UTF8gbsn

DS-PAW Documentation

Hongzhiwei Technology (Shanghai) Co., Ltd.

2023 年 04 月 06 日

目录:


铁电
性质
过渡态
分子
动力学
压电
性质
结构
弛豫
...
电子
结构
力学
性质
杂化
泛函
DS-PAW
强关联
计算
磁性
计算
声子
计算
介电
性质
光学
性质
能带反
折叠
范德瓦
尔斯修
正
自旋轨
道耦合
Bader
电荷

CHAPTER 1

软件简介

DS-PAW 是 Device Studio 平台下一款第一性原理密度泛函计算程序，使用平面波作为基函数组，使用投影缀加平面波方法构造赝势。本程序能广泛应用于材料科学领域，开展例如金属、半导体、绝缘体、表面、磁性、非磁性、锂电等材料的计算研究；能够精确预测材料的电子分布；能够进行原子几何结构优化等多种功能的计算。本程序性能稳定，在 intel 芯片及国产海光芯片下经过百万案例的内部测试，包括各项功能及并行效率。

1.1 install 安装说明

备注:

1. DS-PAW 使用 cmake 编译，要求 cmake 版本 ≥2.8
2. 建议使用 Intel 编译器编译

1.1.1 Intel 编译器编译 MKL 版本 (推荐使用)

```
cd DS-PAW
mkdir build
cd build

cmake .. -DCMAKE_CXX_COMPILER=icc -DCMAKE_INSTALL_PREFIX=install_path/ -DDSPAW_USE_
↪MKL=ON

make -jN #N为编译使用的核数
make install
```

备注:

1. Intel 编译器版本要 ≥2017
2. **CMAKE_CXX_COMPILER** 用于指定 c++ 编译器位置，**CMAKE_INSTALL_PREFIX** 用于指定 DS-PAW 的安装目录
3. 若未成功查找到 MPI 编译器，可设置 **-DMPI_CXX_COMPILER=mpiicc** 来指定 Intel MPI 编译器位置

1.1.2 Gcc 编译器编译 openblas 版本

```
cd DS-PAW
mkdir build
cd build

cmake .. -DCMAKE_CXX_COMPILER=icc -DCMAKE_INSTALL_PREFIX=install_path/ -DOPENBLAS_
↪PATH=openblas_path/

make -jN #N为编译使用的核数
make install
```

备注:

1. gcc 编译器版本要 ≥4.8
2. 系统中需存在 MPI 编译器，如 openmpi、mpich 等
3. **CMAKE_CXX_COMPILER** 用于指定 c++ 编译器位置，**CMAKE_INSTALL_PREFIX** 用于指定 DS-PAW 的安装目录，**OPENBLAS_PATH** 用于指定 openblas 目录、访问 openblas 文档
4. 若未成功查找到 MPI 编译器，可设置 **-DMPI_CXX_COMPILER=mpicxx** 来指定 MPI 编译器位置

1.2 command 命令说明

1.2.1 list 命令列表

- *lic*
- *info*
- *mpi*
- *mpiargs*
- *pob*

1.2.2 detail 命令详细描述

命令名称: *-lic*

使用方法： *-lic* 用于生成序列号，在 DS-PAW 安装目录下执行命令：*DS-PAW -lic* 即可得到 LicenseNumber.txt 文件，该文件用于 license 的申请

命令名称: *-info*

使用方法： *-info* 用于查看软件版权信息，执行命令：*DS-PAW -info*

命令名称: *-mpi xxx*

使用方法： *-mpi* 用于指定 mpi 执行程序的位置，如：*-mpi mpirun*

命令名称: *-mpiargs xxx*

使用方法： *-mpiargs* 用于指定 mpi 运行参数，如：*-mpiargs "-np 16"*

命令名称: *-pob*

使用方法： *-pob* 用于并行计算时合理分配核数加快运行速度，为 *parallel over band* 的简写，可在提交命令中添加此关键词

1.3 run 程序运行

1.3.1 submit 命令提交运行

设置环境变量：

```
export PATH={DS-PAW INSTALLPATH}/bin:$PATH
```

串行执行：

```
DS-PAW input.in
```

并行执行：

```
DS-PAW -mpi mpirun -mpiargs "-np 16" input.in -pob
```

1.3.2 script 脚本提交运行

若使用排队系统（例如 PBS、slurm 等）提交任务，只要配置完成相应的 `.pbs` 或 `.slurm` 脚本，之后使用 *qsub xx.pbs* 或 *sbatch xx.slurm* 提交任务即可。

CHAPTER 2

快速入门

本章将介绍 DS-PAW 的各种功能的基本使用，具体包括：**结构弛豫计算、自洽计算、能带（投影能带）计算、态密度（投影态密度）计算、自洽中直接计算能带和态密度、势函数计算、电子局域密度计算、部分电荷密度计算、杂化泛函计算、范德瓦尔斯修正计算、偶极修正计算、DFT+U 计算、背景电荷计算、光学性质计算、频率计算、弹性常数计算、过渡态计算、声子谱计算、自旋轨道耦合计算、分子动力学模拟、外加电场计算、铁电计算、bader 电荷计算、能带反折叠计算、介电常数计算、压电张量计算、固定基矢弛豫计算、声子热力学性质计算、solid state neb 计算**；在 DS-PAW 软件中大致的可以将参数分类几类：与物理结构相关的参数、与计算性质相关的参数、与计算精度相关的参数、与收敛相关的参数，这些参数中有很多都是有默认值的，且绝大多数参数不会影响计算结果；因此本章只设置了一些特别重要的参数，同时将对这些参数进行简单的讲解。

2.1 relax 结构弛豫

在密度泛函理论（DFT）中结构弛豫指的改变初始的结构的晶胞及原子位置从而优化得到总能量的局域最小值。通过结构弛豫计算，可以减少在每个原子上的受力，从而得到较为稳定的结构（一定程度上，可以通过计算声子谱或者频率来验证结构的稳定性）。使用建模软件搭建的结构一般情况下原子受力都会比较大，且即使是其他 DFT 软件优化过的结构，在另外一个 DFT 计算软件中原子受力也不一定是最小的，因此一般情况下在计算某个结构的具体性质之前需要进行结构弛豫计算。

2.1.1 Si 原子结构弛豫输入文件

输入文件包含参数文件 `relax.in` 和结构文件 `structure.as` ，`relax.in` 如下：

```
# task type
task = relax
#system related
```

(续下页)

(接上页)

```
sys.structure = structure.as
sys.symmetry = false
sys.functional = LDA
sys.spin = none
#scf related
cal.methods = 2
cal.smearing = 1
cal.ksamping = G
cal.kpoints = [10, 10, 10]
cal.cutoffFactor = 1.5

#relax related
relax.max =  60
relax.freedom = atom
relax.convergence =  0.05
relax.methods = CG

io.outStep = 1
io.wave = false
io.charge = false
```

relax.in 文件大致可以分为 4 部分参数：

第一部分指定计算类型，计算类型中只有 task 一个参数：

- `task`： 设置计算类型，本次计算为 relax 结构弛豫；

第二部分指定系统相关参数，系统相关参数以 sys. 开头，一般是一些与体系的结构、泛函、磁性、对称性相关的参数：

- `sys.structure`： 指定体系的结构文件，DS-PAW 支持 *.as* 和 *.json* 的结构文件格式，*.as* 文件可以直接使用 Device Studio 软件生成，也可以自己手动生成；
- `sys.symmetry`： 设置 DS-PAW 计算时是否需要使用对称性；
- `sys.functional`： 设置泛函，目前程序支持 **LDA** 及 **PBE** 泛函；
- `sys.spin`： 设置体系的磁性，由于 **Si** 没有磁性，因此将 sys.spin 设置为 **none** ；

第三部分指定计算相关的参数，这类参数以 cal. 开头：

- `cal.methods`：设置自洽电子步优化方法，2 表示使用 **Residual minimization** 方法；
- `cal.smearing`： 设置每个波函数的部分占有数方法，1 表示使用 **Gaussian smearing** 方法；
- `cal.ksamping`： 自动生成布里渊区 **k 点**网格方法，G 表示使用 **Gamma centered** 方法；
- `cal.kpoints`： 设置布里渊区 **k 点**网格取样大小，一般 K 点的大小需要根据体系晶格的大小和体系的周期性来设置，通常建议在具有周期性的方向上晶格矢量与 K 点的乘积不小于 40，在不具有周期性的方向上将 **K 点**设置为 1 即可；

第四部分指定结构弛豫相关参数，例如结构弛豫的方法、结构弛豫的类型、结构弛豫的精度等相关参数，结构弛豫指的是优化原子位置得到总能量的局域最小值的结构，一般也称之为离子步优化；

- `relax.max`：设置结构弛豫时，最大的离子步步数；
- `relax.freedom`： 设置结构弛豫的自由度，atom 表示只弛豫原子；all 表示弛豫晶格常数和原子；volume 表示只弛豫晶格；
- `relax.convergence`： 设置结构弛豫时，原子受力的收敛判据；
- `relax.methods`： 设置结构弛豫的方法，CG 表示共轭梯度法；

structure.as 文件参考如下：

```
Total number of atoms
2
Lattice
0.00 2.75 2.75
2.75 0.00 2.75
2.75 2.75 0.00
Direct
Si -0.115000000 -0.125000000 -0.125000000
Si 0.125000000 0.125000000 0.125000000
```

structure.as 文件结构固定，必须严格对照每一行写入相应信息:

- 第一行为固定提示行
- 第二行为总的原子个数
- 第三行为固定提示行
- 第四至六行为晶胞信息
- 第七行为原子坐标的形式，可选值为 **Direct** 和 **Cartesian** （全拼且首字母必须为大写）
- 第八行开始写入原子坐标信息，每一行的开头必须标注坐标所描述的原子的名称

由于本案例为结构弛豫计算，因此手动的将第一个 `Si` 原子的 X 坐标从 **-0.125** 改为 **-0.115** ，这样能够更好的体现结构弛豫的效果。

备注:

1. **如果想要固定原子，则在第 7 行加入 Fix_x Fix_y Fix_z 标签，然后在每个原子对应的位置加入 F 或 T，F 表示不固定，T 表示固定**。

```
Direct Fix_x Fix_y Fix_z
Si -0.115000000 -0.125000000 -0.125000000 F F F
Si 0.125000000 0.125000000 0.125000000 T T T
```

2.1.2 run 程序运行

准备好输入文件之后，将 `relax.in` 和 `structure.as` 文件上传到安装了 DS-PAW 的环境上，这里将以 linux 环境为例子进行介绍；

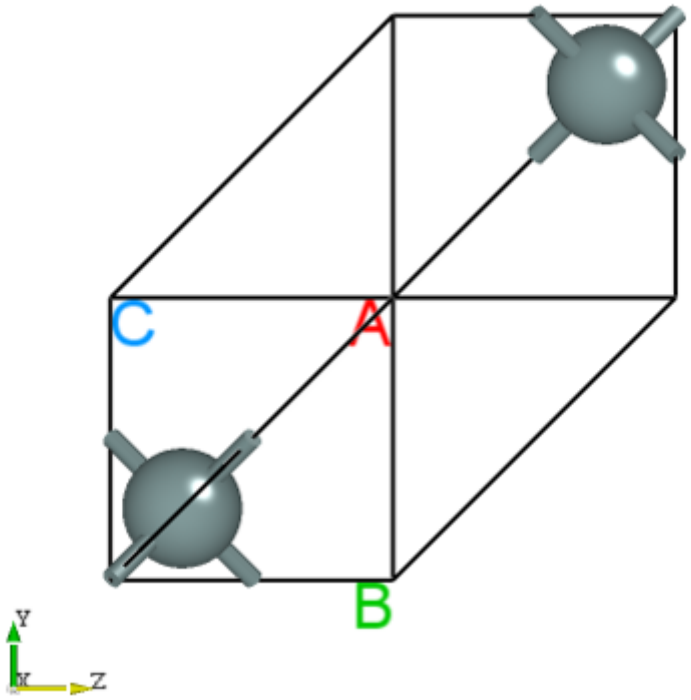
在无图形界面的 linux 环境下运行软件的方式与在 windows 的环境下运行程序会有很大的区别，在 linux 下需要通过命令行的方式来运行程序；一般情况下需要先加载一下环境变量，通常会将需要用的环境变量写入文本或者 **.bashrc** ，通过 **source** 的命令来加载环境；环境加载完成之后，运行 *DS-PAW relax.in* ，此时使用单机版的 DS-PAW 进行计算；如需并行计算则需运行 *DS-PAW -mpi mpirun -mpiargs "-n 2" relax.in* ， **-mpi** 指定 mpirun 的名称，**-mpiargs** 指定 mpirun 后面的参数，用双引号括起来。如果你需要使用排队系统（例如 PBS、slurm 等）提交任务，只要配置完成相应的 `.pbs` 或 `.slurm` 脚本，之后使用 *qsub xx.pbs* 或 *sbatch xx.slurm* 提交任务即可。

2.1.3 analysis 计算结果分析

根据上述的输入文件，计算完成之后将会得到 `DS-PAW.log` 、`relax.json` 、`system.json` 、`paw_tmp/relax.tmp` 这 4 个文件:

- `DS-PAW.log`： DS-PAW 计算之后得到的日志文件；
- `relax.json`：结构弛豫完成之后的结构文件，里面保存着每一个离子步完成之后的结构文件，可以将 `relax.json` 文件导入 Device Studio 软件中进行显示；生成的 `relax.json` 文件也可以直接用于之后自洽计算、能带、态密度等一系列计算中，只需在计算中将 **sys.structure 参数**指定当前 `relax.json` 文件即可；
- `system.json`: `system.json` 文件中保存着最后一个离子步之后的计算结果，包含结构、磁矩、能量本征值、占据态、K 点信息、原子受力、stress 等相应的信息；

将 *relax.json* 拖入 Device Studio 查看结构如下所示:

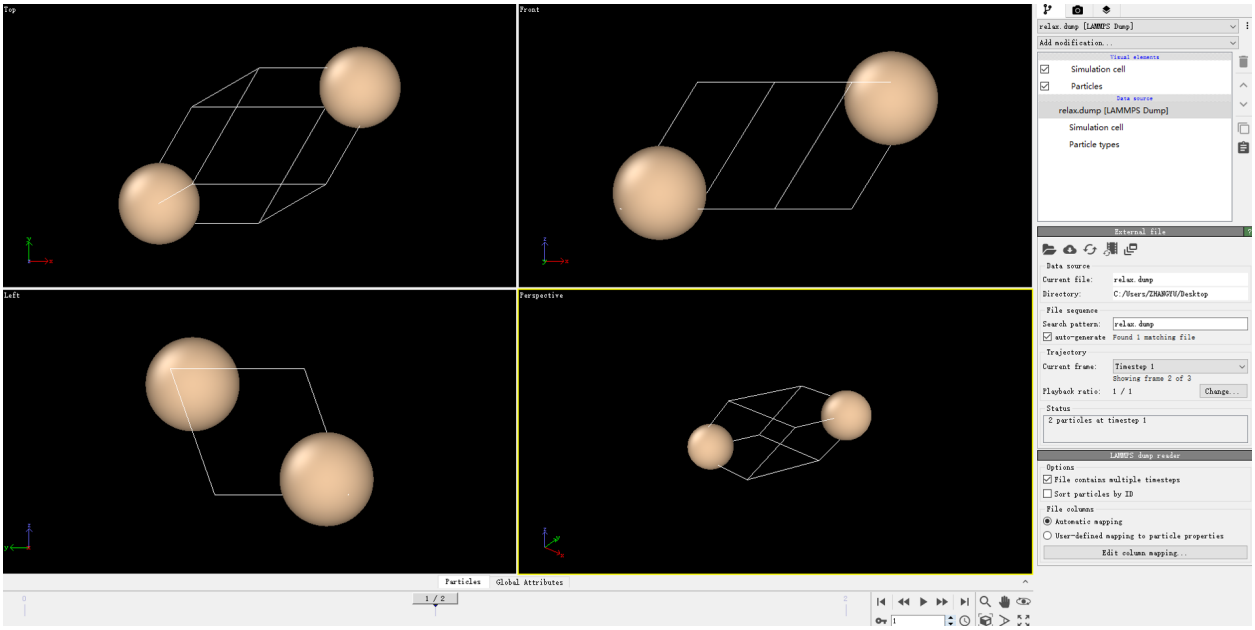


除查看最终构型，DS 可以以 gif 动画的形式展示弛豫过程中结构变化的过程，在面板依次选择 **Simulator-->DS-PAW-->Analysis Plot-->relax.json-->Trajectory** 即可，以下为截取其中一帧的图像:

使用文本编辑器或者在线显示 **json 格式**可打开 `realx.json` 文件，更简单的方法是查看 paw_tmp 文件夹下的 `relax.tmp` 文件，可得弛豫结束后的结构数据如下:

```
Total number of atoms
Lattice
           0         2.75         2.75
        2.75            0         2.75
```

(续下页)



(接上页)

```
        2.75          2.75             0
Direct
Si    0.88019174    0.87481749    0.87481752
Si    0.12980826    0.12518251    0.12518248
```

本次结构弛豫计算进行了 3 个离子步的计算，弛豫结束的构型中，手动移动的第一个 Si 原子在 x 方向上的坐标在弛豫计算之后完成校正。

备注:

1. **单机版 DS-PAW 运行命令为软件名 + 输入文件名，如果你的输入文件名为 abc.in 那么只要执行 DS-PAW abc.in 即可。**
2. **该弛豫计算的收敛判据选取为原子受力，若以能量作为收敛判据，可参考 relax.convergence = 1.0e-4 进行设置。**

2.2 scf 自洽计算

自洽计算能够得到特定晶体的电荷密度和波函数，有了电荷密度之后才能有计算该体系的能带、态密度等电子结构性质。特别需要注意是：自洽与能带、态密度等电子结构性质计算是有先后顺序的，必须先进行自洽计算得到电荷密度才能进一步计算能带、态密度等电子结构性质。

2.2.1 Si 原子自洽计算之准备输入文件

输入文件包含参数文件 scf.in 和结构文件 structure.as ， scf.in 如下：

```
# task type
task = scf
#system related
sys.structure = structure.as
sys.symmetry = false
sys.functional = LDA
sys.spin = none
#scf related
cal.methods = 2
cal.smearing = 1
cal.ksamping = G
cal.kpoints = [10, 10, 10]
cal.cutoffFactor = 1.5
#outputs
io.charge = true
io.wave = true
```

scf.in 输入参数介绍：

可以看到 scf.in 的输入文件中很多参数与结构弛豫的参数名是一致的，其设置方法也是一致的，这里只着重介绍一些前面没设置过或设置有些不同的参数：

- task： 本次计算为 **scf** 自洽计算，因此将 task 设置为 **scf** ；
- cal.cutoffFactor：表示截断能参数 cal.cutoff 的系数，在 DS-PAW 程序中，如果 cal.cutoff 参数缺失，程序将根据元素的截断能设置默认的平面波截断，cal.cutoffFactor 参数就是在 cal.cutoff 上设置乘以一个系数；
- io.charge： 当 io.charge 设置为 **true** 时，表示计算完成之后输出电荷密度的二进制文件 rho.bin 和 json 文件 rho.json，二进制 rho.bin 文件用于后续的后处理计算，例如能带、态密度等，rho.json 文件用于显示；
- io.wave： 当 io.wave 设置为 **true** 时，表示计算完成之后输出波函数的二进制文件 wave.bin，用户可以在后续的后处理计算中选择是否使用 wave.bin 开始计算；

structure.as 文件参考如下：

```
Total number of atoms
2
Lattice
0.00 2.75 2.75
2.75 0.00 2.75
2.75 2.75 0.00
Direct
Si -0.125000000 -0.125000000 -0.125000000
Si 0.125000000 0.125000000 0.125000000
```

由于本案例为自洽计算只是为了计算得到特定体系的电子结构，因此不需要手动改变原子位置；

备注:

1. **io 类参数只在结构弛豫和自洽中起作用，例如 io.charge 在其他计算情况下将不会生成 rho.bin 或 rho.json 文件；**

2. **在结构弛豫和自洽中，还能够保存 elf、potential 的数据，只需要将 io.elf 和 io.potential 设置为 true 即可；**
3. **如果用户想要使用自己优化的结构，只需在计算中将 sys.structure 参数指定绝对路径或相对路径下的 relax.json，也可以将 relax.json 文件复制到本次计算的目录中，设置 sys.structure=./relax.json 即可；**
4. **带自旋体系的计算案例详解第二章的 NiO 案例。**
5. **计算时如需给体系添加背景电荷，可直接设置 sys.electron 参数，该参数指定价电子的总数。**

2.2.2 run 程序运行

准备好输入文件 `scf.in` 和 `structure.as` 后，将文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 *DS-PAW scf.in* 。

2.2.3 analysis 计算结果分析

根据上述的输入文件，计算完成之后将会得到 `DS-PAW.log` 、`system.json` 、`rho.bin` 、`rho.json` 、`wave.bin` 这 5 个文件。

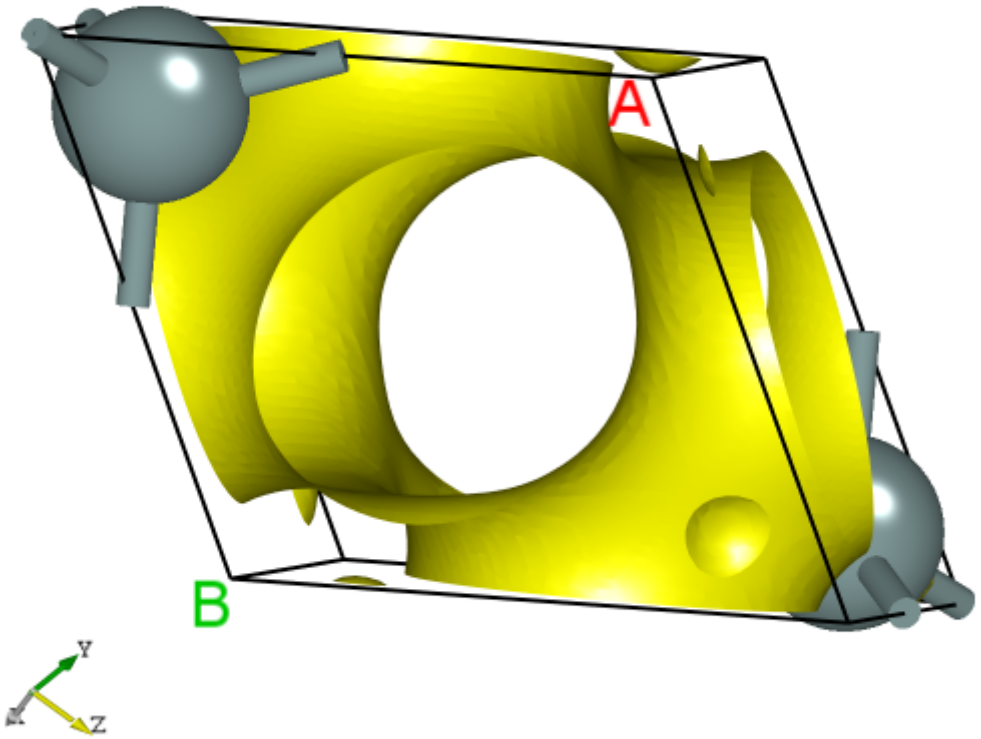
- `rho.bin`： 电荷密度的二进制文件，用于后续的后处理计算；
- `rho.json`： 电荷密度的 **json** 格式文件，用来显示电荷密度的结果；
- `wave.bin`： 波函数的二进制文件，用于后续计算；

