnfor = .false.
  tstress = .false.
  verbosity = 'high'
  wf_collect = .false.
/
&SYSTEM
  ecutwfc = 4.0000000000d+01
  ibrav = 0
  nat = 4
  ntyp = 2
  nbnd = 80          ! 占据态乘以10
/
&ELECTRONS
  conv_thr = 1.0000000000d-7
  electron_maxstep = 80
  mixing_beta = 4.0000000000d-01
  mixing_mode = 'plain'
  diagonalization = 'david'
  diago_david_ndim = 8
  diago_full_acc = .true.

```

(下页继续)

(续上页)

```

/
ATOMIC_SPECIES
B      10.811 B.SG15.PBE.UPF
N      14.0067 N.SG15.PBE.UPF
ATOMIC_POSITIONS crystal
B      0.6666666700      0.3333333300      0.7500000000
B      0.3333333300      0.6666666700      0.2500000000
N      0.6666666700      0.3333333300      0.2500000000
N      0.3333333300      0.6666666700      0.7500000000
CELL_PARAMETERS angstrom
      2.5124282300      0.0000000000      0.0000000000
      -1.2562141150      2.1758266724      0.0000000000
      0.0000000000      0.0000000000      7.7072650000

K_POINTS crystal {crystal_b} ! 能带计算的高对称点路径
4
0      0      0 50
0.5    0      0 50
0.3333 0.3333 0 50
0      0      0 0

```

- pp\_in

```

&input_pw2bgw
  prefix = 'BN'
  real_or_complex = 2
  wfng_flag = .true.
  wfng_file = 'wfn.complex'
  wfng_kgrid = .true.
  wfng_nk1 = 0
  wfng_nk2 = 0
  wfng_nk3 = 0
  wfng_dk1 = 0.0
  wfng_dk2 = 0.0
  wfng_dk3 = 0.0
  wfng_occupation=.ture
  wfng_nvmin=1      ! 价带最小值，一般设置为1
  wfng_nvmax= 8     ! 价带最大值，也就是占据态
/

```

- 提交任务脚本

```

#!/bin/sh
#PBS -N hey
#PBS -l nodes=1:ppn=40,walltime=500:00:00
##PBS -q workq

cd $PBS_O_WORKDIR

```

(下页继续)

(续上页)

```
source /home/gengzi/apps/intel/19/parallel_studio_xe_2019/psxevars.sh

date "+01 Today's date is: %D. The time execution %T" >> time.info
mpirun -np 40 /home/gengzi/apps/qe-6.5/bin/pw.x -nk 1 -in ./input &> ./out
mpirun -np 40 /home/gengzi/apps/qe-6.5/bin/pw2bgw.x -in ./pp_in &> ./pp_out
rm ./*.wfc*
rm ./*.igk*
date "+02 Today's date is: %D. The time finish %T" >> time.info
```

## BerkeleyGW 计算

### 5-epsilon

- epsilon.inp

```
epsilon_cutoff 10      ! 不需要修改
number_bands 79        ! 取2-wfn/aa中能带的最大值
band_occupation 8*1 71*0 ! 价带*1, 导带*0
cell_slab_truncation

begin qpoints ! 来源于WFN.out, 最后加1列, 第一行为1, 其余都是0
0.001000000 0.000000000 0.000000000 1.0 1 ! 第一个数改为0.001
0.000000000 0.200000000 0.000000000 1.0 0
0.000000000 0.400000000 0.000000000 1.0 0
0.000000000 0.600000000 0.000000000 1.0 0
0.000000000 0.800000000 0.000000000 1.0 0
0.200000000 0.000000000 0.000000000 1.0 0
0.200000000 0.200000000 0.000000000 1.0 0
0.200000000 0.400000000 0.000000000 1.0 0
0.200000000 0.600000000 0.000000000 1.0 0
0.200000000 0.800000000 0.000000000 1.0 0
0.400000000 0.000000000 0.000000000 1.0 0
0.400000000 0.200000000 0.000000000 1.0 0
0.400000000 0.400000000 0.000000000 1.0 0
0.400000000 0.600000000 0.000000000 1.0 0
0.400000000 0.800000000 0.000000000 1.0 0
0.600000000 0.000000000 0.000000000 1.0 0
0.600000000 0.200000000 0.000000000 1.0 0
0.600000000 0.400000000 0.000000000 1.0 0
0.600000000 0.600000000 0.000000000 1.0 0
0.600000000 0.800000000 0.000000000 1.0 0
0.800000000 0.000000000 0.000000000 1.0 0
0.800000000 0.200000000 0.000000000 1.0 0
0.800000000 0.400000000 0.000000000 1.0 0
0.800000000 0.600000000 0.000000000 1.0 0
0.800000000 0.800000000 0.000000000 1.0 0
end
```

- 提交任务脚本