使用 **Device Studio** 可直接对 `rho.json` 文件处理出图，其操作步骤为：Simulator-->DS-PAW-->Analysis Plot，选择 `rho.json` 即可，可根据作图要求自定义设置面板参数，处理可得一维、二维、三维电荷密度图，其中三维图如下所示：

另可使用 python 脚本将 `rho.json` 格式的转化成 *VESTA* 软件支持的格式，具体操作见 **辅助工具使用教程**部分。

2.3 band 能带计算

本节将从自洽出发介绍如何使用 DS-PAW 计算能带和投影能带。以 Si 体系为例进行自洽计算（见 2.2 节），自洽完成之后准备能带计算和投影能带计算，并对能带和投影能带作图分别进行分析。



2.3.1 Si 能带计算输入文件

输入文件包含参数文件 band.in 、结构文件 structure.as 和上次自洽计算得到的二进制电荷密度文件 rho.bin ， band.in 如下：

```
# task type
task = band
#system related
sys.structure = structure.as
sys.symmetry = false
sys.functional = LDA
sys.spin = none
#scf related
cal.iniCharge = ./rho.bin
cal.methods = 2
cal.smearing = 1
cal.ksamping = G
cal.kpoints = [10, 10, 10]
cal.cutoffFactor = 1.5
#band related
band.kpointsLabel= [G,X,W,K,G,L]
band.kpointsCoord= [0, 0, 0, 0.5, 0, 0.5, 0.5, 0.25, 0.75, 0.375, 0.375, 0.75, 0, 0,␣
↪0, 0.5, 0.5, 0.5]
band.kpointsNumber= [30, 30, 30, 30, 30]
```

band.in 输入参数介绍：

在能带计算中可以尽量保留 **sys.** 和 **cal.** 的参数到 band.in 中，之后设置能带计算特有的参数即可：

- task： 本次计算为 band 能带计算，设置 task 为 **band** ；
- cal.iniCharge： 表示读取电荷密度二进制文件，支持绝对路径及相对路径，这里./表示当前路径下的 rho.bin 文件；

能带计算中新增了一部分能带相关的参数，这些参数只在能带计算中起作用：

- band.kpointsLabel：表示能带计算时高对称点标签，一个 band.kpointsCoord 对应一个 band.kpointsLabel ；
- band.kpointsCoord： 表示能带计算时高对此点的分数坐标，每三个数为一组；
- band.kpointsNumber： 表示每相邻两个高对称点的间隔；

structure.as 文件同自洽计算。（见 2.2 节）

备注:

1. **若读取电荷密度文件 rho.bin，cal.cutoffFactor 和 cal.cutoff 须和之前自洽计算的设置保持一致，否则会出现格点数据不匹配的问题**
2. **band.kpointsCoord、band.kpointsLabel 和 band.kpointsNumber 三个参数的维度需要统一**

2.3.2 run 程序运行

准备好输入文件 band.in 和 structure.as 以及 rho.bin 之后，将文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 *DS-PAW band.in* 。

2.3.3 analysis 计算结果分析

根据上述的输入文件，计算完成之后将会得到 DS-PAW.log 、band.json 这 2 个文件。

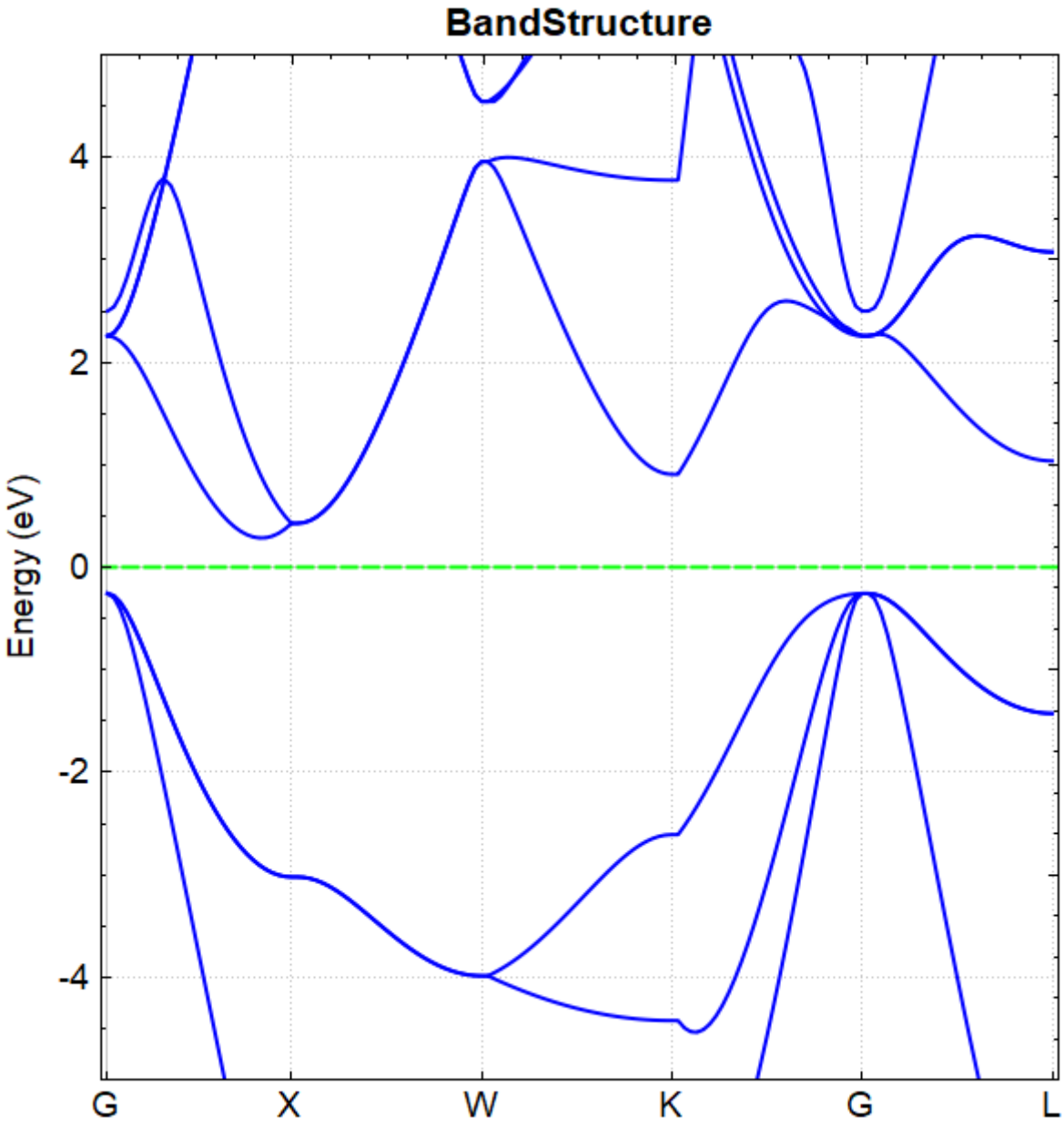
band.json： 能带计算完成之后的 **json** 数据文件；此时能带的数据也将会被保存在 band.json 中，具体的数据结构详见数据结构解析部分；

使用 **Device Studio** 可直接对 band.json 文件处理出图，其操作步骤为：Simulator-->DS-PAW-->Analysis Plot，选择 band.json 即可，可根据作图要求自定义设置面板参数，DS 处理得到的能带图如下所示：

另可使用 **python** 进行数据处理，具体操作见 **辅助工具使用教程**部分。

2.3.4 ioband 自洽中直接计算能带

在 2.2 自洽计算的输入文件中加入 io.band=true，之后将 band 相关参数写入 scf.in 中即可。输入文件如下：


BandStructure
Energy (eV)
4
2
0
-2
-4
G
X
W
K
G
L

```
# task type
task = scf
#system related
sys.structure = structure.as
sys.symmetry = false
sys.functional = LDA
sys.spin = none
#scf related
cal.methods = 2
cal.smearing = 1
cal.ksamping = G
cal.kpoints = [10, 10, 10]
cal.cutoffFactor = 1.5
#outputs
io.charge = true
io.wave = true
#band related
io.band=true
band.kpointsLabel= [G,X,W,K,G,L]
band.kpointsCoord= [0, 0, 0, 0.5, 0, 0.5, 0.5, 0.25, 0.75, 0.375, 0.375, 0.75, 0, 0,␣
↪0, 0.5, 0.5, 0.5]
band.kpointsNumber= [30, 30, 30, 30, 30]
```

此时计算出来的能带图与单独计算得到的能带图效果一致。

备注:

1. **只有在 task=scf 时 io.band=true 才能生效;**
2. **当 io.band=true 生效时，band 相关的参数都将生效;**
3. **当 io.band 生效时，不再需要设置 cal.iniCharge = ./rho.bin，此过程将在程序内部完成。**
4. **ioband.in 文件中需给出两类 k 点，cal.kpoints 给出自洽计算的 k 点，band.kpoints 相关参数给出能带计算的 k 点，两部分 k 点缺一不可。**

2.4 pband 投影能带计算

投影能带是指在能带计算过程中将每条能带每个 K 点上的能量展开为每个原子及其轨道的贡献。

2.4.1 Si 投影能带计算输入文件

投影能带的输入文件包含参数文件 pw_band.in 结构文件 structure.as 和上次自洽计算得到的二进制电荷密度文件 rho.bin，pw_band.in 如下：

```
# task type
task = band
#system related
sys.structure = structure.as
sys.symmetry = false
sys.functional = LDA
sys.spin = none
#scf related
cal.iniCharge = ./rho.bin
cal.methods = 2
cal.smearing = 1
cal.ksamping = G
cal.kpoints = [10, 10, 10]
cal.cutoffFactor = 1.5
#band related
band.kpointsLabel= [G,X,W,K,G,L]
band.kpointsCoord= [0, 0, 0, 0.5, 0, 0.5, 0.5, 0.25, 0.75, 0.375, 0.375, 0.75, 0, 0,␣
↪0, 0.5, 0.5, 0.5]
band.kpointsNumber= [30, 30, 30, 30, 30]
band.project=true
```

pw_band.in 输入参数介绍：

pw_band.in：与普通能带的区别在于在计算参数中设置了 *band.project* 参数:

band.project： 表示在能带计算中打开能带投影的开关；

2.4.2 run 程序运行

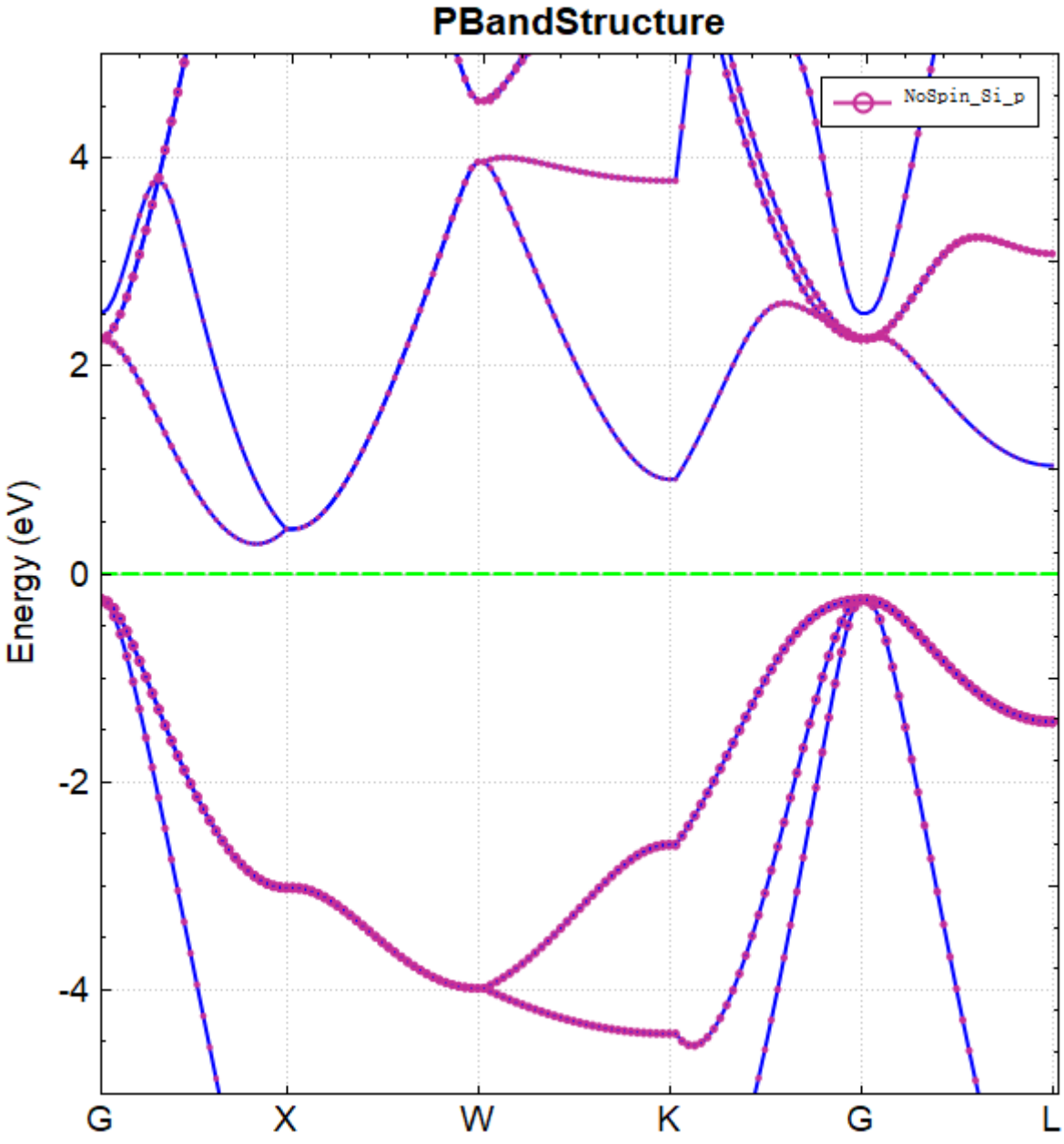
准备好输入文件 pw_band.in 和 structure.as 以及 rho.bin 之后，将文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 *DS-PAW pw_band.in* 。

2.4.3 analysis 计算结果分析

根据上述的输入文件，计算完成之后将会得到 `DS-PAW.log` 、`band.json` 这 2 个文件。

band.json： 能带计算完成之后的 **json** 数据文件；此时投影能带的数据也将会被保存在 `band.json` 中，具体的数据结构详见数据结构解析部分；

使用 **Device Studio** 可直接对 `band.json` 文件处理出图，其操作步骤为：Simulator-->DS-PAW-->Analysis Plot，选择 `band.json` 即可，可根据作图要求自定义设置面板参数。DS 处理得到的能带图如下所示：



另可使用 **python** 进行数据处理，具体操作见 **辅助工具使用教程**部分。

2.5 dos 态密度计算

本节将从自洽出发介绍如何使用 DS-PAW 计算态密度计算和投影态密度。以 Si 体系为例进行自洽计算（见 2.2 节），自洽完成之后准备态密度计算和投影态密度能带计算，并对能带和投影能带作图分别进行分析。

2.5.1 Si 态密度计算输入文件

输入文件包含参数文件 `dos.in` 结构文件 `structure.as` 和上次自洽计算得到的二进制电荷密度文件 `rho.bin` ，`dos.in` 如下：

```
# task type
task = dos
#system related
sys.structure = structure.as
sys.symmetry = false
sys.functional = LDA
sys.spin = none


#scf related
cal.iniCharge = ./rho.bin
cal.methods = 2
cal.smearing = 4
cal.ksamping = G
cal.kpoints = [20, 20, 20]
cal.cutoffFactor = 1.5
cal.sigma = 0.05

#dos related
dos.range=[-10, 10]
dos.resolution=0.05
dos.EfShift = true
```

dos.in 输入参数介绍：

在态密度计算中可以尽量保留 **sys.** 和 **cal.** 的参数到 `dos.in` 中，之后设置态密度计算特有的参数即可：

- `task`： 本次计算为 dos 态密度计算，设置 task 为 **dos** ；
- `cal.iniCharge`： 表示读取电荷密度二进制文件，支持绝对路径及相对路径，这里./表示当前路径下的 `rho.bin` 文件；
- `cal.kpoints`： 为得到较为精确的态密度计算结果，需要设置更为密的 **K** 网格，因此需要重新设置 `cal.kpoints` ，一般该参数的取值为自洽时的两倍左右；
- `cal.sigma`： 在态密度计算中为了得到比较精细的态密度，往往需要将 **sigma** 的值设置为尽可能的小；

态密度计算中新增了一部分态密度相关的参数，这些参数只在态密度计算中起作用：

- `dos.range`： 态密度计算时能量的区间；

- `dos.resolution`： 表示态密度计算能量间隔精度，态密度计算的点数即为 **dos.range** 的差值与 **dos.resolution** 的比值 +1；
- `dos.EfShift`： 当 dos.EfShift 为 **true** 时，表示将 **dos.range** 中设置的能量是相对于自洽费米能级的相对能量；

structure.as 文件同自洽计算。（见 2.2 节）

备注:

1. **需要设置 cal.iniCharge 参数的时候，cal.cutoffFactor 和 cal.cutoff 参数必须和之前自洽计算的结果一致，不然读取 rho.bin 的时候会出现格点数据不匹配的问题**。

2.5.2 run 程序运行

准备好输入文件 `dos.in` 和 `structure.as` 以及 `rho.bin` 之后，将文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 *DS-PAW dos.in* 。

2.5.3 analysis 计算结果分析

根据上述的输入文件，计算完成之后将会得到 `DS-PAW.log` 、`dos.json` 这 2 个文件。

dos.json： 态密度计算完成之后的 json 数据文件；

使用 **Device Studio** 可直接对 `dos.json` 文件处理出图，其操作步骤为：Simulator-->DS-PAW-->Analysis Plot，选择 `dos.json` 即可，可根据作图要求自定义设置面板参数。DS 处理得到的态密度图如下所示：

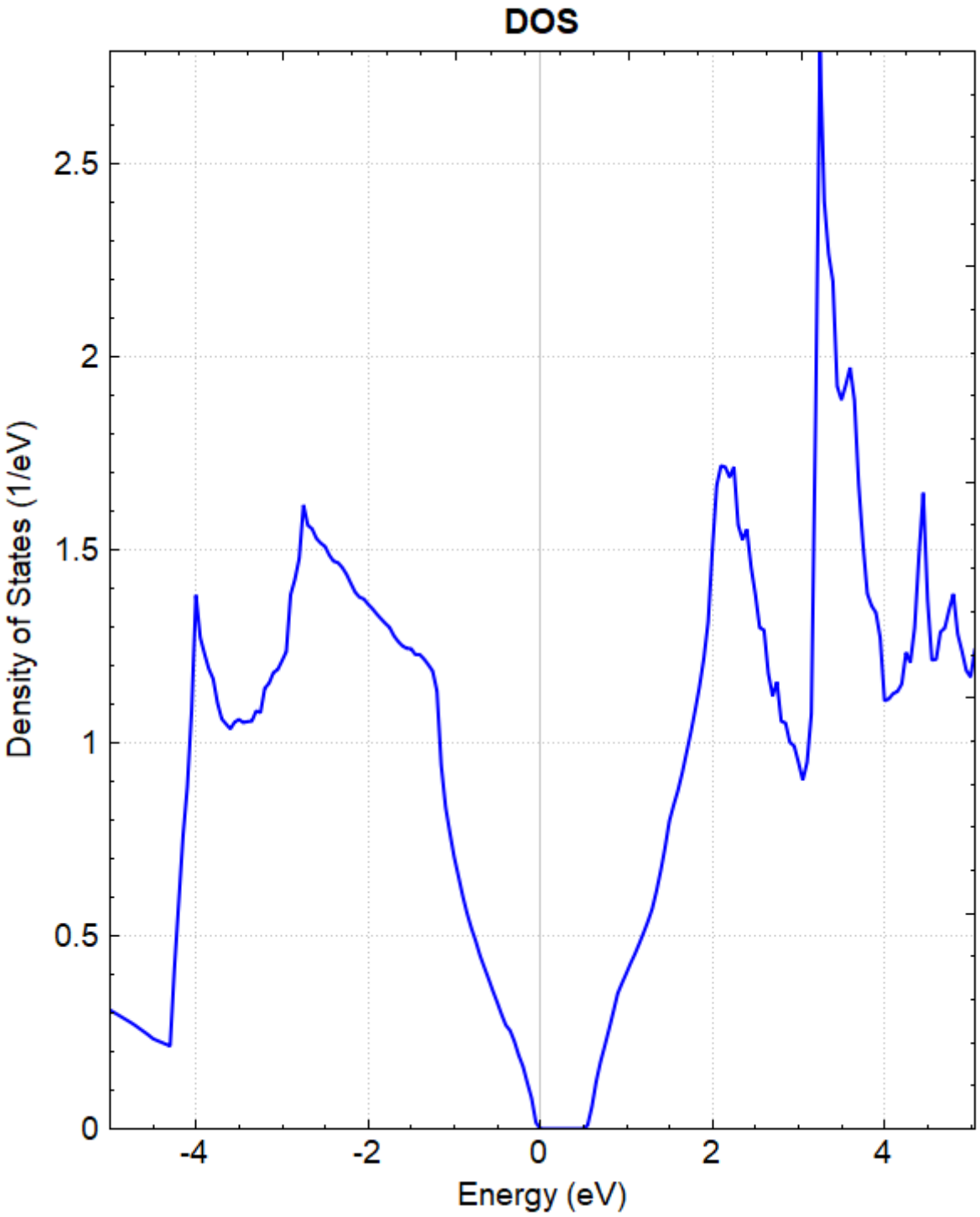
另可使用 **python** 进行数据处理，具体操作见 **辅助工具使用教程**部分。

2.5.4 iodos 自洽中直接计算态密度

在 2.2 自洽计算的输入文件中加入 io.dos=true，之后将 dos 相关参数写入 `scf.in` 中即可。输入文件如下：

```
# task type
task = scf
#system related
sys.structure = structure.as
sys.symmetry = false
sys.functional = LDA
sys.spin = none
#scf related
cal.methods = 2
cal.smearing = 1
cal.ksamping = G
cal.kpoints = [10, 10, 10]
cal.cutoffFactor = 1.5
#outputs
```

(续下页)


DOS
Density of States (1/eV)
2.5
2
1.5
1
0.5
0
-4
-2
0
2
4
Energy (eV)

(接上页)

```
io.charge = true
io.wave = true
#dos related
io.dos=true
dos.range=[-10, 10]
dos.resolution=0.05
dos.EfShift = true
```

备注:

1. **只有在 task=scf 时 io.dos=true 才能生效;**
2. **当 io.dos=true 生效时，dos 相关的参数都将生效;**
3. **当 io.dos 生效时，不再需要设置 cal.iniCharge = ./rho.bin，此过程将在程序内部完成。**
4. **iodos.in 文件中 cal.kpoints 提供自洽计算和态密度计算的 k 点，两个过程的 k 点的大小需保持一致。**

2.6 pdos 投影态密度计算

投影态密度的计算是指在态密度计算过程中将态每个能量下的态密度展开为每个原子及其轨道的态密度贡献。

2.6.1 Si 投影态密度计算输入文件

投影态密度的输入文件包含参数文件 `pdos.in` 结构文件 `structure.as` 和上次自洽计算得到的二进制电荷密度文件 `rho.bin`，`pdos.in` 如下:

```
# task type
task = dos
#system related
sys.structure = structure.as
sys.symmetry = false
sys.functional = LDA
sys.spin = none
#scf related
cal.iniCharge = ./rho.bin
cal.methods = 2
cal.smearing = 4
cal.ksamping = G
cal.kpoints = [20, 20, 20]
cal.cutoffFactor = 1.5
cal.sigma = 0.05
#dos related
dos.range=[-10, 10]
```

(续下页)

(接上页)

```
dos.resolution=0.05
dos.EfShift = true
dos.project = true
```

pdos.in 输入参数介绍：

pdos.in 与普通态密度的区别在于在计算参数中设置了 dos.project 参数：

- dos.project： 表示在态密度计算中打开态密度投影的开关；

2.6.2 run 程序运行

准备好输入文件 pdos.in 和 structure.as 以及 rho.bin 之后，将文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 *DS-PAW pdos.in* 。

2.6.3 analysis 计算结果分析

根据上述的输入文件，计算完成之后将会得到 DS-PAW.log 、dos.json 这 2 个文件。

- *dos.json*： 态密度计算完成之后的 **json** 数据文件；此时投影态密度的数据也将会被保存在 dos.json 中，具体的数据结构详见数据结构解析部分；

使用 **Device Studio** 可直接对 dos.json 文件处理出图，其操作步骤为：Simulator-->DS-PAW-->Analysis Plot，选择 dos.json 即可，可根据作图要求自定义设置面板参数。DS 处理得到的投影态密度图如下所示：

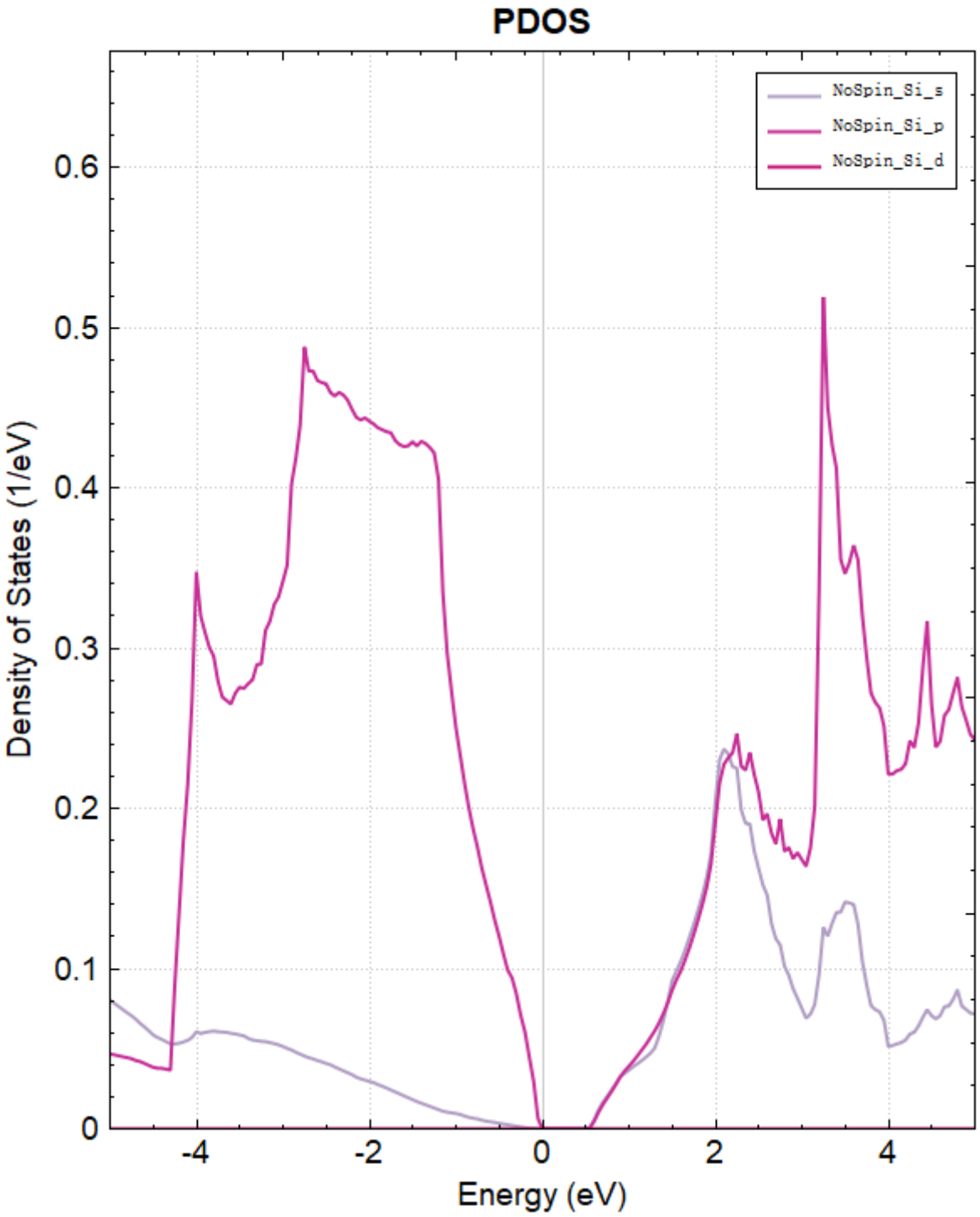
另可使用 **python** 进行数据处理，具体操作见 **辅助工具使用教程**部分。

2.7 potential 势函数计算

本节将从自洽出发介绍如何使用 DS-PAW 计算势函数。以 **Si** 体系为例进行自洽计算（见 2.2 节），自洽完成之后准备势函数输入文件，并对势函数的结果进行分析。

2.7.1 Si 势函数计算输入文件

输入文件包含参数文件 potential.in 结构文件 structure.as 和上次自洽计算得到的二进制电荷密度文件 rho.bin ，potential.in 如下：


PDOS
NoSpin_Si_s
NoSpin_Si_p
NoSpin_Si_d
Density of States (1/eV)
0
0.1
0.2
0.3
0.4
0.5
0.6
-4
-2
0
2
4
Energy (eV)

```
# task type
task = potential
#system related
sys.structure = structure.as
sys.symmetry = false
sys.functional = LDA
sys.spin = none
#scf related
cal.iniCharge = ./rho.bin
cal.methods = 2
cal.smearing = 1
cal.ksamping = G
cal.kpoints = [10, 10, 10]
cal.cutoffFactor = 1.5
#potential related
potential.type=all
```

potential.in 输入参数介绍：

在势函数计算中可以尽量保留 **sys.** 和 **cal.** 的参数到 `potential.in` 中，之后设置势函数计算特有的参数即可：

- `task`： 本次计算为 `potential` 势函数计算，设置 task 为 **potential**；
- `cal.iniCharge`： 表示读取电荷密度二进制文件，支持绝对路径及相对路径，这里./表示当前路径下的 `rho.bin` 文件；

势函数计算中新增了一个相关的参数：

- `potential.type`： 表示保存势函数的类型，当选择 **all** 的时候，势函数计算完成之后会同时保存静电势及总的局域势；

structure.as 文件同自洽计算。（见 2.2 节）

备注：

1. **需要设置 cal.iniCharge 参数的时候，cal.cutoffFactor 和 cal.cutoff 参数必须和之前自洽计算的结果一致，不然读取 rho.bin 的时候会出现格点数据不匹配的问题；**
2. **本案例是调用自洽的结果进行势函数计算，用户也可以在 task=scf 时设置 io.potential=true，此时也能保存 potential.json 文件。**
3. **如果所计算的体系需要考虑偶极修正，此时用户需要在自洽和势函数计算文件中都加入 corr.dipol = true 以及 corr.dipolDirection 参数，corr.dipol = true 表示打开偶极修正的开关，corr.dipolDirection 表示设置偶极修正的方向 a、b、c 分别表示沿着晶格矢量的 a、b、c 方向。**
4. **偶极修正的具体案例见应用案例：Au-Al 体系功函数计算案例。**

2.7.2 run 程序运行

准备好输入文件 potential.in 和 structure.as 以及 rho.bin 之后，将文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 *DS-PAW potential.in* 。

2.7.3 analysis 计算结果分析

根据上述的输入文件，计算完成之后将会得到 DS-PAW.log 、potential.json 这 2 个文件。

potential.json： 势函数计算完成之后的 **json** 数据文件；

使用 **Device Studio** 可直接对 potential.json 文件处理出图，其操作步骤为：Simulator-->DS-PAW-->Analysis Plot，选择 potential.json 即可，可根据作图要求自定义设置面板参数作一维、二维、三维的势函数曲线。DS 处理得到的势函数一维图如下所示：

另可使用 **python** 脚本将 potential.json 格式的转化成 **VESTA** 软件支持的格式，具体操作见 **辅助工具使用教程**部分。

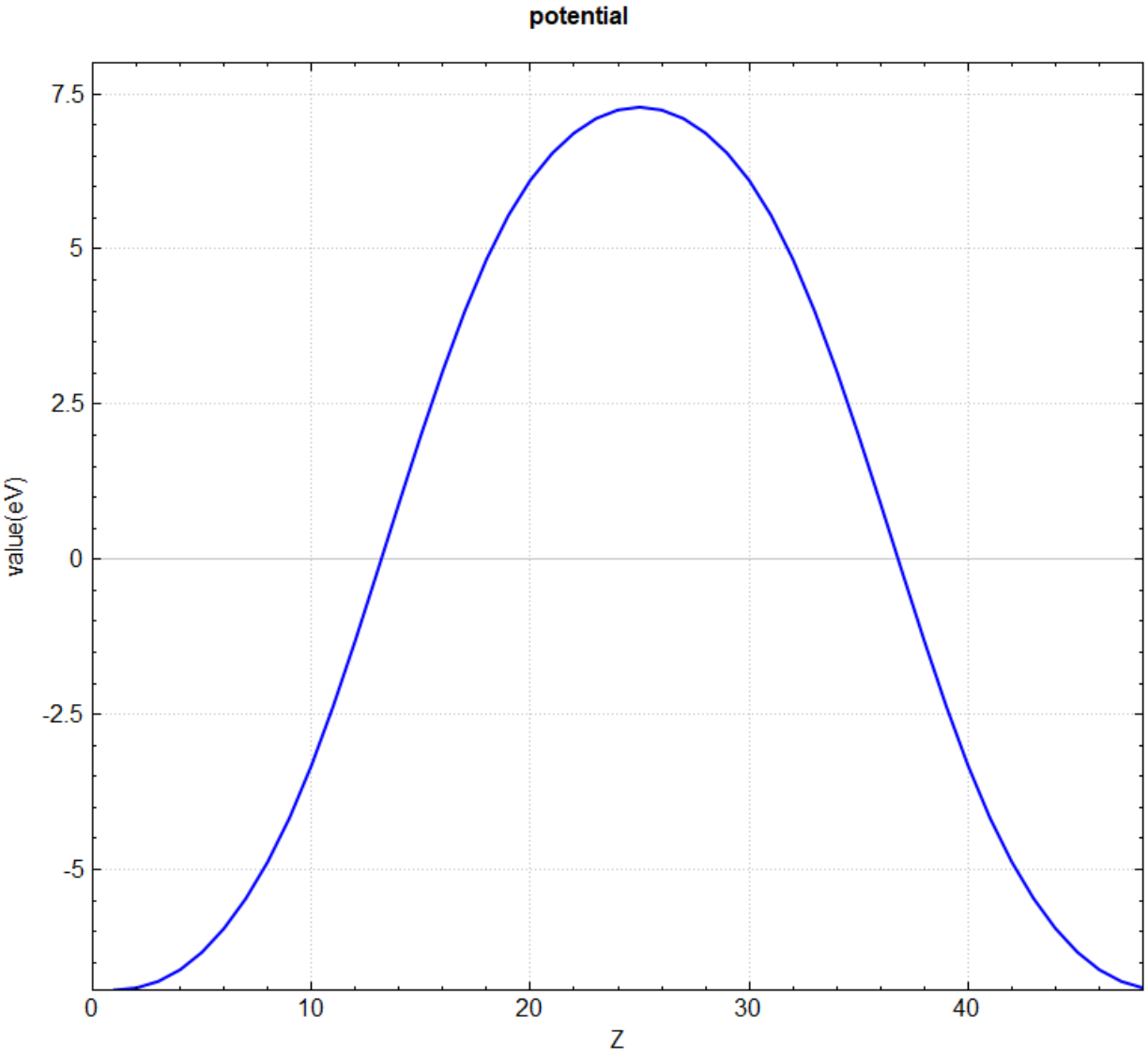
2.8 elf 电子局域密度计算

本节将从自洽出发介绍如何使用 DS-PAW 计算电子局域密度 ELF。以 Si 体系为例进行自洽计算（见 2.2 节），自洽完成之后准备 ELF 计算，并对 ELF 作图进行分析。

2.8.1 Si 电子局域密度计算输入文件

输入文件包含参数文件 ELF.in 结构文件 structure.as 和上次自洽计算得到的二进制电荷密度文件 rho.bin ， ELF.in 如下：

```
# task type
task = elf
#system related
sys.structure = structure.as
sys.symmetry = false
sys.functional = LDA
sys.spin = none
#scf related
cal.iniCharge = ./rho.bin
cal.methods = 2
cal.smearing = 1
cal.ksamping = G
cal.kpoints = [10, 10, 10]
cal.cutoffFactor = 1.5
```


potential
7.5
5
2.5
0
-2.5
-5
value(eV)
0
10
20
30
40
Z

ELF.in 输入参数介绍：在 ELF 计算中可以尽量保留 **sys.** 和 **cal.** 的参数到 `ELF.in` 中：task：本次计算为 ELF 计算，设置 task 为 elf；

- `cal.iniCharge`：表示读取电荷密度二进制文件，支持绝对路径及相对路径，这里./表示当前路径下的 `rho.bin` 文件；

structure.as 文件同（见 2.2 节）自洽计算的。

备注:

1. **需要设置 cal.iniCharge 参数的时候，cal.cutoffFactor 和 cal.cutoff 参数必须和之前自洽计算的结果一致，不然读取 rho.bin 的时候会出现格点数据不匹配的问题；**
2. **本案例是调用自洽的结果进行 ELF 计算，用户也可以在 task=scf 时设置 io.elf=true，此时也能保存 elf.json 文件。**

2.8.2 run 程序运行

准备好输入文件之后，将 `ELF.in` 和 `structure.as` 以及 `rho.bin` 文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 *DS-PAW ELF.in* 。

2.8.3 analysis 计算结果分析

根据上述的输入文件，计算完成之后将会得到 `DS-PAW.log` 、`elf.json` 这 2 个文件。

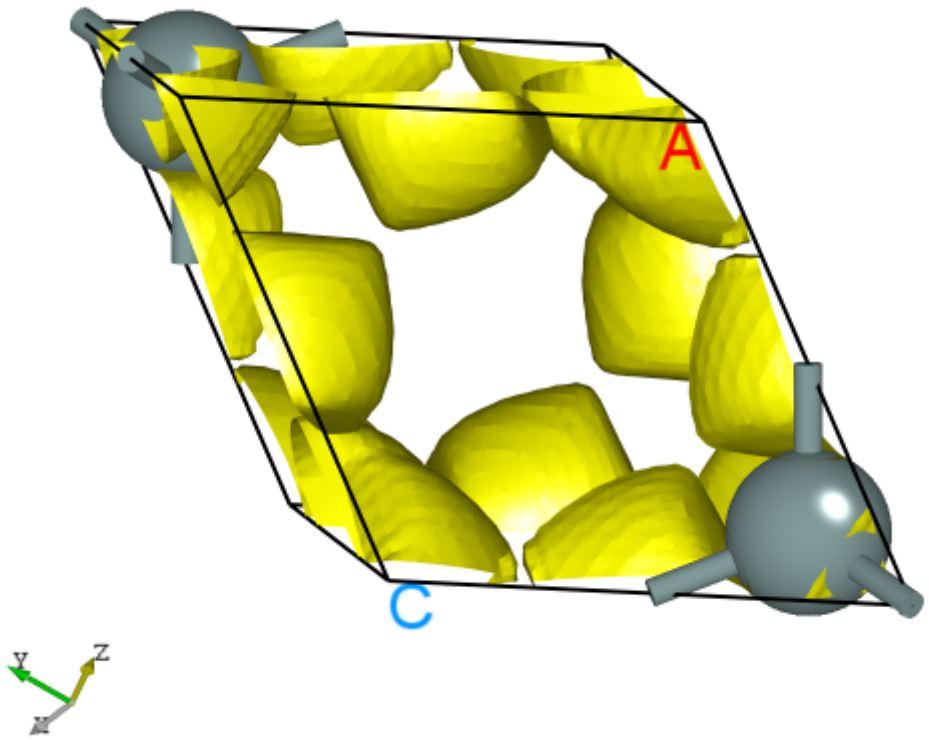
elf.json： ELF 计算完成之后的 **json** 数据文件；

使用 **Device Studio** 可直接对 `elf.json` 文件处理出图，其操作步骤为：Simulator-->DS-PAW-->Analysis Plot，选择 `elf.json` 即可，可根据作图要求自定义设置面板参数，得到一维、二维、三维的电子局域密度图。DS 处理得到的三维电子局域密度图如下所示：

另可使用 **python** 脚本将 `elf.json` 格式的转化成 **VESTA** 软件支持的格式，具体操作见 **辅助工具使用教程** 部分。

2.9 pcharge 部分电荷密度计算

本节将以石墨烯为例分析指定 k 点下特定能带的电荷密度，自洽完成之后准备部分电荷密度的计算，并对部分电荷密度作图进行分析。



2.9.1 graphene 石墨烯部分电荷密度计算输入文件

输入文件包含参数文件 pcharge.in 和结构文件 structure.as ， 上次自洽计算得到的二进制电荷密度文件 rho.bin 和二进制波函数文件 wave.bin , pcharge.in 如下：

```
# task type
task = pcharge
#system related
sys.structure = structure.as
sys.symmetry = false
sys.functional = PBE
sys.spin = collinear
#scf related
cal.methods = 2
cal.smearing = 1
cal.ksamping = G
cal.kpoints = [9, 9, 1]
cal.cutoffFactor = 1.5
#pcharge related
cal.iniCharge = ./rho.bin
cal.iniWave = ./wave.bin
pcharge.bandIndex = [4,5]
pcharge.kpointsIndex = [12]
pcharge.sumK= false
```

pcharge.in 输入参数介绍：

在部分电荷密度计算中可以尽量保留 sys. 和 cal. 的参数到 pcharge.in 中，之后设置部分电荷密度计算特有的参数即可：

- task：本次计算为部分电荷密度计算，设置 task 为 pcharge；
- cal.iniCharge：表示读取电荷密度二进制文件，支持绝对路径及相对路径，这里./表示当前路径下的 rho.bin 文件；
- cal.iniWave：表示读取波函数二进制文件，支持绝对路径及相对路径，这里./表示当前路径下的 wave.bin 文件；
- pcharge.bandIndex：指定需进行电荷密度分析的能带的序号，这里 [4,5] 表示分析能带 4 和能带 5 的电荷密度；
- pcharge.kpointsIndex：指定计算某条能带的电荷密度时所用 K 点数，这里 [12] 表示分析两条能带的电荷密度时 k 点都取 12；
- pcharge.sumK：表示计算部分电荷密度之后保存数据是否将所有 K 点，不同能带的数据相加。这里 false 表示不相加；

structure.as 文件参考如下：

```
Total number of atoms
2
Lattice
2.46120000 0.00000000 0.00000000
-1.23060000 2.13146172 0.00000000
0.00000000 0.00000000 6.70900000
Cartesian
C  0.61530000  0.35524362  3.35450000
C  0.61530000  1.77621810  3.35450000
```

备注：

1. **部分电荷密度分两步完成，第二步必须读取自洽计算的电荷密度文件以及二进制波函数文件 wave.bin**。

2.9.2 run 程序运行

准备好输入文件 pcharge.in 、structure.as 以及自洽计算得到的 rho.bin 、wave.bin 文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 *DS-PAW pcharge.in* 。

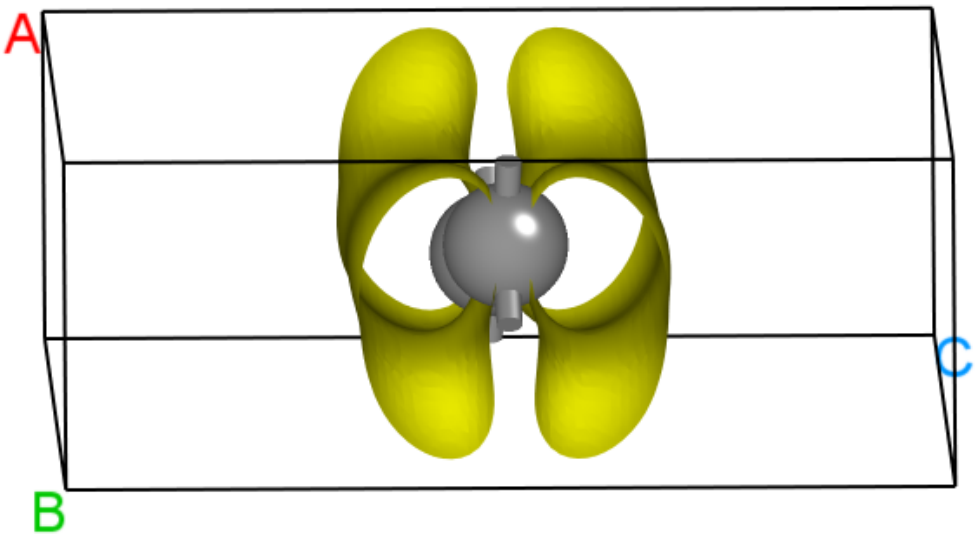
2.9.3 analysis 计算结果分析

根据上述的输入文件，计算完成之后将会得到 DS-PAW.log 、pcharge.json 这 2 个文件。

pcharge.json ： 部分电荷密度计算完成之后的 json 数据文件，此时两条能带的电荷密度数据被保存在 pcharge.json 中，具体的数据结构详见数据结构解析部分；

使用 **Device Studio** 可直接对 pcharge.json 文件处理出图，其操作步骤为：Simulator-->DS-PAW-->Analysis Plot，选择 pcharge.json 即可，可根据作图要求自定义设置面板参数。DS 处理得到的 k 点为 **12** 时能带 **4** 的电荷密度图如下所示：

另可使用 **python** 进行数据处理，具体操作见 **辅助工具使用教程**部分。


A
B
C


X
Y
Z

2.10 hse 杂化泛函计算

本节将以 Si 体系为例，介绍 DS-PAW 程序通过自洽中直接计算能带的方法计算杂化泛函能带，观察使用杂化泛函计算后能带带隙的变化。

2.10.1 Si 杂化泛函计算输入文件

输入文件包含参数文件 `ioband.in` 和结构文件 `structure.as` ，`ioband.in` 如下：

```
# task type
task = scf
#system related
sys.structure = structure.as
sys.symmetry = true
sys.functional = PBE
sys.spin = none
#scf related
cal.methods = 1
cal.smearing = 1
cal.ksamping = G
cal.kpoints = [5, 5, 5]
cal.cutoffFactor = 1.5
#band related
io.band = true
band.kpointsCoord=[0.62500000,0.25000000,0.62500000,0.50000000,0.00000000,0.50000000,
↪0.00000000,0.00000000,0.00000000,0.50000000,0.00000000,0.50000000,0.50000000,0.
↪25000000,0.75000000,0.37500000,0.37500000,0.75000000,0.00000000,0.00000000,0.
↪00000000]
band.kpointsLabel = [U,X,G,X,W,K,G]
band.kpointsNumber = [20,20,20,20,20,20]
band.project = false
#HSE related
sys.hybrid=true
sys.hybridType=HSE06
#outputs
io.charge = false
io.wave = false
```

ioband.in 输入参数介绍：

在杂化泛函计算中可以尽量保留 sys. 和 cal. 的参数到 `ioband.in` 中，之后设置杂化泛函计算特有的参数即可：

- `sys.hybrid`：是否进行杂化泛函计算的开关，true 表示引入杂化泛函计算；
- `sys.hybridType`：指定杂化泛函的类型，此例为 HSE06；

structure.as 文件同自洽计算。（见 2.2 节）

备注：

1. **杂化泛函计算时赝势的类型必须为 PBE**。
2. **若分两步计算能带，能带计算的输入文件中需要设置 cal.iniWave**。
3. **若分两步计算能带，自洽计算和能带计算的输入文件中都需要打开 sys.hybrid=true**。