```
#!/bin/sh
#PBS -N hey
#PBS -l nodes=1:ppn=40,walltime=500:00:00
###PBS -q workq

cd $PBS_O_WORKDIR
source /home/gengzi/apps/intel/19/mkl/bin/mklvars.sh intel64
export PATH=$PATH:/home/gengzi/apps/openmpi404/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/gengzi/apps/openmpi404/lib

date "+01 Today's date is: %D. The time execution %T" >> time.info
mpirun -np 40 /home/gengzi/apps/BerkeleyGW-2.1/bin/epsilon.cplx.x &> ./OUT.eps
date "+02 Today's date is: %D. The time finish %T" >> time.info
```

6-sigma

- sigma.inp

```
screened_coulomb_cutoff 10
bare_coulomb_cutoff 70

number_bands 79
band_occupation 8*1 71*0
#frequency_dependence 1
#no_symmetries_q_grid
band_index_min 4 ! 取为2-wfn/pp_in中的vxc_diag_nmin值
band_index_max 13 ! 取为2-wfn/pp_in中的vxc_diag_nmax值
cell_slab_truncation

screening_semiconductor
begin kpoints ! k点来源于WFN.out, 最后加一列, 第一行为1, 其余都是0
0.000000000 0.000000000 0.000000000 1.0 1
0.000000000 0.200000000 0.000000000 1.0 0
0.000000000 0.400000000 0.000000000 1.0 0
0.000000000 0.600000000 0.000000000 1.0 0
0.000000000 0.800000000 0.000000000 1.0 0
0.200000000 0.000000000 0.000000000 1.0 0
0.200000000 0.200000000 0.000000000 1.0 0
0.200000000 0.400000000 0.000000000 1.0 0
0.200000000 0.600000000 0.000000000 1.0 0
0.200000000 0.800000000 0.000000000 1.0 0
0.400000000 0.000000000 0.000000000 1.0 0
0.400000000 0.200000000 0.000000000 1.0 0
0.400000000 0.400000000 0.000000000 1.0 0
0.400000000 0.600000000 0.000000000 1.0 0
0.400000000 0.800000000 0.000000000 1.0 0
0.600000000 0.000000000 0.000000000 1.0 0
```

(下页继续)

(续上页)

```

0.600000000 0.200000000 0.000000000 1.0 0
0.600000000 0.400000000 0.000000000 1.0 0
0.600000000 0.600000000 0.000000000 1.0 0
0.600000000 0.800000000 0.000000000 1.0 0
0.800000000 0.000000000 0.000000000 1.0 0
0.800000000 0.200000000 0.000000000 1.0 0
0.800000000 0.400000000 0.000000000 1.0 0
0.800000000 0.600000000 0.000000000 1.0 0
0.800000000 0.800000000 0.000000000 1.0 0
end

```

- 提交任务脚本

```

#!/bin/sh
#PBS -N hey
#PBS -l nodes=1:ppn=40,walltime=500:00:00
##PBS -q workq

cd $PBS_O_WORKDIR
source /home/gengzi/apps/intel/19/mkl/bin/mklvars.sh intel64
export PATH=$PATH:/home/gengzi/apps/openmpi404/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/gengzi/apps/openmpi404/lib

date "+01 Today's date is: %D. The time execution %T" >> time.info
mpirun -np 40 /home/gengzi/apps/BerkeleyGW-2.1/bin/sigma.cplx.x &> ./OUT.eps

python eqp.py eqp1 sigma_hp.log ../8-absorption/eqp_co.dat
python eqp.py eqp1 sigma_hp.log ../9-inteqp/eqp_co.dat

date "+02 Today's date is: %D. The time finish %T" >> time.info

```

注解： eqp.py 脚本位于安装文件 bin 文件夹中，使用 python3 环境

## 7-kernel

- kernel.inp

```

number_val_bands 5 ! 画图时包含的价带数
number_cond_bands 5 ! 画图时包含的导带数

screened_coulomb_cutoff 10
bare_coulomb_cutoff 70

use_symmetries_coarse_grid

```

(下页继续)

(续上页)

```
screening_semiconductor  
cell_slab_truncation
```

- 提交任务脚本

```
#!/bin/sh  
#PBS -N hey  
#PBS -l nodes=1:ppn=40,walltime=500:00:00  
##PBS -q workq  
  
cd $PBS_O_WORKDIR  
source /home/gengzi/apps/intel/19/mkl/bin/mklvars.sh intel64  
export PATH=$PATH:/home/gengzi/apps/openmpi404/bin  
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/gengzi/apps/openmpi404/lib  
  
date "+01 Today's date is: %D. The time execution %T" >> time.info  
mpirun -np 40 /home/gengzi/apps/BerkeleyGW-2.1/bin/kernel.cplx.x &> ./OUT.eps  
date "+02 Today's date is: %D. The time finish %T" >> time.info
```

## 8-absorption

- absorption.inp