2.10.2 run 程序运行

准备好输入文件 `ioband.in` 和 `structure.as` 上传到服务器上运行，按照结构弛豫中介绍的方法执行 *DS-PAW ioband.in* 。

2.10.3 analysis 计算结果分析

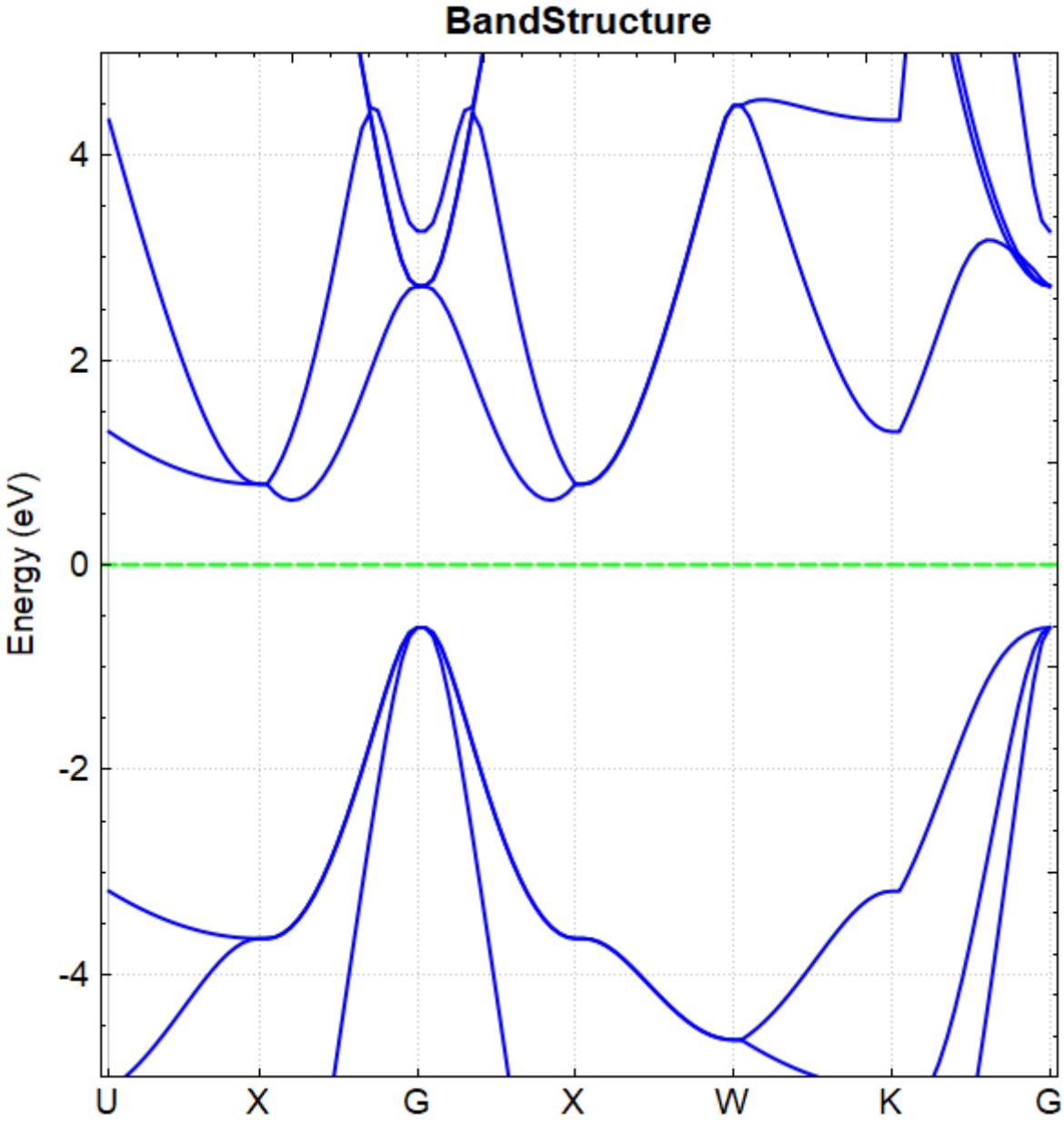
根据上述的输入文件，计算完成之后将会得到 `DS-PAW.log` 、`band.json` 这 2 个文件。处理 `band.json` 的方法同（见 2.3 节）能带计算的方法，DS 处理得到的 `bandplot.png` 文件如下所示，能带图表明打开杂化泛函计算后价带与导带之间的带隙变大，约为 **1.2394** eV，不进行杂化泛函计算得到的能带带隙约为 **0.6433** eV。

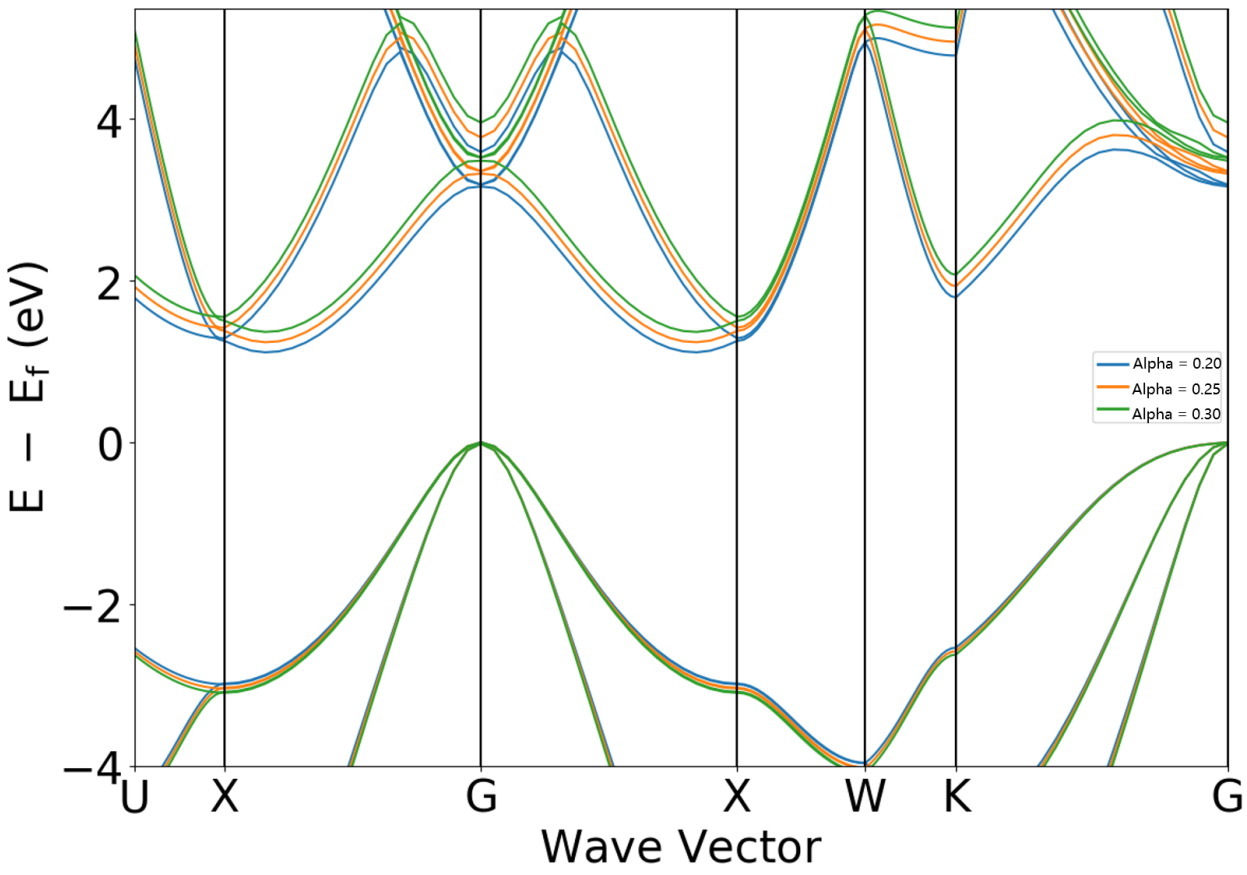
2.10.4 修改杂化泛函 Alpha 系数

2.10.1 章节展示的杂化泛函方法为 HSE06，其对应的杂化泛函系数为默认值 0.25，DS-PAW2022A 版本新增 `sys.hybridAlpha` 和 `sys.hybridOmega` 参数用于控制杂化泛函相关系数，以 `sys.hybridAlpha` 为例，若对 2.10.1 的计算参数做如下修改进行分别完成两次计算：`scf.in` 和 `band.in` 中增加参数 `sys.hybridAlpha = 0.20`，以及 `scf.in` 和 `band.in` 中增加参数 `sys.hybridAlpha = 0.30` ，得到的能带对比图如下所示，分析该图可得通过加大 `sys.hybridAlpha` 系数可以使能带带隙进一步增大。从 `band.json` 文件中可读取当 `sys.hybridAlpha` 分别取值 **0.20** 、**0.25** 、**0.30** 时，对应带隙值分别为 **1.1146** 、**1.2394** 、**1.3665** 。

2.11 vdw 范德瓦尔斯修正计算

本节将以石墨体系的结构弛豫为例，介绍在 DS-PAW 中如何正确的设置范德瓦尔斯修正，并将设置范德瓦尔斯修正与不设置该参数的结果进行对比分析。


BandStructure
Energy (eV)
4
2
0
-2
-4
U
X
G
X
W
K
G


E − E_f (eV)
4
2
0
−2
−4
U
X
G
X
W
K
G
Wave Vector
Alpha = 0.20
Alpha = 0.25
Alpha = 0.30

2.11.1 graphite 石墨结构弛豫输入文件

输入文件包含参数文件 `relax.in` 和结构文件 `structure.as` ， `relax.in` 如下：

```
# task type
task = relax
#system related
sys.structure = structure.as
sys.symmetry = false
sys.functional = PBE
sys.spin = none
#scf related
cal.methods = 1
cal.smearing = 1
cal.ksamping = G
cal.kpoints = [21, 21, 7]
cal.cutoff  = 450
scf.convergence = 1.0e-05
#relax related
relax.max =  60
relax.freedom = all
relax.convergence =  0.01
relax.methods = CG
#vdw related
corr.VDW = true
corr.VDWType = D3G
```

relax.in 输入参数介绍：

在范德瓦尔斯修正计算中可以尽量保留 sys. 和 cal. 的参数到 `relax.in` 中，之后设置范德瓦尔斯修正计算特有的参数即可：

- `corr.VDW` : 表示是否引入范德瓦尔斯修正，这里 true 表示已打开；
- `corr.VDWType` : 表示使用哪种范德瓦尔斯修正，D3G 表示 DFT-D3 of Grimme 方法；

structure.as 文件参考如下：

```
Total number of atoms
4
Lattice
 2.46729136 0.00000000 0.00000000
 -1.23364568 2.13673699 0.00000000
 0.00000000 0.00000000 7.80307245
Cartesian
C 0.00000000 0.00000000 1.95076811
C 0.00000000 0.00000000 5.85230434
C 0.00000000 1.42449201 1.95076811
C 1.23364689 0.71224492 5.85230434
```

备注:

1. **设置范德瓦尔斯修正时赝势的类型必须为 PBE。**

2.11.2 run 程序运行

准备好输入文件之后，将 `relax.in` 和 `structure.as` 文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 *DS-PAW relax.in* 。

2.11.3 analysis 计算结果分析

根据上述的输入文件，计算完成之后将会得到 `DS-PAW.log` 、`relax.json` 和 `system.json` 这 3 个文件。（为作对比另添加一组不考虑范德瓦尔斯的计算）

将 *relax.json* 拖入 Device Studio 查看结构，或直接查看 **paw_tmp** 目录下的 relax.tmp 文件，可得弛豫结束后晶胞常数如下表所示，通过对比可发现添加范德瓦尔斯修正进行结构弛豫所得晶胞向量 **c** 的值与实验报道结果 [1] 更接近。

Procedure	a (Å)	c (Å)
vdw-D3G this work	2.464	7.083
PBE this work	2.464	7.914
Experiment	2.462	6.707

2.12 optical 光学性质计算

本节将以 Si 体系为例，介绍在 DS-PAW 中如何进行光学性质的计算，并对一系列光学性质的物理量进行作图分析。

2.12.1 Si 光学性质计算输入文件

输入文件包含参数文件 `scf.in` 和结构文件 `structure.as` ，`scf.in` 如下：

```
# task type
task = scf
#system related
sys.structure = structure.as
sys.symmetry = true
sys.functional = PBE
sys.spin = none
#scf related
cal.methods = 1
cal.smearing = 1
cal.ksamping = G
cal.kpoints = [4, 4, 4]
```

(续下页)

(接上页)

```
cal.cutoffFactor = 1.5
#optical related
io.optical = true
```

scf.in 输入参数介绍：

在光学性质计算中可以尽量保留 sys. 和 cal. 的参数到 `scf.in` 中，之后设置光学性质计算特有的参数即可：

- `io.optical`：表示是否进行光学性质的计算，当 io.optical=true 时，对体系进行光学性质的计算；

structure.as 文件同自洽计算。（见 2.2 节）

2.12.2 run 程序运行

准备好输入文件之后，将 `scf.in` 和 `structure.as` 文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 *DS-PAW scf.in* 。

2.12.3 analysis 计算结果分析

根据上述的输入文件，计算完成之后将会得到 `DS-PAW.log` 、`optical.json` 这 2 个文件。

optical.json：光学性质计算完成之后的 **json** 数据文件；此时吸光系数，折射率等物理数据被保存在 `optical.json` 中，具体的数据结构详见数据结构解析部分；

使用 **Device Studio** 可直接对 `optical.json` 文件处理出图，其操作步骤为：Simulator-->DS-PAW-->Analysis Plot，选择 `optical.json` 即可，可根据作图要求自定义设置面板参数。DS 处理得到的吸光系数、消光系数、反射率、折射因子随能量的变化曲线分别如下所示：

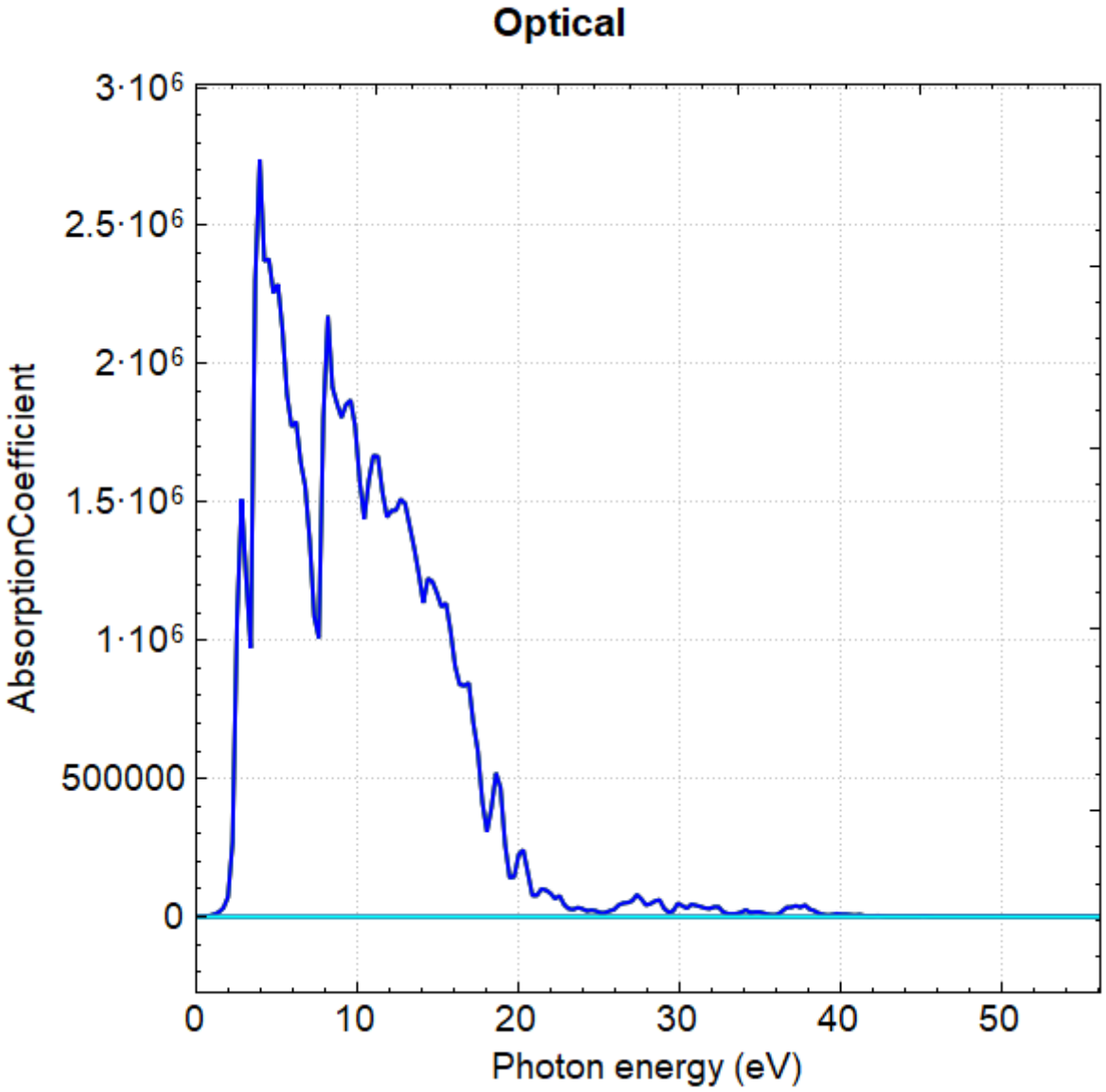
另可使用 **python** 进行数据处理，具体操作见 **辅助工具使用教程**部分。

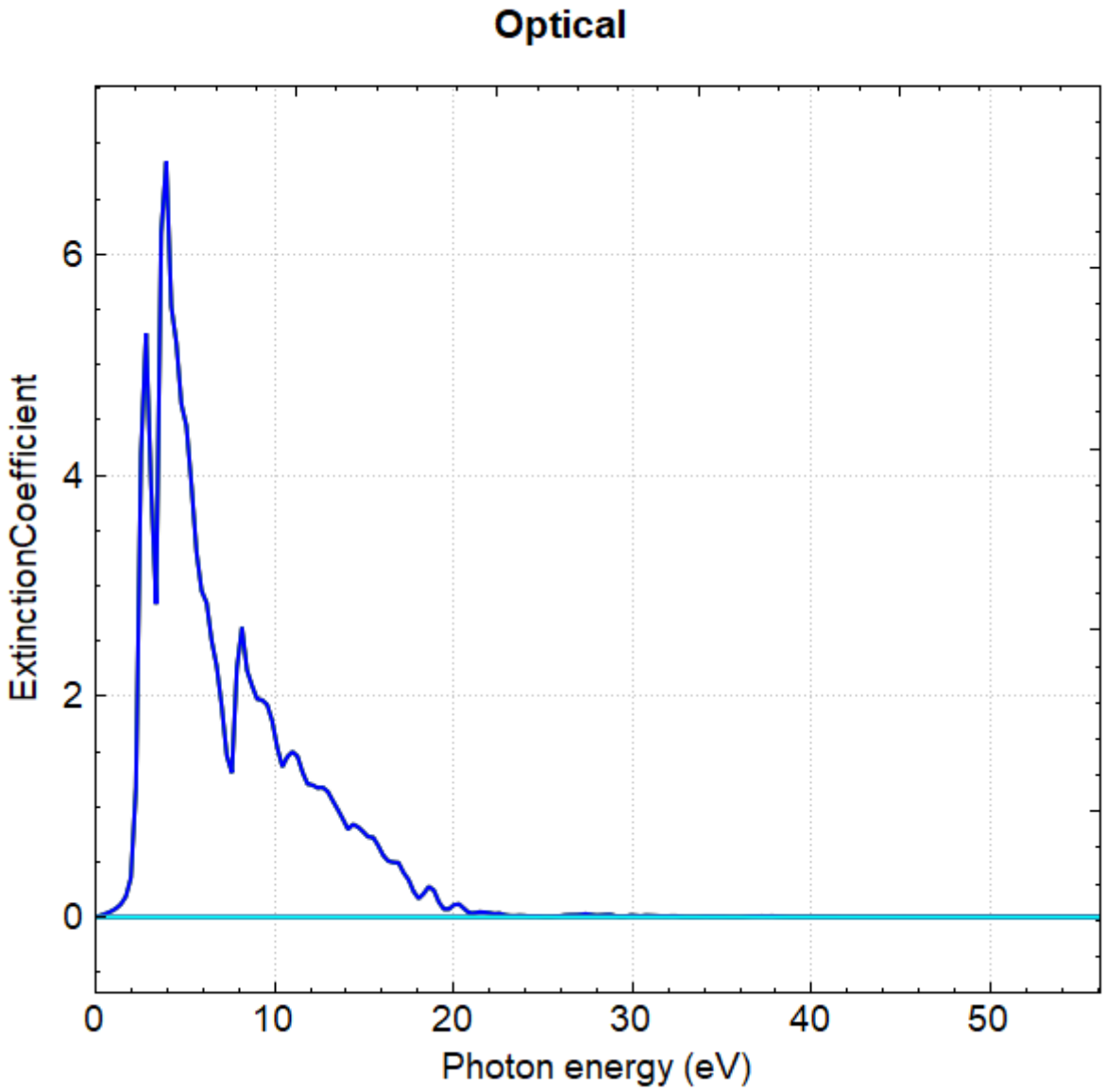
2.13 frequency 频率计算

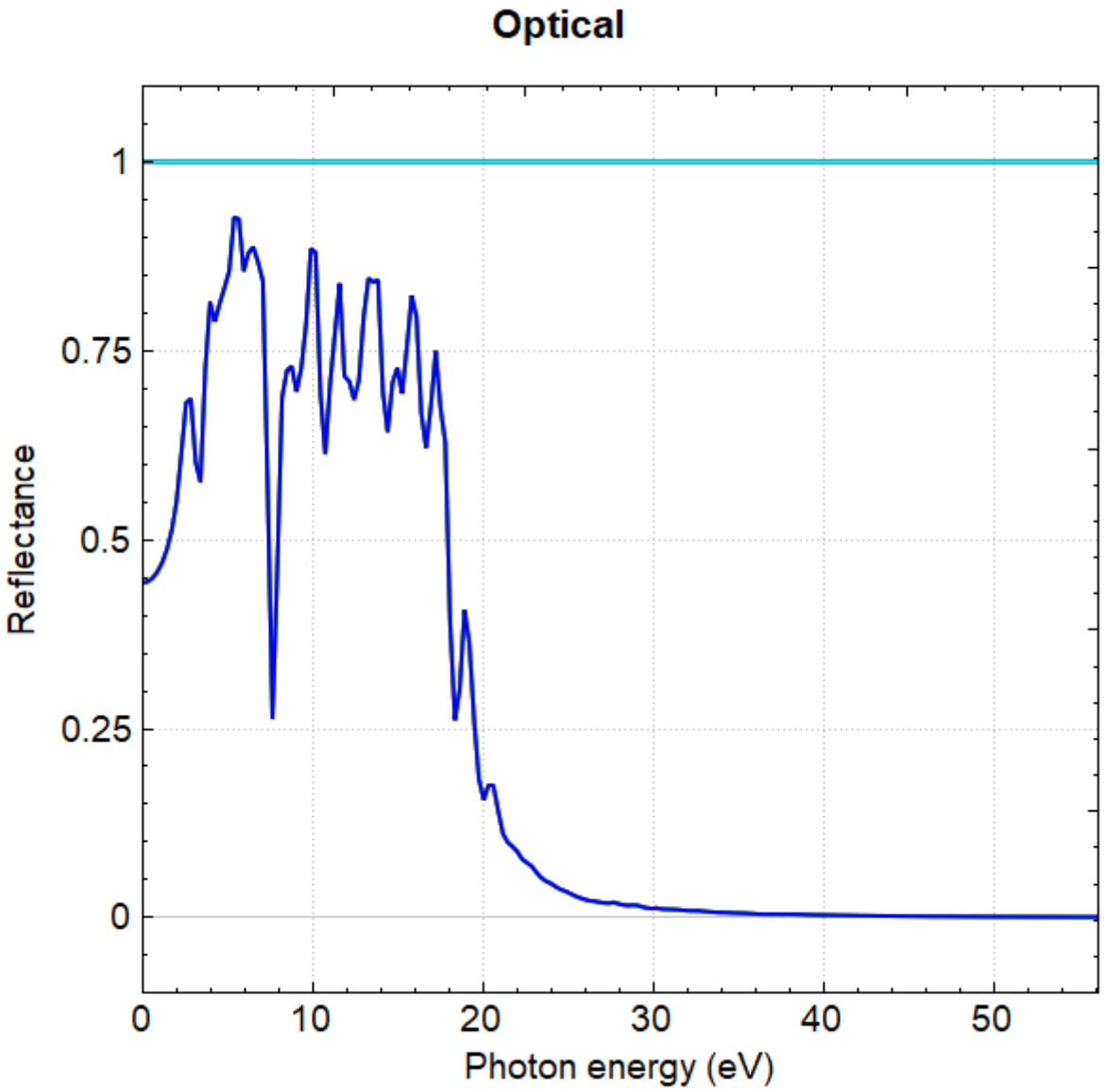
本节将以 CO 分子为例，介绍在 DS-PAW 中如何进行频率计算。

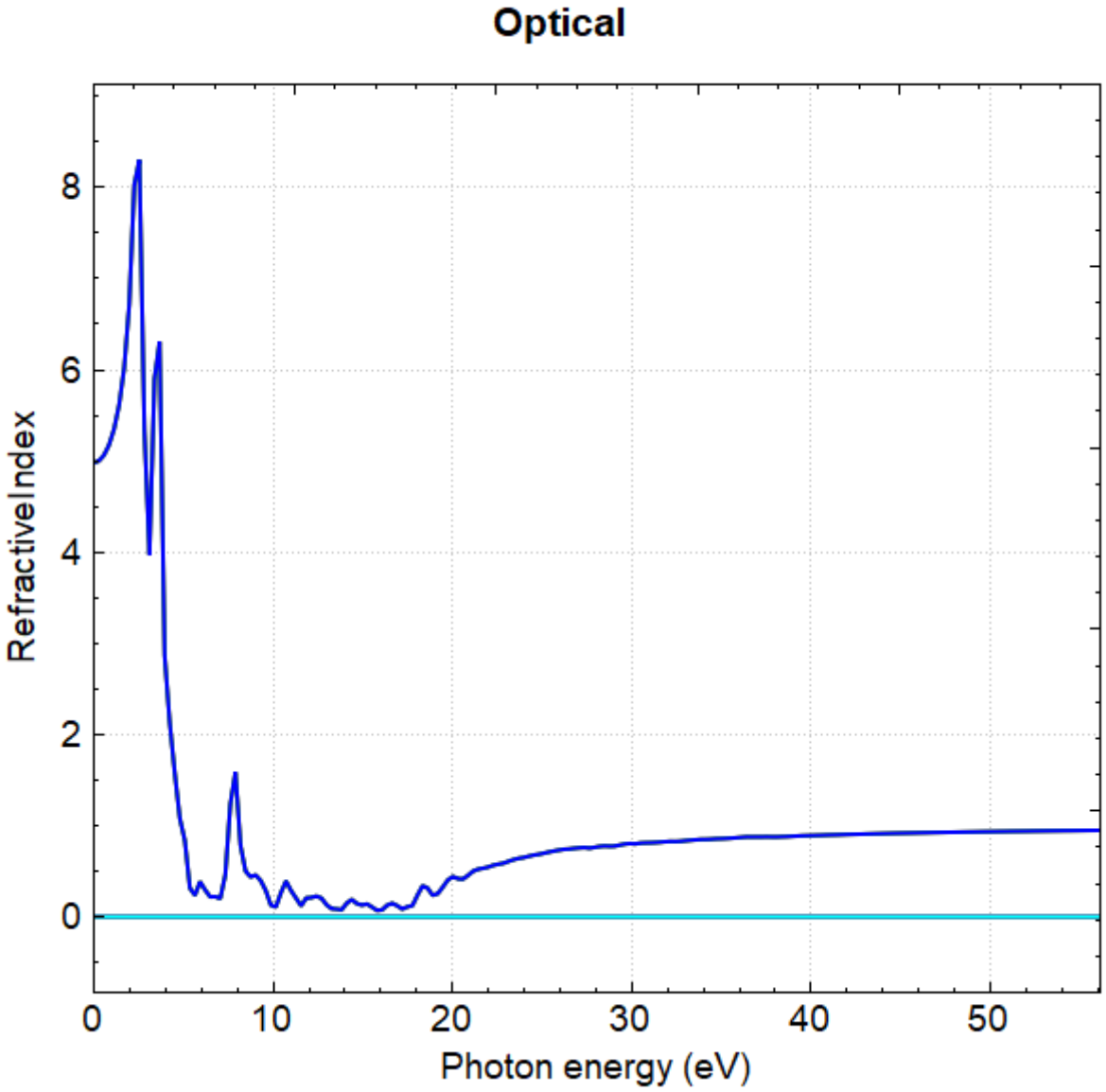
2.13.1 CO 频率计算输入文件

输入文件包含参数文件 `frequency.in` 和结构文件 `structure.as` ， `frequency.in` 如下：


Optical
3·10^6
2.5·10^6
2·10^6
1.5·10^6
1·10^6
500000
0
AbsorptionCoefficient
0
10
20
30
40
50
Photon energy (eV)


Optical
ExtinctionCoefficient
0
2
4
6
0
10
20
30
40
50
Photon energy (eV)


Optical
Reflectance
1
0.75
0.5
0.25
0
0
10
20
30
40
50
Photon energy (eV)


Optical
RefractiveIndex
8
6
4
2
0
0
10
20
30
40
50
Photon energy (eV)

```
# task type
task = frequency
#system related
sys.structure = structure.as
sys.symmetry = false
sys.functional = PBE
sys.spin = none
#scf related
cal.methods = 2
cal.smearing = 1
cal.ksamping = MP
cal.kpoints = [9, 9, 1]
cal.cutoffFactor = 1.5
scf.convergence = 1.0e-6
#frequency related
frequency.dispOrder = 1
frequency.dispRange = 0.02
#outputs
io.charge = false
io.wave = false
```

frequency.in 输入参数介绍：

在频率计算中可以尽量保留 sys. 和 cal. 的参数到 `frequency.in` 中，之后设置频率计算特有的参数即可：

- `task`：本次计算为频率计算，设置 task 为 frequency；
- `frequency.dispOrder`：表示频率计算时原子振动的方式。1 对应中心差分法，即 2 种原子振动方式：每个笛卡尔方向上原子的位移为 `±frequency.dispRange`； 2 对应 4 种原子振动方式：每个笛卡尔方向上原子的位移为 `±frequency.dispRange` 和 `±2*frequency.dispRange`；
- `frequency.dispRange`：表示频率计算时的原子位移大小；

structure.as 文件参考如下：

```
Total number of atoms
2
Lattice
 8.0 0.0 0.0
 0.0 8.0 0.0
 0.0 0.0 8.0
Cartesian  Fix_x Fix_y Fix_z
O 0 0 0       T T F
C 0 0 1.143   T T F
```

备注:

1. **频率计算时应提高自洽计算的收敛精度，建议设置为 1.0e-6 以上。**
2. **CO 只在 z 方向上两个原子可动。**

2.13.2 run 程序运行

准备好输入文件之后，将 `frequency.in` 和 `structure.as` 文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 *DS-PAW frequency.in* 。

2.13.3 analysis 计算结果分析

根据上述的输入文件，计算完成之后将会得到 `DS-PAW.log` 、`frequency.json` 、`frequency.txt` 这 3 个文件。

frequency.json： 频率计算完成之后的 json 数据文件，此时频率数据被保存在该文件中，具体的数据结构详见数据结构解析部分。

frequency.txt : 频率计算完成之后的 **txt** 文本文件，该文件写入频率相关数据，与 `frequency.json` 文件数据一致，便于用户快速获取信息。

从 `frequency.txt` 中可获取以下数据:

Frequency	THz	2PiTHz	cm-1	meV
1 f	63.843104	401.138041	2129.576594	264.033942
2 f/i	0.051123	0.321218	1.705293	0.211429

CO 只在 z 方向上两个原子可动，因此只有两个频率，通过上表可以看到一个振动模式的频率约为 **63.8** THz，还有一个在 **0** 附近的虚频，一般情况虚频小于 2THz 基本可以忽略不计。

2.14 elastic 弹性常数计算

本节将以 Si 体系为例，介绍在 DS-PAW 中如何进行弹性计算。

2.14.1 Si 弹性常数计算输入文件

输入文件包含参数文件 `elastic.in` 和结构文件 `structure.as` ， `elastic.in` 如下:

```
# task type
task = elastic
#system related
sys.structure = structure.as
sys.symmetry = true
sys.functional = PBE
sys.spin = none
#scf related
```

(续下页)

(接上页)

```
cal.methods = 1
cal.smearing = 1
cal.ksamping = G
cal.kpoints = [5, 5, 5]
cal.cutoffFactor = 1.5
scf.convergence = 1.0e-6
#frequency related
elastic.dispOrder = 1
elastic.dispRange = 0.01
#outputs
io.charge = false
io.wave = false
```

elastic.in 输入参数介绍：

在弹性计算中可以尽量保留 sys. 和 cal. 的参数到 `elastic.in` 中，之后设置弹性计算特有的参数即可：

- `task`：本次计算为弹性计算，设置 task 为 elastic；
- `elastic.dispOrder`：表示弹性计算时原子振动的方式, 1 对应中心差分法；
- `elastic.dispRange`：表示弹性计算时的原子位移大小；

structure.as 文件参考如下：

```
Total number of atoms
8
Lattice
 5.43070000 0.00000000 0.00000000
 0.00000000 5.43070000 0.00000000
 0.00000000 0.00000000 5.43070000
Cartesian
Si 0.67883750 0.67883750 0.67883750
Si 3.39418750 3.39418750 0.67883750
Si 3.39418750 0.67883750 3.39418750
Si 0.67883750 3.39418750 3.39418750
Si 2.03651250 2.03651250 2.03651250
Si 4.75186250 4.75186250 2.03651250
Si 4.75186250 2.03651250 4.75186250
Si 2.03651250 4.75186250 4.75186250
```

备注:

1. **弹性计算时应提高自洽计算的收敛精度，建议设置在 1.0e-6 以上。**
2. **弹性计算不支持固定原子。**

2.14.2 run 程序运行

准备好输入文件之后，将 `elastic.in` 和 `structure.as` 文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 *DS-PAW elastic.in* 。

2.14.3 analysis 计算结果分析

根据上述的输入文件，计算完成之后将会得到 `DS-PAW.log` 、`elastic.json` 、`elastic.txt` 这 3 个文件。

elastic.json : 弹性计算完成之后的 **json** 数据文件，此时弹性模量被保存在 `elastic.json` 中，具体的数据结构详见数据结构解析部分；

elastic.txt : 弹性计算完成之后的 **txt** 文本文件，该文件写入弹性相关数据，与 `elastic.json` 文件数据一致，便于用户快速获取信息。

从 `elastic.txt` 文件可得如下弹性常数矩阵：

158.7644	62.9858	62.9858	0.0000	-0.0000	0.0000
62.9858	158.7644	62.9858	0.0000	0.0000	0.0000
62.9858	62.9858	158.7644	-0.0000	0.0000	0.0000
0.0000	0.0000	-0.0000	75.8807	-0.0000	0.0000
-0.0000	0.0000	0.0000	-0.0000	75.8807	-0.0000
0.0000	0.0000	0.0000	0.0000	-0.0000	75.8807

刚性弹性矩阵

0.0081	-0.0023	-0.0023	-0.0000	0.0000	-0.0000
-0.0023	0.0081	-0.0023	-0.0000	-0.0000	0.0000
-0.0023	-0.0023	0.0081	0.0000	-0.0000	0.0000
-0.0000	-0.0000	0.0000	0.0132	0.0000	-0.0000
0.0000	-0.0000	-0.0000	0.0000	0.0132	0.0000
-0.0000	0.0000	0.0000	-0.0000	0.0000	0.0132

柔性弹性矩阵

Properties	Vogit	Reuss	Hill
BulkModulus(GPa)	94.9120	94.9120	94.9120
ShearModulus(GPa)	64.6841	61.5016	63.0929
YoungModulus(GPa)	158.1297	151.7315	154.9452
PoissonRatio	0.2223	0.2336	0.2279

体积模量、剪切模量、杨氏模量和泊松比

Si 体系为 Cubic 晶系，该晶系的独立矩阵元有三个：**C11** ，**C12** ，**C44** ，分别对应表中的 158.7644、62.9858、75.8807。

2.15 neb 过渡态计算

本节将以 Pt 原子体系为例，介绍在 DS-PAW 中如何进行过渡态计算（Cl-NEB），并对结果进行作图分析。

2.15.1 Pt 过渡态计算输入文件

输入文件包含参数文件 `neb.in` 和多个结构文件 `structure##.as` ，`neb.in` 文件如下：

```
# task type
task = neb
#system related
sys.structure = structure.as
sys.symmetry = false
sys.functional = PBE
sys.spin = none
#scf related
cal.methods = 1
cal.smearing = 1
cal.ksamping = G
cal.kpoints = [3, 3, 1]
cal.cutoffFactor = 1.5
#neb related
neb.max = 60
neb.iniFin=true
neb.springK = 5
neb.images = 5
neb.method = LBFGS
neb.convergence = 0.1
#outputs
io.charge = false
io.wave = false
```

neb.in 输入参数介绍：

在过渡态计算中可以尽量保留 sys. 和 cal. 的参数到 `neb.in` 中，之后设置过渡态计算特有的参数即可：

- `task`： 本次计算为过渡态计算，设置 task 为 neb；
- `neb.stepRange`：表示过渡态计算中结构弛豫的步长；
- `neb.max`：表示过渡态计算中结构弛豫的最大步数；
- `neb.iniFin`：表示过渡态计算中初态结构和末态结构是否进行自洽计算，true 表示进行自洽计算；
- `neb.springK`：表示过渡态计算中弹簧系数 K；
- `neb.images`：表示过渡态计算中的中间结构的数目；
- `neb.method`：表示过渡态计算中使用的算法；
- `neb.convergence`：表示过渡态计算中受力的收敛标准；

structure.as 需提供多个，初态结构 `structure00.as` 参考如下

```
Total number of atoms
13
Lattice
```

(续下页)

(接上页)

```
5.6058       0       0
0  5.6058       0
0       0 16.8174
Cartesian Fix_x Fix_y Fix_z
Pt  1.40145 1.40145 1.98193  T T T
Pt  4.20435 1.40145 1.98193  T T T
Pt  1.40145 4.20435 1.98193  T T T
Pt  4.20435 4.20435 1.98193  T T T
Pt        0       0 3.89338  F F F
Pt        0  2.8029 3.89869  F F F
Pt   2.8029       0 3.96355  F F F
Pt   2.8029  2.8029 3.89338  F F F
Pt  1.43659 1.36631 5.86749  F F F
Pt  4.16921 1.36631 5.86749  F F F
Pt  1.43659 4.23949 5.86749  F F F
Pt  4.16921 4.23949 5.86749  F F F
Pt        0  2.8029 7.51195  F F F
```

末态结构 `structure06.as` 参考如下

```
Total number of atoms
13
Lattice
 5.6058       0       0
      0  5.6058       0
      0       0 16.8174
Cartesian Fix_x Fix_y Fix_z
Pt  1.40145 1.40145 1.98193  T T T
Pt  4.20435 1.40145 1.98193  T T T
Pt  1.40145 4.20435 1.98193  T T T
Pt  4.20435 4.20435 1.98193  T T T
Pt        0       0 3.89338  F F F
Pt        0  2.8029 3.89869  F F F
Pt   2.8029       0 3.96355  F F F
Pt   2.8029  2.8029 3.89338  F F F
Pt   2.8029       0 7.51195  F F F
Pt  4.23949 1.43659 5.86749  F F F
Pt  1.36631 4.16921 5.86749  F F F
Pt  4.23949 4.16921 5.86749  F F F
Pt  1.36631 1.43659 5.86749  F F F
```

备注:

1. **过渡态计算时需准备多个结构文件，命名规则为 structure##.as，## 为结构序号，从 00 开始编号，00 对应初态，依次递增。**
2. **中间结构的生成可参考" 辅助工具使用教程-过渡态部分 "。**
3. **过渡态计算时的结构文件 structure##.as 需存放在命名为 ## 的文件夹中，文件夹序号与结构文件的序号需一致。文件夹外放置一个 neb.in 文件即可，在 neb.in 所在目录执行程序。**
4. **过渡态计算执行程序时调用的核数设置为 images 的整数倍。**
5. **neb 计算时初态末态需先进行结构弛豫。**

2.15.2 run 程序运行

准备好输入文件之后，将 neb.in 文件和包含 structure##.as 文件的多个文件夹文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 *DS-PAW neb.in* 。

2.15.3 analysis 计算结果分析

根据上述的输入文件，计算完成之后初态和末态结构所在文件夹会生成自洽计算所得的 DS-PAW.log 、system.json 这 2 个文件，中间结构 structure##.as 所在文件夹 ## （参与过渡态计算的结构，由 neb.images 参数决定）会生成结构优化所得的 relax.json 、system.json 、neb##.json 、paw_tmp/neb.tmp 这 4 个文件。最外层目录将会生成 DS-PAW.log 、neb.json 这 2 个文件，其中 neb.json 为各 neb##.json 文件的信息汇总。

neb.json： 过渡态计算完成之后的 **json** 数据文件；此时反应坐标及能量变化等数据被保存在 neb.json 中，具体的数据结构详见数据结构解析部分；

paw_tmp/neb.tmp： 中间构型在优化过程中的轨迹文件，默认 neb 计算过程中每个离子步记录一次结构信息；

使用 **Device Studio** 可直接对 neb.json 文件处理出图，其操作步骤为：Simulator-->DS-PAW-->Analysis Plot，选择 neb.json 即可，可根据作图要求自定义设置面板参数。DS 处理得到的势垒曲线及 image3 在弛豫过程中的能量与受力曲线如下所示：

除此以外，DS 可以以 gif 动画的形式展示过渡态搜寻中的轨迹变化，在面板依次选择 **Simulator-->DS-PAW-->Analysis Plot-->neb.json-->Trajectory** 即可，以下为截取其中一帧的图像：

另可使用 **python** 进行数据处理，具体操作见 **辅助工具使用教程**部分。

2.16 phonon 声子谱计算

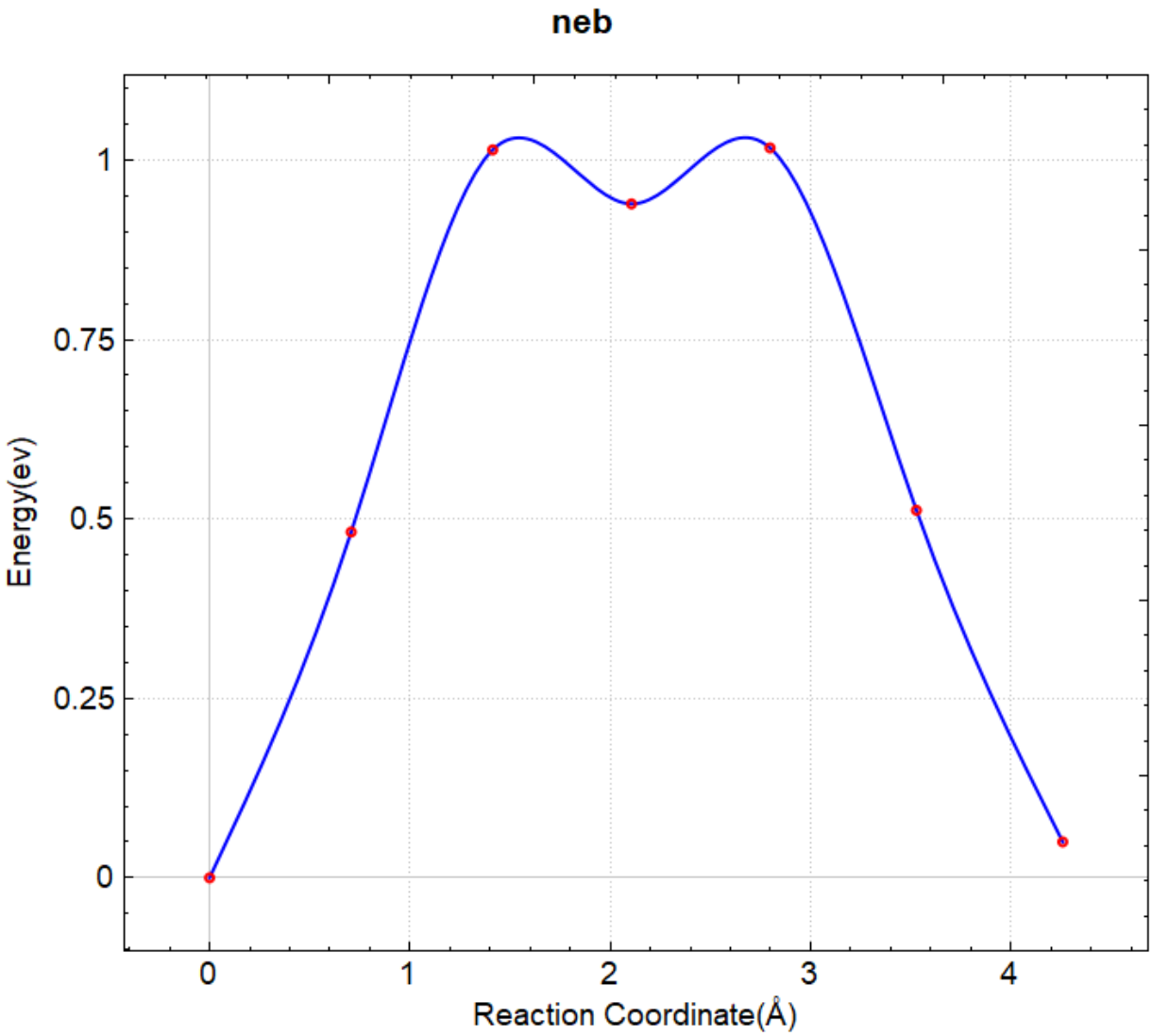
本节介绍 DS-PAW 程序如何进行声子计算及声子能带和声子态密度计算。DS-PAW 支持两种声子谱计算的方法：fd 有限位移法和 dfpt 密度泛函微扰理论方法。本节以单个 MgO 体系为例，介绍如何用 dfpt 方法计算声子能带和态密度，并对声子能带和态密度作图进行分析。