```
number_val_bands_coarse 5 ! 画图时包含的价带数  
number_val_bands_fine 5 ! 画图时包含的价带数  
number_cond_bands_coarse 5 ! 画图时包含的导带数  
number_cond_bands_fine 5 ! 画图时包含的导带数  
  
use_symmetries_fine_grid  
use_symmetries_shifted_grid  
use_symmetries_coarse_grid  
diagonalization  
screening_semiconductor  
cell_slab_truncation  
#use_velocity  
  
eqp_co_corrections  
#q_shift 0.0006 0.0006 0.000  
use_momentum  
polarization 1.0 0.0 0.0  
  
energy_resolution 0.05  
gaussian_broadening  
#write_eigenvectors 10
```

- 提交任务脚本

```
#!/bin/sh
#PBS -N hey
#PBS -l nodes=1:ppn=40,walltime=500:00:00
##PBS -q workq

cd $PBS_O_WORKDIR
source /home/gengzi/apps/intel/19/mkl/bin/mklvars.sh intel64
export PATH=$PATH:/home/gengzi/apps/openmpi404/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/gengzi/apps/openmpi404/lib

date "+01 Today's date is: %D. The time execution %T" >> time.info
mpirun -np 40 /home/gengzi/apps/BerkeleyGW-2.1/bin/absorption.cplx.x &> ./OUT.eps
date "+02 Today's date is: %D. The time finish %T" >> time.info
```

## 9-inteqp

- inteqp.inp

```
number_val_bands_coarse 5 ! 画图时包含的价带数
number_val_bands_fine 5 ! 画图时包含的价带数
number_cond_bands_coarse 5 ! 画图时包含的导带数
number_cond_bands_fine 5 ! 画图时包含的导带数
```

```
#use_symmetries_fine_grid
#no_symmetries_shifted_grid
#use_symmetries_coarse_grid
use_momentum
#comm_mpi
```

- 提交任务脚本

```
#!/bin/sh
#PBS -N hey
#PBS -l nodes=1:ppn=40,walltime=500:00:00
##PBS -q workq

cd $PBS_O_WORKDIR
source /home/gengzi/apps/intel/19/mkl/bin/mklvars.sh intel64
export PATH=$PATH:/home/gengzi/apps/openmpi404/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/gengzi/apps/openmpi404/lib

date "+01 Today's date is: %D. The time execution %T" >> time.info
mpirun -np 40 /home/gengzi/apps/BerkeleyGW-2.1/bin/inteqp.cplx.x &> ./OUT.inteqp
date "+02 Today's date is: %D. The time finish %T" >> time.info
```

- 提交任务完成计算，得到 ‘bandstructure.dat’ 文件

## 数据处理

- kline.py (刘仕明开发)
- klable.txt

```
2.5124282300    0.0000000000    0.0000000000 !晶格常数
-1.2562141150    2.1758266724    0.0000000000 !晶格常数
0.0000000000    0.0000000000    7.7072650000 !晶格常数
4 ! 能带路径设置
0    0    0 50
0.5    0    0 50
0.3333 0.3333 0 50
0    0    0 0
```

- bandstructure.dat
- 运行脚本

```
python3 kline.py
```

- 得到 kline.dat 文件，画图即可

## 4.3 Quantum-ESPRESSO

### 4.4 EPW

#### 4.4.1 NSCF Kpoints

- Generate by Python script

```
#!/usr/bin/env python3

import numpy as np
f = open("./nscf.txt", "w")

X = 6
Y = 6
Z = 6
print("K_POINTS crystal", file = f)
print(X * Y * Z, file = f)

for ii in np.arange(0,1.0,1.0/X):
    for jj in np.arange(0,1.0,1.0/Y):
        for kk in np.arange(0,1.0,1.0/Z):
            # print('%10f' %(ii), ' ', '%10f' %(jj), ' ', '%10f' %(kk), ' ', '%10f' %(1.0 / (X *
            ↪Y * Z)))
```

(下页继续)



(续上页)

```
print('%0.10f' %(ii), ' ', '%0.10f' %(jj), ' ', '%0.10f' %(kk), ' ', '%0.10f' %(1.0 / (X * Y_
→* Z)), file = f)
f.close()
```

- Generate by kmesh.pl script, which can be found in wannier90-x.y.z/utility/kmesh.pl

```
./kmesh.pl 3 3 3 >> nscf.txt
```

## 4.5 pymatgen

### 4.5.1 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')
```

## 4.6 Gnuplot

### 4.6.1 希腊字母表

- 用法：`{/symbol De}` 结果： $\Delta$

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	omicron

## 4.7 Atomic Simulation Environment (ASE)

### 4.7.1 Install

```
pip install --upgrade --user ase
pip install --upgrade --user ase[test]
```

## CHAPTER 5

---

database

---

### 5.1 无机材料数据库

#### 5.1.1 三维材料数据库

1. Materials Project<sup>29</sup>
2. ICSD<sup>30</sup>

---

<sup>29</sup> <https://www.materialsproject.org/>

<sup>30</sup> <https://icsd.fiz-karlsruhe.de/search/>



## CHAPTER 6

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`