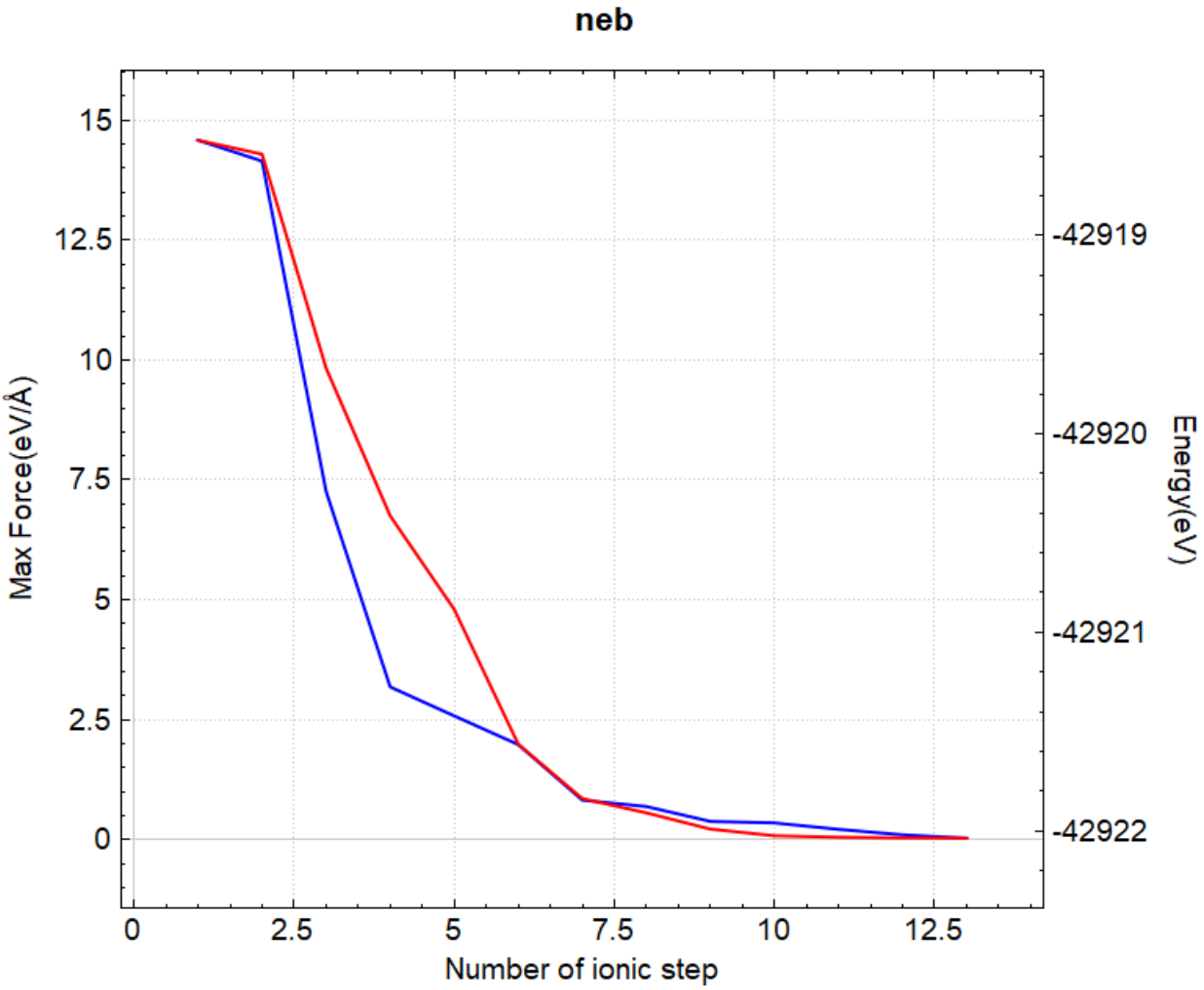
2.16.1 MgO 声子谱能带计算输入文件

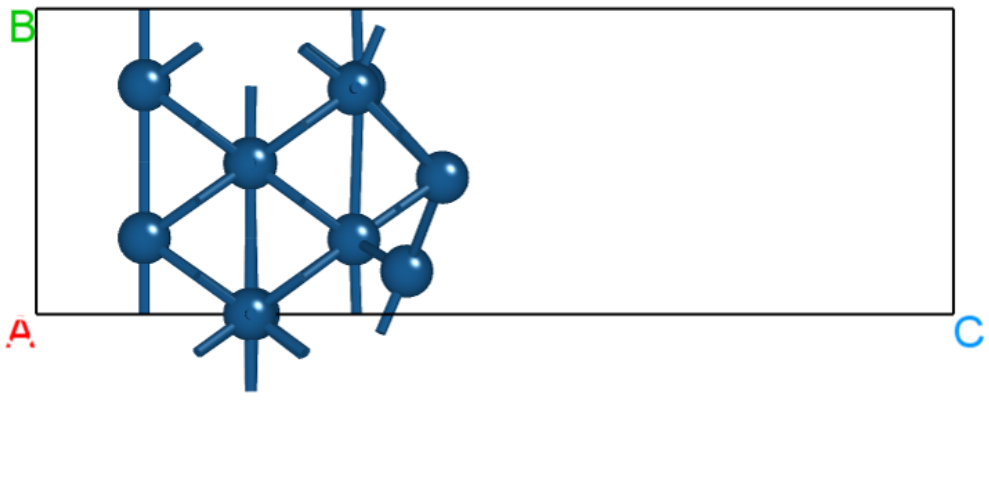
输入文件包含参数文件 phonon.in 和结构文件 structure.as ，phonon.in 如下：

```
task = phonon

sys.structure = structure.as
sys.functional = PBE
sys.spin = none
```

(续下页)


neb
Energy(ev)
1
0.75
0.5
0.25
0
0
1
2
3
4
Reaction Coordinate(Å)


neb
15
12.5
10
7.5
5
2.5
0
Max Force(eV/Å)
-42919
-42920
-42921
-42922
Energy(eV)
0
2.5
5
7.5
10
12.5
Number of ionic step


B
A
C


Y
X
Z

(接上页)

```
cal.methods = 1
cal.smearing = 1
sys.symmetry = true
scf.convergence = 1.0e-07
cal.ksamping = G
cal.kpoints = [3,3,3]
cal.sigma = 0.25

phonon.type = bandDos
phonon.structureSize = [2,2,2]
phonon.primitiveUVW = [0.0, 0.5, 0.5, 0.5, 0.0, 0.5, 0.5, 0.5, 0.0]
phonon.method = dfpt
phonon.qpoints = [41,41,41]
phonon.dosRange = [0,20]
phonon.qpointsLabel = [G,X,W,G,M]
phonon.qpointsCoord = [0.0, 0.0, 0.0, 0.5, 0.0, 0.0, 0.5, 0.5, 0.0, 0.0, 0.0, 0.0, 0.
↪5, 0.5, 0.5]
phonon.qpointsNumber = 51

io.charge = false
io.wave = false
```

phonon.in 输入参数介绍：

在声子计算中可以尽量保留 sys. 和 cal. 的参数到 `phonon.in` 中，之后设置声子计算特有的参数即可：

- `task`：本次计算为声子计算，设置 task 为 phonon；
- `phonon.type`：表示声子计算哪些性质，bandDos 对应计算声子能带和态密度；
- `phonon.structureSize`：声子计算时超胞的大小；
- `phonon.primitiveUVW`：表示声子能带计算时原胞 UVW 的系数；
- `phonon.method`：表示声子计算的方式，dfpt 为密度泛函微扰理论方法；
- `phonon.qpoints`：表示声子计算 q 空间网格取样为 41*41*41；
- `phonon.dosRange`：表示声子态密度计算的能量区间为 [0,20]；
- `phonon.qpointsLabel`：表示声子能带计算时高对称点标签；
- `phonon.qpointsCoord`：表示声子能带计算时高对称点坐标；
- `phonon.qpointsNumber`：表示声子能带相邻两个高对称点的间隔；

structure.as 文件参考如下：

```
Total number of atoms
8
Lattice
     4.2555564654942897    0.0000000000000000    0.0000000000000000
     0.0000000000000000    4.2555564654942888    0.0000000000000000
     0.0000000000000000    0.0000000000000000    4.2555564654942897
Direct
Mg  0.0000000000000000  0.0000000000000000  0.0000000000000000
Mg  0.0000000000000000  0.5000000000000000  0.5000000000000000
Mg  0.5000000000000000  0.0000000000000000  0.5000000000000000
Mg  0.5000000000000000  0.5000000000000000  0.0000000000000000
O   0.5000000000000000  0.5000000000000000  0.5000000000000000
```

(续下页)

(接上页)

```
O   0.5000000000000000  0.0000000000000000  0.0000000000000000
O   0.0000000000000000  0.5000000000000000  0.0000000000000000
O   0.0000000000000000  0.0000000000000000  0.5000000000000000
```

备注:

1. **声子计算时应提高自洽计算的收敛精度，建议设置在 1.0e-7 以上。**
2. **计算声子之前建议先作结构弛豫计算（relax.freedom = all），便于得到准确的声子谱图。**
3. **声子计算时若打开对称性，建议适当提高对称性判断的精度，参数 sys.symmetryAccuracy 可设置为 1.0e-6 或更小，助于得到准确的计算结果。**
4. **phonon.iniPhonon 可指定路径读取声子计算（phonon.type = phonon）得到的 phonon.json 文件，从而直接进行能带与态密度的计算。**
5. **phonon.type 控制计算声子的类型，phonon 对应计算声子，band 对应计算声子能带，dos 对应计算声子态密度，bandDos 对应同时计算声子能带和态密度。当 phonon.type = band/dos/bandDos 且 phonon.iniPhonon 未指定文件路径时，程序先自动完成 phonon.type = phonon 的声子计算，然后根据任务计算能带或态密度。**

2.16.2 run 程序运行

准备好输入文件之后，将 `phonon.in` 和 `structure.as` 文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 *DS-PAW phonon.in* 。

2.16.3 analysis 计算结果分析

根据上述的输入文件，计算完成之后将会得到 `DS-PAW.log` 、`phonon.json` 、`dfpt.json` 和 `dfpt.as` 这 4 个文件。

dfpt.as： 声子计算时超胞的结构文件，计算声子时读取该文件信息。

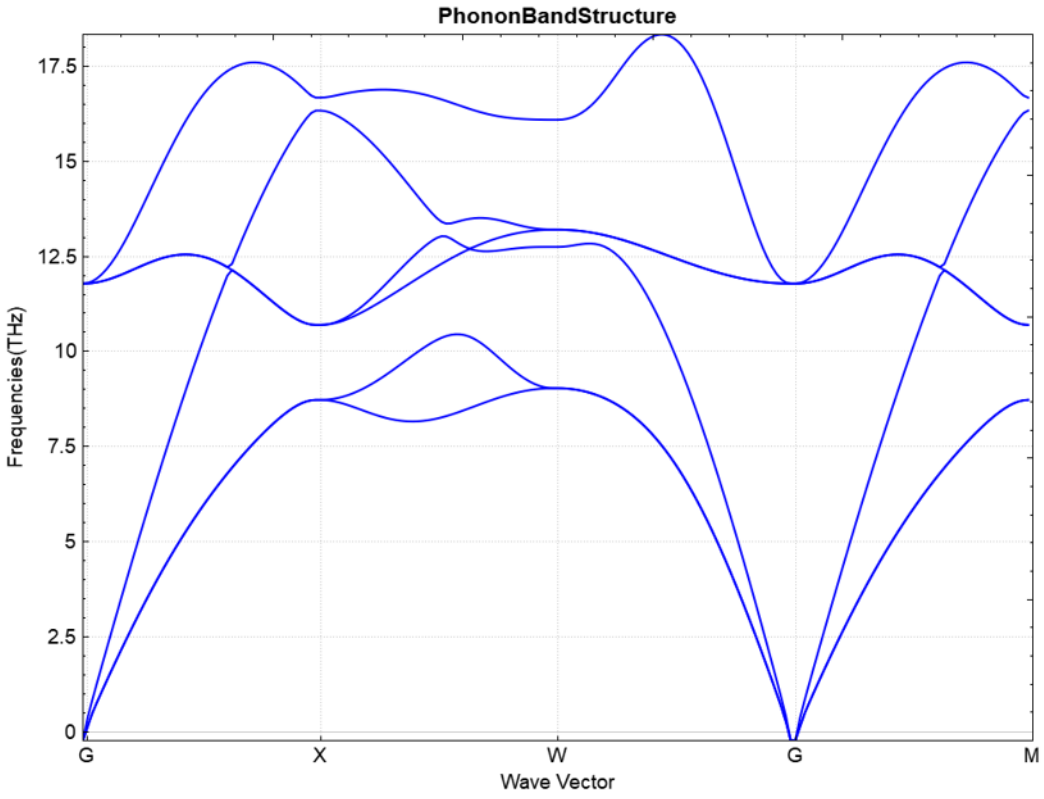
dfpt.json： 声子计算时的参数文件，该文件与 `phonon.in` 文件信息一致，计算声子时读取该文件信息。

phonon.json： 声子计算完成之后的 **json** 数据文件；此时声子能带数据被保存在 `phonon.json` 中，具体的数据结构详见数据结构解析部分；

使用 **Device Studio** 可直接对 `phonon.json` 文件处理出图，其操作步骤为：Simulator-->DS-PAW-->Analysis Plot，选择 `phonon.json` 即可，可根据作图要求自定义设置面板参数，处理得到的声子能带和态密度图如下（a）、（b）所示：

(a)

(b)



2.16.4 nac 计算结果分析

上节展示的为不考虑长程相互作用的声子能带计算，若打开 **non-analytical term correction (nac)** 进行声子计算，在上节所示的 `phonon.in` 文件中添加以下两个参数即可：

```
phonon.dfptEpsilon=true
phonon.nac = true
```

得到的声子能带图如下（c）所示：

(c)

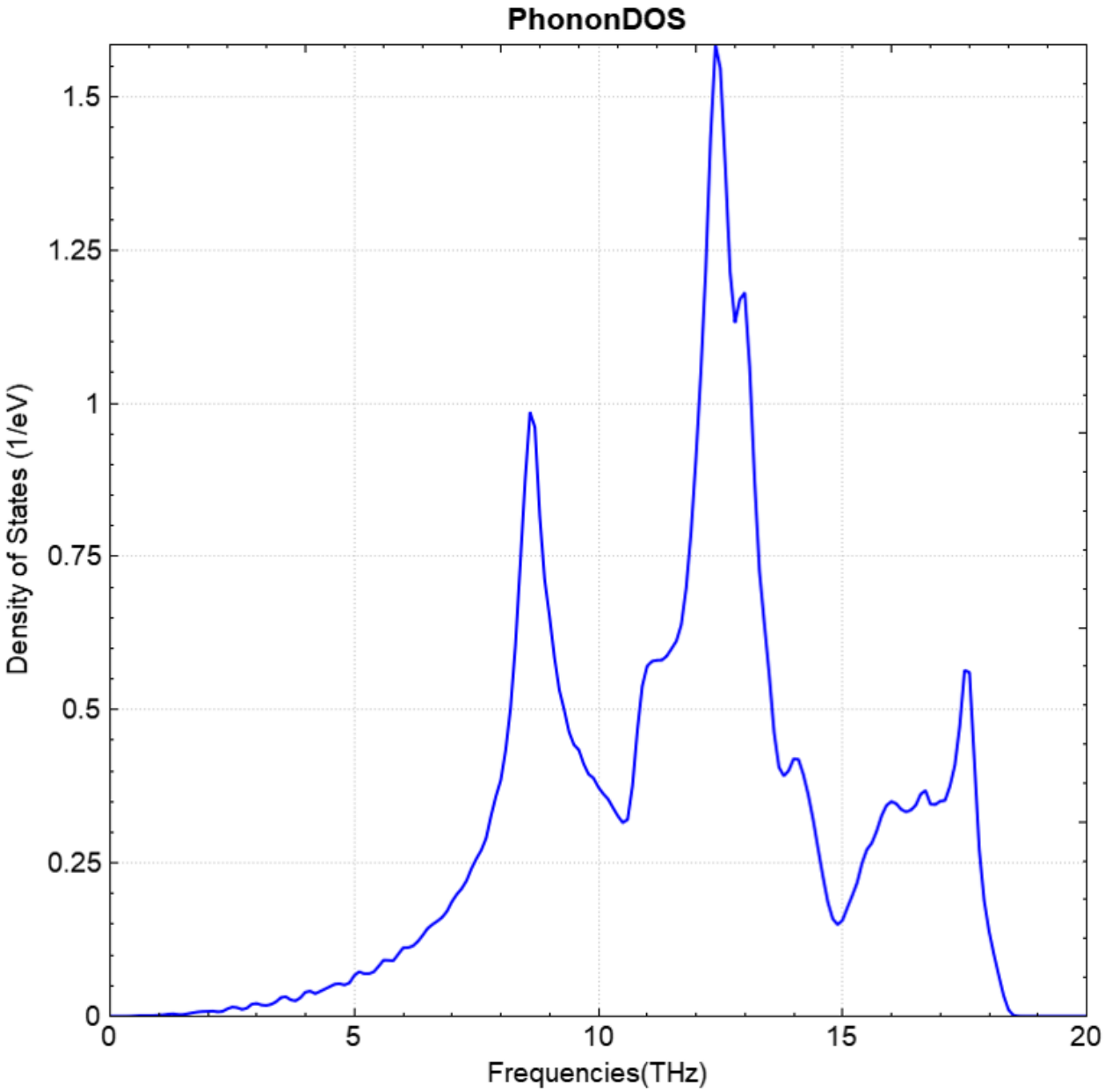
另可使用 **python** 进行数据处理，具体操作见 **辅助工具使用教程**部分。

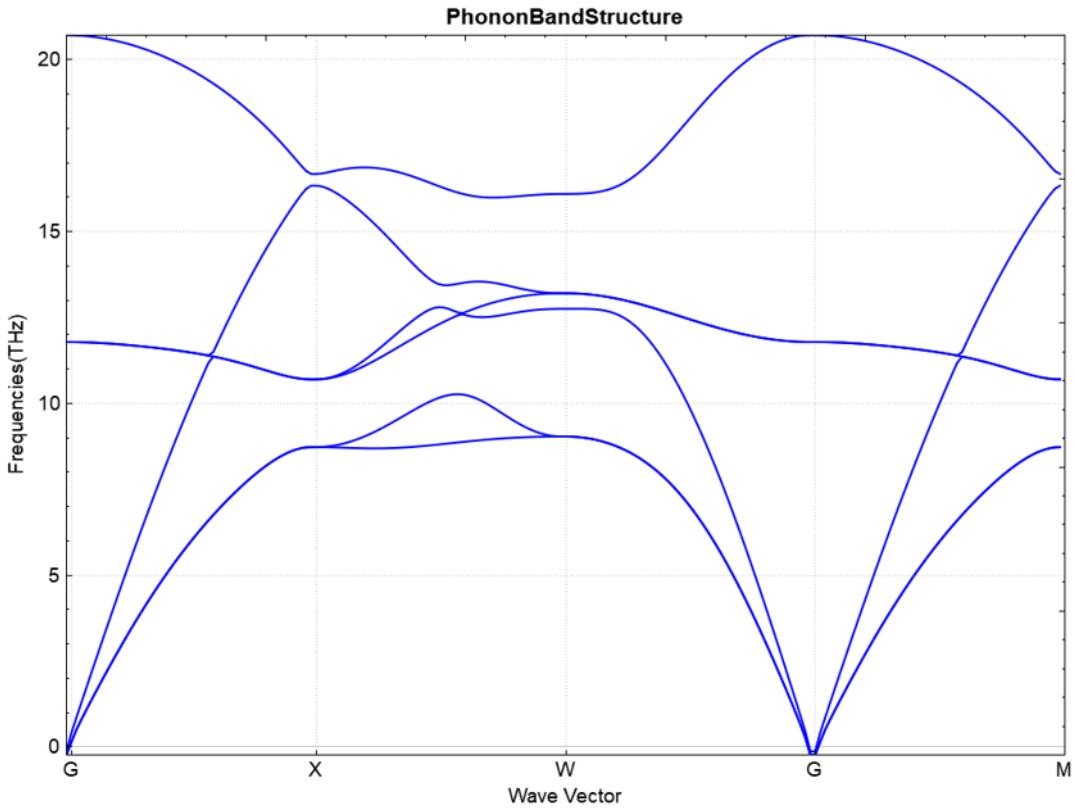
2.16.5 fdphonon 有限位移法计算声子

有限位移法（fd）计算的输入文件如下所示，将参数 `phonon.method = dfpt` 修改为 `phonon.method = fd` 即可，需要注意的是，fd 法计算得到输出文件与 dfpt 法不同。

```
task = phonon

sys.structure = structure.as
sys.functional = PBE
sys.spin = none
```

(续下页)


PhononDOS
Density of States (1/eV)
1.5
1.25
1
0.75
0.5
0.25
0
0
5
10
15
20
Frequencies(THz)



(接上页)

```
cal.methods = 1
cal.smearing = 1
sys.symmetry = true
scf.convergence = 1.0e-07
cal.ksamping = G
cal.kpoints = [3,3,3]
cal.sigma = 0.25

phonon.type = bandDos
phonon.structureSize = [2,2,2]
phonon.primitiveUVW = [0.0, 0.5, 0.5, 0.5, 0.0, 0.5, 0.5, 0.5, 0.0]
phonon.method = fd
phonon.qpointsLabel = [G,X,W,G,M]
phonon.qpointsCoord = [0.0, 0.0, 0.0, 0.5, 0.0, 0.0, 0.5, 0.5, 0.0, 0.0, 0.0, 0.0, 0.
↪5, 0.5, 0.5]
phonon.qpointsNumber = 51

io.charge = false
io.wave = false
```

以 MgO 体系为例，`phonon.structureSize` 设置为 `[2,2,2]`，fd 法计算完成之后将会得到 `DS-PAW.log`、`phonon.json` 两个文件和 `001`、`002` 文件夹。`001` 文件夹下存在 `input.json` 和 `disp-001.as` 文件，`002` 文件夹下存在 `input.json` 和 `disp-002.as` 文件，子文件夹中的两个文件等同于写入输入参数的 **in** 文件和结构参数的 **as** 文件，生成文件夹（001 002...）的个数取决于体系的对称性。

用 **Device Studio** 处理 fd 法计算得到的 *phonon.json* 文件，得到能带图及态密度图同 dfpt 方法计算得到的图（a）与（b）一致。

备注:

1. **介电常数的计算只在 phonon.method = dfpt 时才能完成**
2. **phonon.nac 的开关只在 phonon.method = dfpt 且 phonon.dfptEpsilon=true 时生效**

2.17 soc 自旋轨道耦合计算

本节介绍 DS-PAW 如何进行自旋轨道耦合计算。以 Bi_2Se_3 体系为例，使用两步法进行能带计算并对能带进行作图分析。

2.17.1 Bi_2Se_3 自旋轨道耦合计算输入文件

首先进行自洽计算：输入文件包含参数文件 `soi.in` 和结构文件 `structure.as` ， `soi.in` 如下：

```
# task type
task = scf
#system related
sys.structure = structure.as
sys.symmetry = true
sys.functional = LDA
#scf related
cal.methods = 2
cal.smearing = 1
cal.ksamping = G
cal.kpoints = [7, 7, 7]
cal.cutoffFactor = 1.5
#soi related
sys.spin= non-collinear
sys.soi = true
#outputs
io.charge = true
io.wave = false
```

soi.in 输入参数介绍：

在自旋轨道耦合计算中可以尽量保留 sys. 和 cal. 的参数到 `soi.in` 中，之后设置自旋轨道耦合计算特有的参数即可：

- `soi.spin`：指定计算的自旋性质，non-collinear 表示一般自旋；
- `sys.soi`：表示是否考虑自旋轨道耦合效应；自旋轨道耦合效应需要在 sys.spin=non-collinear 时才会生效；

structure.as 文件参考如下：

```
Total number of atoms
5
Lattice
-2.069  -3.583614  0.000000
 2.069  -3.583614  0.000000
 0.000   2.389075  9.546667
Direct
Bi 0.3990 0.3990 0.6970
Bi 0.6010 0.6010 0.3030
Se 0.0000 0.0000 0.5000
Se 0.2060 0.2060 0.1180
Se 0.7940 0.7940 0.8820
```

然后进行能带计算：输入文件包含参数文件 `soiband.in`，内容如下

```
# task type
task = band
#system related
sys.structure = structure.as
sys.symmetry = true
sys.functional = LDA
#scf related
cal.methods = 2
cal.smearing = 1
cal.ksamping = G
cal.kpoints = [7, 7, 7]
cal.cutoffFactor = 1.5
#band related
cal.iniCharge = ./rho.bin
band.kpointsCoord = [0.00000000,0.00000000,0.00000000,0.00000000,0.00000000,0.
↪50000000,0.50000000,0.50000000,0.00000000,0.00000000,0.00000000,0.00000000,0.
↪50000000,0.00000000,0.00000000]
band.kpointsLabel = [G,Z,F,G,L]
band.kpointsNumber = [20,20,20,20]
band.project = true
#soi related
sys.spin= non-collinear
sys.soi = true
```

soiband.in 输入参数介绍：

在自旋轨道耦合能带计算中，保留自洽计算和自旋轨道耦合计算的参数到 `soiband.in` 中，之后设置能带计算的特有参数即可。

备注:

1. **初始磁矩的设置参考“应用案例-NiO 体系的反铁磁计算”，在 structure.as 文件的第七行设置 Mag 标签即可。**

2.17.2 run 程序运行

准备好输入文件之后，将 soi.in 、soiband.in 和 structure.as 文件上传到服务器上运行，按照结构弛豫中介绍的方法分别执行 *DS-PAW soi.in* 和 *DS-PAW soiband.in* 。

2.17.3 analysis 计算结果分析

根据上述的输入文件，计算完成之后将会得到 DS-PAW.log 、system.json 和 band.json 这 3 个文件。

处理 band.json 的方法同（见 2.3 节）能带计算的方法，DS 处理得到的 bandplot.png 文件如下图（a）所示，另作不考虑自旋轨道耦合的计算，得到的能带图如下（b）所示：

(a)

(b)

从 band.json 可读取 **BandGap** 值，图（a）和图（b）的带隙值分别为 **0.3251** 和 **0.0814** ，可得结论：进行自旋轨道耦合计算可加大价带与导带之间的带隙。

2.18 aimd 分子动力学模拟

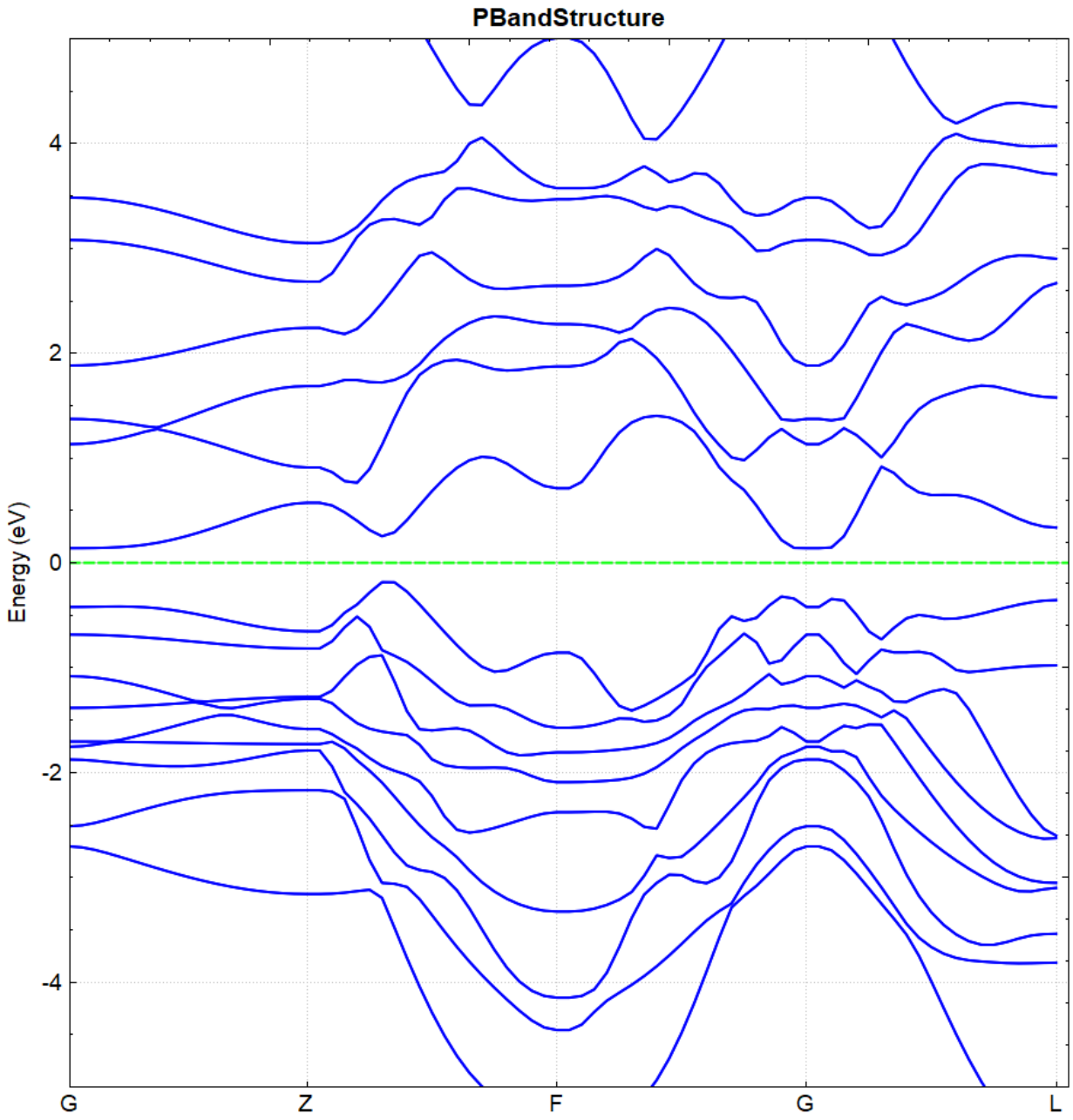
本节将以水分子体系为例，介绍在 DS-PAW 中如何进行分子动力学模拟计算。

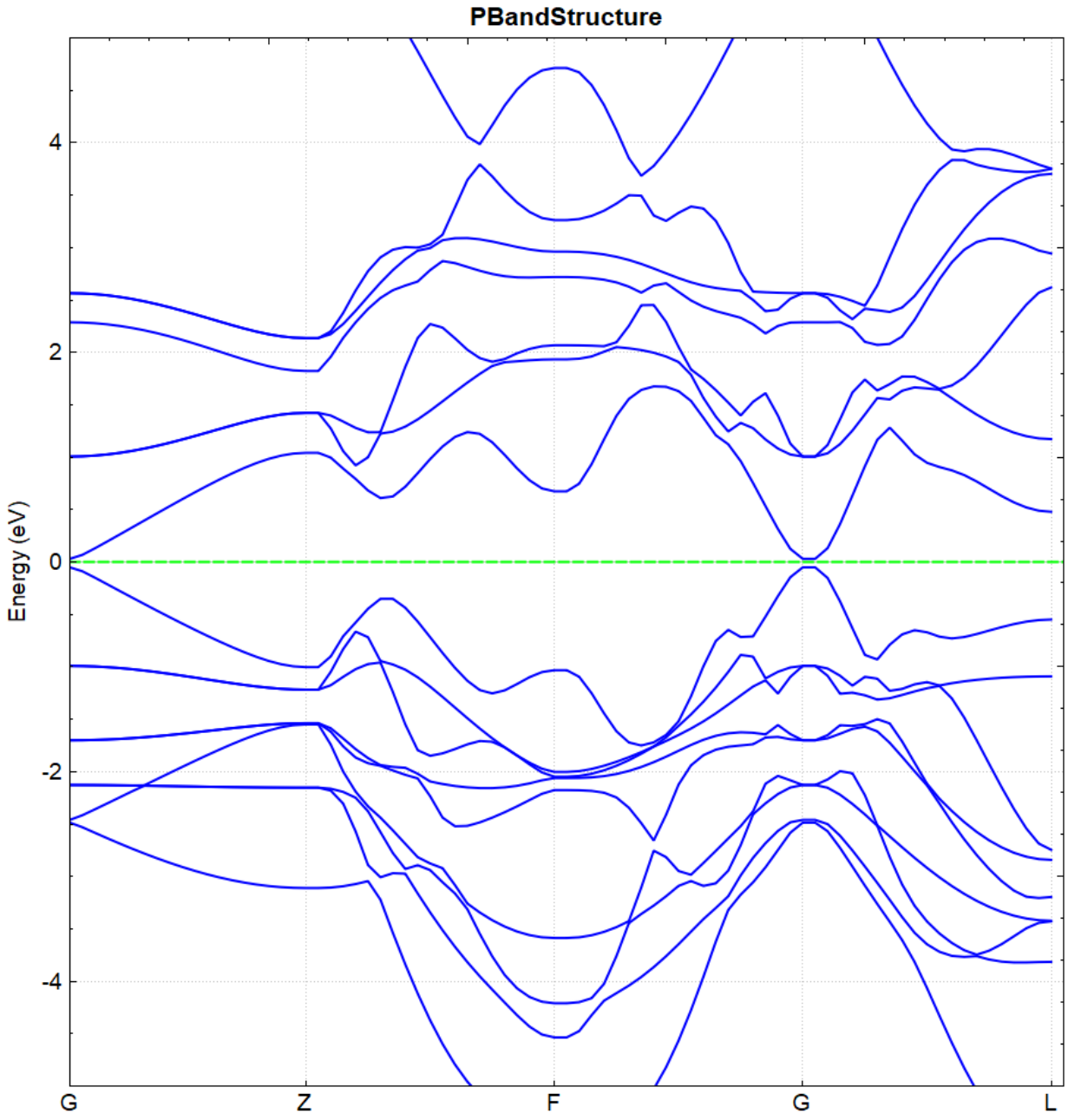
2.18.1 H_2O 分子动力学模拟输入文件

输入文件包含参数文件 aimd.in 和结构文件 structure.as ， aimd.in 如下：

```
# task type
task = aimd
#system related
sys.structure = structure.as
sys.symmetry = false
sys.functional = LDA
sys.spin = none
#scf related
cal.methods = 1
cal.smearing = 2
cal.ksamping = G
cal.kpoints = [1, 1, 1]
cal.cutoffFactor = 1.5
#aimd related
aimd.ensemble = NVE
aimd.timeStep = 1
```

(续下页)


PBandStructure
Energy (eV)
4
2
0
-2
-4
G
Z
F
G
L


PBandStructure
Energy (eV)
4
2
0
-2
-4
G
Z
F
G
L

(接上页)

```
aimd.totalSteps = 1000
aimd.iniTemp = 2000
#outputs
io.charge = false
io.wave = false
```

aimd.in 输入参数介绍：

在分子动力学模拟计算中可以尽量保留 sys. 和 cal. 的参数到 `aimd.in` 中，之后设置分子动力学模拟计算特有的参数即可：

- `task`：本次计算为分子动力学计算，设置 task 为 aimd；
- `aimd.ensemble`：表示分子动力学模拟时选用的系综；
- `aimd.timeStep`：表示分子动力学模拟时的时间步长；
- `aimd.totalSteps`：表示分子动力学模拟的总步数；
- `aimd.iniTemp`：表示分子动力学模拟时的初始温度；

structure.as 文件参考如下：

```
Total number of atoms
3
Lattice
6.35016 0 0
0 6.35016 0
0 0 6.35016
Direct
O 0.00000    0.00000    0.00000
H 0.09167   -0.11917    0.00000
H 0.09167    0.11917    0.00000
```

2.18.2 run 程序运行

准备好输入文件之后，将 `aimd.in` 和 `structure.as` 文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 *DS-PAW aimd.in* 。

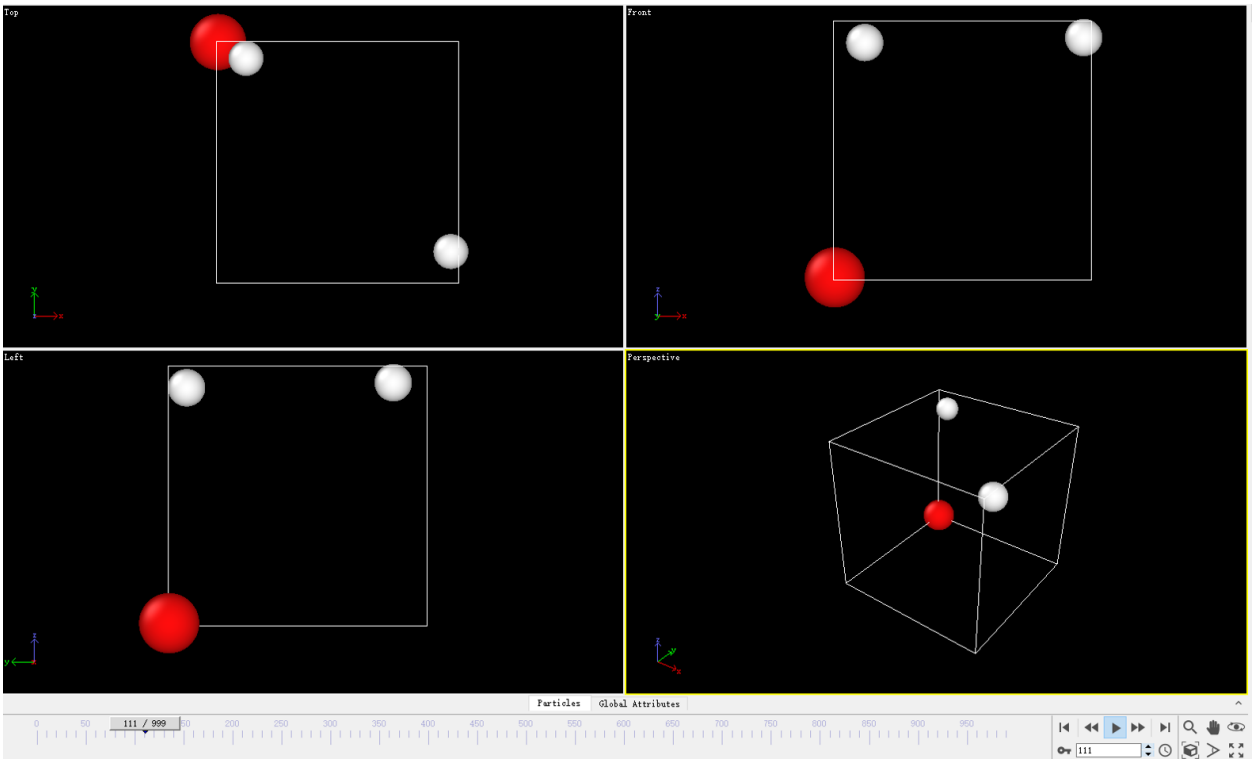
2.18.3 analysis 计算结果分析

根据上述的输入文件，计算完成之后将会得到 `DS-PAW.log` 、`aimd.json` 、`paw_tmp/aimd.tmp` 这 3 个文件。

aimd.json：分子动力学计算完成之后的 **json** 数据文件；此时模拟时间内原子位置、体系能量和温度等数据被保存在 `aimd.json` 中，具体的数据结构详见数据结构解析部分；

paw_tmp/aimd.tmp：分子动力学计算中的轨迹文件，默认 aimd 计算中每 20 个离子步记录一次结构信息；

使用 **Device Studio** 可直接对 `aimd.json` 文件处理出图，其操作步骤为：Simulator-->DS-PAW-->Analysis Plot，选择 `aimd.json` 即可，可根据作图要求自定义设置面板参数。DS 可以播放在模拟过程中分子的运动轨迹，截取其中一帧的结构图如下图所示：



另可使用 **python** 进行数据处理，将 `aimd.json` 转为 **pdb** 文件，具体操作见 **辅助工具使用教程**部分。

备注:

1. **本例设置的系综为 nve，设置 nvt 系综时可选择程序支持的两种控温器，通过 aimd.thermostat 参数设置**;
2. **如需模拟高温退火过程，设置 aimd.ensemble 为 TS，同时通过 aimd.iniTemp 和 aimd.iniTemp 设置初始和末态温度即可**。

2.19 efield 外加电场计算

本节将以硅烯模型为例，介绍在 DS-PAW 中如何进行外加电场计算，分析加电场前后带隙打开情况。

2.19.1 Si 硅烯真空方向外加电场计算输入文件

输入文件包含参数文件 `Efield.in` 和结构文件 `structure.as` ，`Efield.in` 如下：

```
# task type
task = scf
#system related
sys.structure = structure.as
sys.symmetry = true
sys.functional = PBE
sys.spin = none

#scf related
cal.sigma = 0.1
cal.cutoff = 520
cal.ksamping = G
cal.kpoints = [9, 9, 1]

scf.convergence = 1e-5

#outputs
io.charge = false
```

(续下页)

(接上页)

```
io.wave = false
io.band = true

corr.dipol=true
corr.dipolDirection = c
corr.dipolEfield = 0.2

band.kpointsLabel = [G,M,K,G]
band.kpointsCoord = [0.00000000,0.00000000,0.00000000,0.50000000,0.00000000,0.
↪00000000,0.33333333,0.33333333,0.00000000,0.00000000,0.00000000,0.00000000]
band.kpointsNumber = [100,100,100]
```

Efield.in 输入参数介绍：

该计算是在一步能带计算的基础上外加电场，除能带计算的基本参数，新增参数为下：

- `corr.dipolEfield`：设置外加电场的大小，注意该参数只在 `corr.dipol = true` 和设置 `corr.dipolDirection` 的情况下生效；

structure.as 文件参考如下：

```
Total number of atoms
2
Lattice
  3.860000   0.000000   0.000000
 -1.930000   3.342860   0.000000
  0.000000   0.000000  26.460000
Direct
Si   0.333333   0.166667   0.396825
Si   0.666758   0.833380   0.379216
```

2.19.2 run 程序运行

准备好输入文件之后，将 `Efield.in` 和 `structure.as` 文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 *DS-PAW Efield.in* 。

2.19.3 analysis 计算结果分析

根据上述的输入文件，计算完成之后将会得到 `DS-PAW.log` 、`band.json` 这 2 个文件。

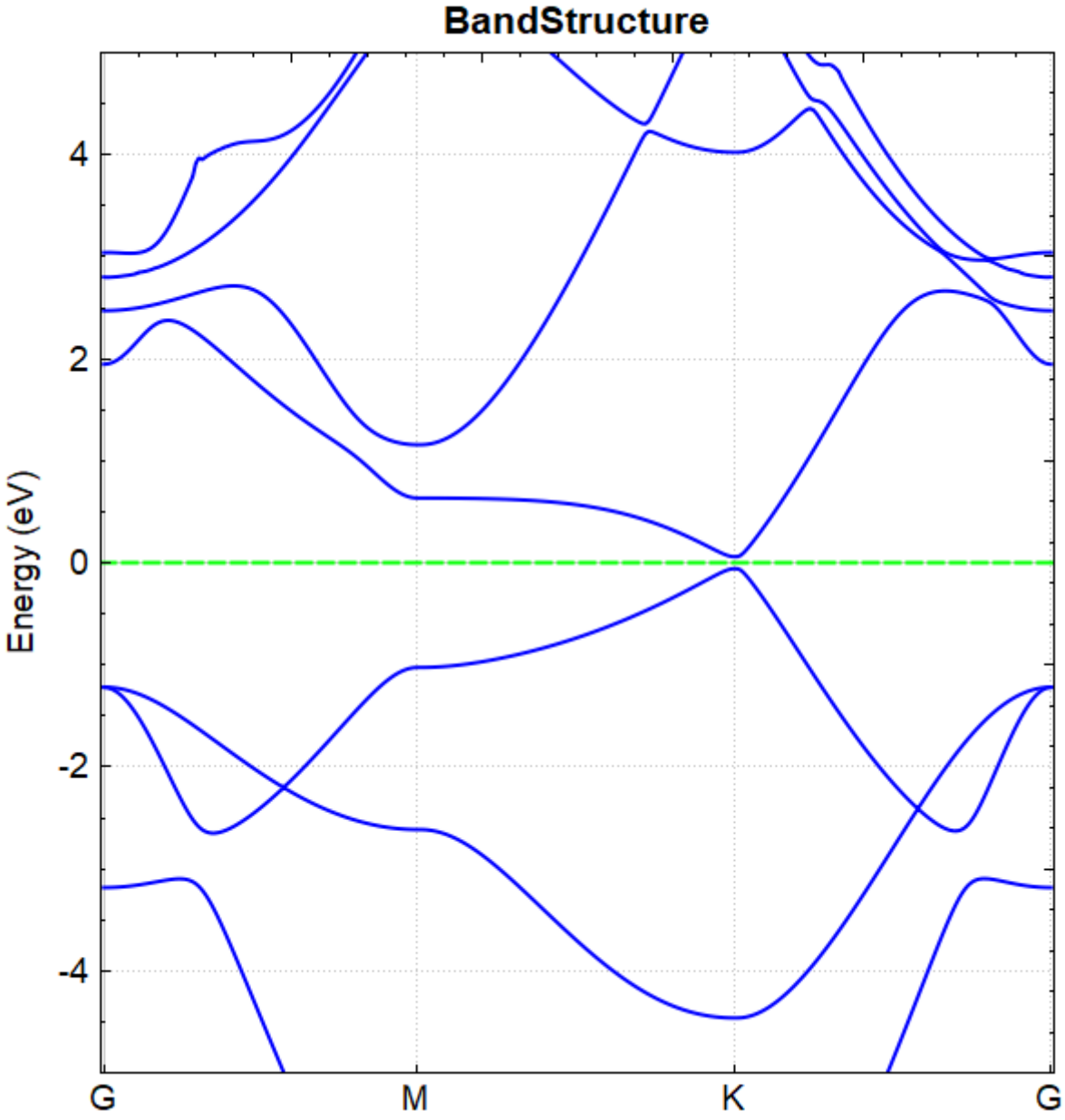
band.json： 计算完成之后的 **json** 数据文件；

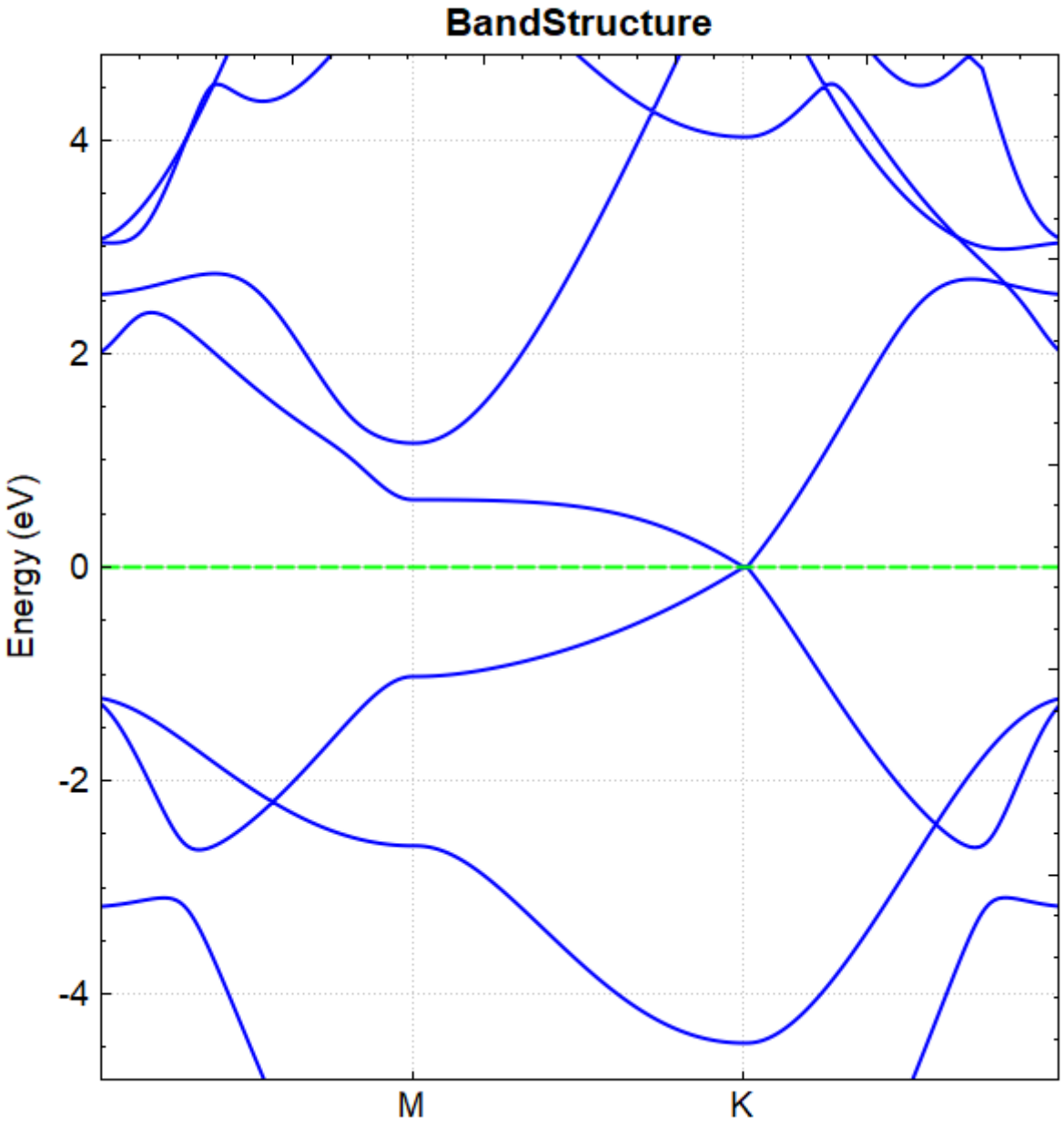
此例中参数 `corr.dipolEfield = 0.2` ，即外加电场的大小为 **0.2** eV/Å，在该电场下进行能带计算得到的能带图如图（a）所示，

(a)

若设置参数 `corr.dipolEfield = 0` 重复以上计算，即在无电场的情况下进行能带计算得到的能带图如图（b）所示，

(b)


BandStructure
Energy (eV)
4
2
0
-2
-4
G
M
K
G


BandStructure
Energy (eV)
4
2
0
-2
-4
M
K

对比图（a）和图（b）可得结论：通过外加电场可以打开硅烯的带隙。从 band.json 文件可读出加电场与不加电场 **BandGap** 的值分别为 **0.1176** 和 **0.0010** 。

备注:

1. **外加电场的单位 eV/Å 为原子受力的单位**

2.20 ferri 铁电计算

本节将以 HfO_2 为例，介绍在 DS-PAW 中如何使用现代极化理论进行铁电计算，分析 HfO_2 的铁电极化。

2.20.1 HfO_2 铁电计算输入文件

输入文件包含参数文件 polarization.in 和结构文件 structure.as ，polarization.in 如下：

```
# task type
task = scf
#system related
sys.structure = structure.as
sys.symmetry = true
sys.functional = PBE
sys.spin = none

#scf related
cal.methods = 3
cal.smearing = 4
cal.sigma = 0.05
cal.cutoff = 520
cal.ksamping = MP
cal.kpoints = [4, 4, 4]

scf.convergence = 1e-5

#outputs
io.charge = false
io.wave = false
io.polarization = true
```

polarization.in 输入参数介绍：

该计算是在自洽计算的基础上进行铁电计算，除自洽计算的基本参数，新增参数为下：

- io.polarization：在自洽计算中打开控制铁电计算的开关，即可快速实现铁电计算；

structure.as 文件参考如下：

```
Total number of atoms
12
Lattice
 5.04621935 0.00000000 0.00000000
 0.00000000 5.07315250 0.00000000
 0.00000000 0.00000000 5.25768906
Cartesian
Hf 1.34815269 1.22145223 0.17639072
Hf 1.34815269 3.75802848 2.45245381
Hf 3.69806666 1.22145223 2.80523525
Hf 3.69806666 3.75802848 5.08129834
O 0.35195212 1.93667285 1.92589951
O 0.35195212 4.47324910 0.70294502
O 2.32678304 2.48829365 3.85528784
O 2.32678304 5.02486990 4.03124575
O 2.71943630 5.02486990 1.40240122
O 2.71943630 2.48829365 1.22644331
O 4.69426723 1.93667285 4.55474404
O 4.69426723 4.47324910 3.33178955
```

2.20.2 run 程序运行

准备好输入文件之后，将 `polarization.in` 和 `structure.as` 文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 *DS-PAW polarization.in* 。

2.20.3 analysis 计算结果分析

根据上述的输入文件，计算完成之后将会得到 `DS-PAW.log` 、`polarization.json` 和 `polarization.txt` 这 3 个文件。

polarization.json： 铁电极化计算完成之后的 **json** 数据文件，电子、离子贡献的极化部分及总的极化量子数存储在该文件中，具体文件解析见输出文件格式说明部分；

polarization.txt : 铁电极化计算完成之后的 **txt** 文本文件，该文件写入极化数据，与 `polarization.json` 文件数据一致，便于用户快速获取信息。

从 *polarization.txt* 文件可得 HfO_2 的铁电极化数据如下所示：

Total(x y z) ((μC/cm^2))		
-0.000001	-8.715113	0.000000
Quantum(x y z) (μC/cm^2)		
60.067225	60.387821	62.584436

通过极化量子的周期性换算，可得 y 方向上 HfO_2 的极化数为 **51.672 μC/cm^2** ，与文献报道结果 [2] **53 μC/cm^2** 一致。

2.21 bader 电荷计算

本节将以 NaCl 晶体为例，介绍在 DS-PAW 中如何进行 bader 电荷计算，分析 NaCl 体系中各原子的价态分布。

2.21.1 $NaCl$ 晶体 Bader 电荷计算输入文件

输入文件包含参数文件 `bader.in` 和结构文件 `structure.as` ，`bader.in` 如下：

bader.in 文件参考如下：

```
# task type
task = scf
#system related
sys.structure = structure.as
sys.symmetry = false
sys.functional = PBE
sys.spin = none

#scf related
cal.methods = 1
cal.smearing = 1
cal.ksamping = G
cal.kpoints = [10, 10, 10]
cal.cutoff = 500
#outputs
io.charge = true
io.wave = false
io.bader = true
```

bader.in 输入参数介绍：

该计算是在自洽计算的基础上进行 bader 电荷计算，除自洽计算的基本参数，新增参数为下：

- `io.bader`：在自洽计算中打开控制 bader 电荷计算的开关，即可快速实现 bader 电荷计算；

structure.as 文件参考如下：

```
Total number of atoms
 8
 Lattice
  5.68452692 0.00000000 0.00000000
  0.00000000 5.68452692 0.00000000
  0.00000000 0.00000000 5.68452692
 Cartesian
 Na 4.26339519 1.42113173 1.42113173
 Na 1.42113173 4.26339519 1.42113173
 Na 1.42113173 1.42113173 4.26339519
 Na 4.26339519 4.26339519 4.26339519
 Cl 1.42113173 1.42113173 1.42113173
 Cl 4.26339519 4.26339519 1.42113173
 Cl 4.26339519 1.42113173 4.26339519
 Cl 1.42113173 4.26339519 4.26339519
```

备注：

1. **io.bader = true 时，io.charge 必须设置为 true**

2.21.2 run 程序运行

准备好输入文件之后，将 `bader.in` 和 `structure.as` 文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 *DS-PAW bader.in* 。

2.21.3 analysis 计算结果分析

根据上述的输入文件，计算完成之后将会得到 `DS-PAW.log` 、`bader.json` 、`bader.txt` 这 3 个文件。

bader.json： bader 电荷计算完成之后的 **json** 数据文件，各原子的 Bader 电荷等数据存储在该文件中，具体的数据结构详见 输出文件格式说明部分；

bader.txt : bader 电荷计算完成之后的 **txt** 文本文件，该文件写入 bader 电荷数据，与 `bader.json` 文件数据一致，便于用户快速获取信息。

bader.txt 文本内容如下所示，bader 电荷分析得到的数据与 utexas 大学的 Henkelman 小组得到的数据吻合。https://theory.cm.utexas.edu/henkelman/code/bader/

Total number of valence electronics: 64

Element	X	Y	Z	Charge	AtomicVolume	MinDistance
Cl	0.25	0.25	0.25	7.83165	34.846	1.59877
Cl	0.75	0.75	0.25	7.87519	36.8613	1.59877
Cl	0.75	0.25	0.75	7.84614	35.5159	1.59877
Cl	0.25	0.75	0.75	7.86064	36.1858	1.59877
Na	0.75	0.25	0.25	8.1466	10.07	1.11292
Na	0.25	0.75	0.25	8.1466	10.07	1.11292
Na	0.25	0.25	0.75	8.1466	10.07	1.11292
Na	0.75	0.75	0.75	8.1466	10.07	1.11292

2.22 bandunfolding 能带反折叠计算

本节将以 Cu_3Au 体系为例，介绍在 DS-PAW 中如何进行能带反折叠计算，分析 Cu_3Au 反折叠的能带图。

2.22.1 Cu_3Au 能带反折叠计算输入文件

能带反折叠计算需两步法完成能带计算，因此输入文件包含参数文件 scf.in 、bandunfolding.in 和结构文件 structure.as ，

scf.in 如下：

```
task = scf

sys.structure = structure.as
sys.symmetry = true
sys.functional = PBE
sys.spin = none

cal.methods = 1
cal.smearing = 1
cal.ksamping = MP
cal.kpoints = [3, 3, 3]
cal.cutoff = 500

scf.convergence = 1.0e-05

io.charge = true
io.wave = false
```

bandunfolding.in 如下：

```
task = band
cal.iniCharge = ./rho.bin

sys.structure = structure.as
sys.symmetry = true
sys.functional = PBE
sys.spin = none

cal.methods = 1
cal.smearing = 1
cal.ksamping = MP
cal.kpoints = [3, 3, 3]
cal.cutoff = 500

scf.convergence = 1.0e-05

band.unfolding = true
band.primitiveUVW=[0.0, 0.5, 0.5, 0.5, 0.0, 0.5, 0.5, 0.5, 0.0]
band.kpointsLabel= [R,G,X]
band.kpointsCoord= [0.5, 0.5, 0.5, 0.0, 0.0, 0.0, 0.5, 0.0, 0.5]
band.kpointsNumber= [101, 101]

io.charge = false
io.wave = false
```

bandunfolding.in 输入参数介绍：

能带反折叠计算是在能带计算的基础上完成的，且能带计算必须通过两步法完成。除能带计算的基本参数，新增参数为下：

- band.unfolding：在能带计算中打开能带反折叠计算的开关，即可实现能反折叠计算；

- band.primitiveUVW：超胞的晶格常数乘上 UVW 系数等于原胞的晶格矢量，用于控制能带反折叠的参数；

structure.as 文件参考如下：

```
Total number of atoms
4
Lattice
        3.7530000210          0.0000000000          0.0000000000
        0.0000000000          3.7530000210          0.0000000000
        0.0000000000          0.0000000000          3.7530000210
Direct
Au      0.000000000          0.000000000          0.000000000
Cu      0.000000000          0.500000000          0.500000000
Cu      0.500000000          0.000000000          0.500000000
Cu      0.500000000          0.500000000          0.000000000
```

2.22.2 run 程序运行

准备好输入文件之后，将 scf.in 、bandunfolding.in 和 structure.as 文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 *DS-PAW scf.in* ，自洽计算完成后执行 *DS-PAW bandunfolding.in* 。

2.22.3 analysis 计算结果分析

根据上述的输入文件，计算完成之后将会得到 DS-PAW.log 、band.json 、log 这 3 个文件。

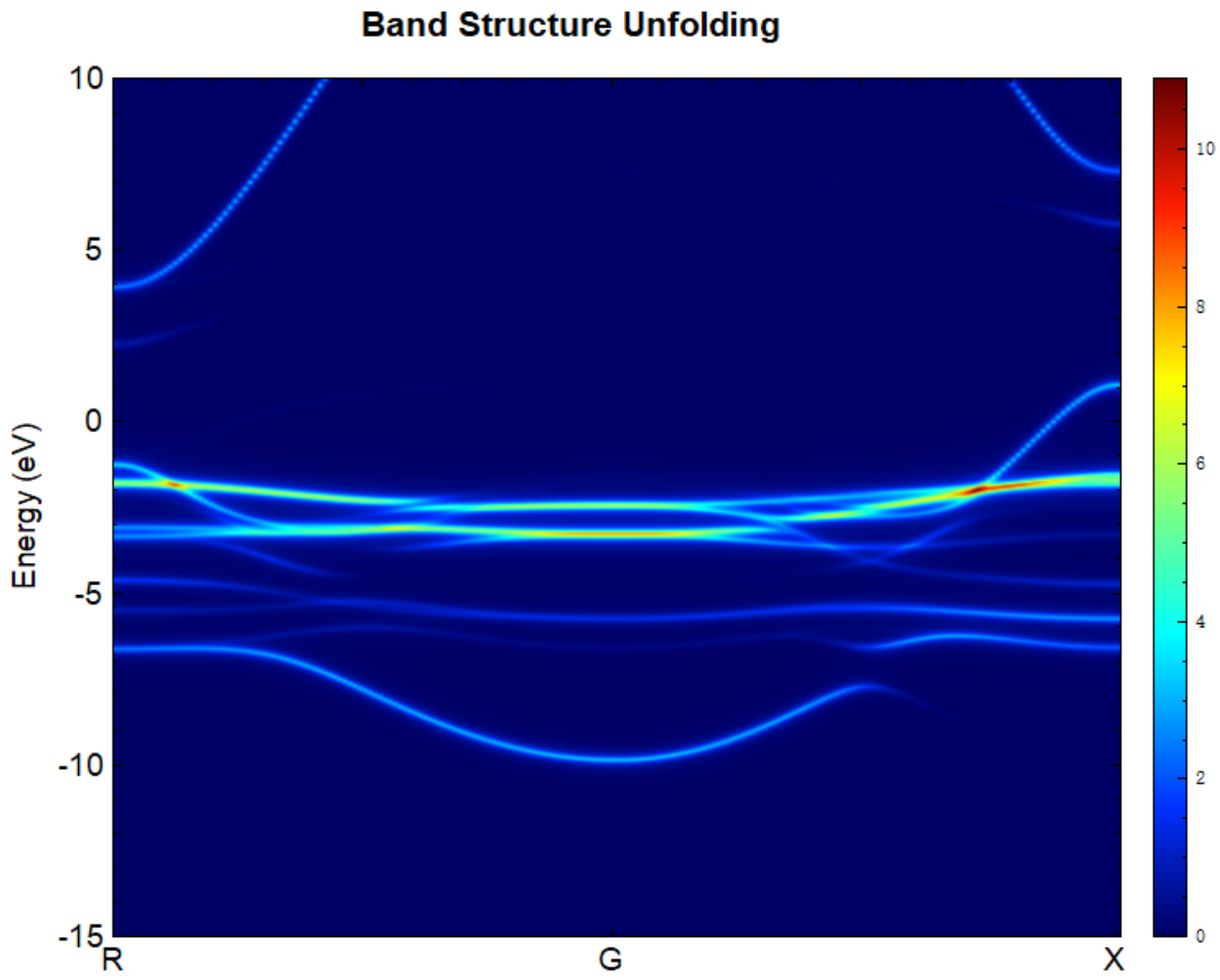
band.json ： 计算完成之后的 **json** 数据文件，能带反折叠计算所得 json 文件与普通能带计算不同，会新增 **UnfoldingBandInfo** 部分，具体结构解析见输出文件格式说明部分。

使用 **Device Studio** 可直接对 band.json 文件处理出图，其操作步骤为：Simulator-->DS-PAW-->Analysis Plot，选择 band.json 即可，可根据作图要求自定义设置面板参数。

该例得到的能带图如下所示，与文献报道结果 [3] 一致。

2.23 epsilon 介电常数计算

本节将以 Si 体系为例，介绍在 DS-PAW 中如何进行介电常数计算。


Band Structure Unfolding
Energy (eV)
10
5
0
-5
-10
-15
R
G
X
10
8
6
4
2
0

2.23.1 Si 介电常数计算输入文件

输入文件包含参数文件 epsilon.in 和结构文件 structure.as，epsilon.in 如下：

```
# task type
task = epsilon
#system related
sys.structure = structure.as
sys.symmetry = true
sys.functional = PBE
sys.spin = none

#scf related
cal.methods = 1
cal.smearing = 1
cal.ksamping = G
cal.kpoints = [5, 5, 5]
cal.cutoff = 500
scf.convergence = 1.0e-7
```

epsilon.in 输入参数介绍：

介电常数的计算可通过直接指定 **task** 完成，新增参数为下：

- task = epsilon：通过控制任务类型完成介电常数的计算；

备注： 在 task = phonon 且 phonon.method = dfpt 时也可完成介电常数的计算，通过添加 phonon.dfptEpsilon = true 参数即可。

structure.as 文件参考如下：

```
Total number of atoms
8
Lattice
 5.43070000 0.00000000 0.00000000
 0.00000000 5.43070000 0.00000000
 0.00000000 0.00000000 5.43070000
Cartesian
Si 0.67883750 0.67883750 0.67883750
Si 3.39418750 3.39418750 0.67883750
Si 3.39418750 0.67883750 3.39418750
Si 0.67883750 3.39418750 3.39418750
Si 2.03651250 2.03651250 2.03651250
Si 4.75186250 4.75186250 2.03651250
Si 4.75186250 2.03651250 4.75186250
Si 2.03651250 4.75186250 4.75186250
```

2.23.2 run 程序运行

准备好输入文件之后，将 `epsilon.in` 和 `structure.as` 文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 *DS-PAW epsilon.in* 。

2.23.3 analysis 计算结果分析

根据上述的输入文件，计算完成之后将会得到 `DS-PAW.log` 、`epsilon.json` 、`epsilon.txt` 这 3 个文件。

epsilon.json： 介电常数计算完成之后的 **json** 数据文件，介电常数、波恩有效电荷等数据会写入该文件，具体的数据结构详见 输出文件格式说明部分；

epsilon.txt : 介电常数计算完成之后的 **txt** 文本文件，该文件写入介电常数相关数据，与 `epsilon.json` 文件数据一致，便于用户快速获取信息。

从 *epsilon.txt* 文件中可获取如下数据：

Total Part		
13.309902	0.000000	-0.000000
-0.000000	13.309902	-0.000000
-0.000000	0.000000	13.309902

分析上表可得该体系的介电常数为 **13.309902** ，与文献报道值 [4] **13.31** 一致。

2.24 piezo 压电张量计算

本节将以 AlN 体系为例，介绍在 DS-PAW 中如何进行压电张量计算，得到材料的压电系数 $e_{33}(0)$。

2.24.1 AlN 压电张量计算输入文件

输入文件包含参数文件 `piezo.in` 和结构文件 `structure.as` ， `piezo.in` 如下：

```
task = epsilon
#system related
sys.structure = structure.as
sys.symmetry = true
sys.functional = PBE
sys.spin = none

#scf related
```

(续下页)

(接上页)

```
cal.methods = 1
cal.smearing = 1
cal.ksamping = G
cal.kpoints = [10, 10, 10]
cal.cutoffFactor = 1.5
scf.convergence = 1.0e-7

#outputs
io.charge = false
io.wave = false
```

piezo.in 输入参数介绍：

- `task`：本次计算为压电张量计算，设置 task 为 epsilon；
- `scf.convergence`：介电张量计算建议提高电子收敛的精度，此处设置为 1.0e-7；

structure.as 文件参考如下：

```
Total number of atoms
8
Lattice
 3.11606630 0.00000000 0.00000000
 0.00000000 5.39683518 0.00000000
 0.00000000 0.00000000 5.00770902
Cartesian
Al 0.00000000 3.59735137 0.00946380
Al 0.00000000 1.79945276 2.51320124
Al 1.55803315 0.89899597 0.00945662
Al 1.55803315 4.49786165 2.51308138
N 0.00000000 3.59851112 1.91845914
N 0.00000000 1.79831356 4.42266820
N 1.55803315 0.90013952 1.91851680
N 1.55803315 4.49672497 4.42258192
```

2.24.2 run 程序运行

准备好输入文件之后，将 `piezo.in` 和 `structure.as` 文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 *DS-PAW piezo.in* 。

2.24.3 analysis 计算结果分析

根据上述的输入文件，计算完成之后将会得到 `DS-PAW.log` 、`epsilon.json` 、`epsilon.txt` 这 3 个文件。

epsilon.json： 介电常量计算完成之后的 **json** 数据文件；介电常数、压电张量矩阵、波恩有效电荷的数据被保存在 `epsilon.json` 中，具体的数据结构详见数据结构解析部分；

epsilon.txt：压电计算完成之后的 **txt** 文本文件，该文件写入压电相关数据，与 `epsilon.json` 文件数据一致，便于用户快速获取信息。

从 *epsilon.txt* 文件中可获取如下数据：

Piezoelectric Tensor (C/m^2)(Row: x y z Column: XX YY ZZ XY YZ ZX)					
Electronic Part					
0.000000	0.000000	0.000000	0.000006	0.000000	0.336610
-0.000001	0.000007	0.000003	0.000000	0.336662	0.000000
0.266339	0.265888	-0.419569	0.000000	-0.000014	0.000000
Ionic Part:					
-0.000004	0.000002	0.000002	0.000032	-0.000000	-0.681702
-0.000163	-0.000239	0.000314	-0.000000	-0.699012	-0.000000
-0.911456	-0.913265	1.943887	-0.000000	-0.000633	-0.000000
Total Part:					
-0.000004	0.000002	0.000002	0.000039	-0.000000	-0.345092
-0.000164	-0.000232	0.000317	-0.000000	-0.362350	-0.000000
-0.645117	-0.647377	1.524318	-0.000000	-0.000647	-0.000000

分析上表可得压电张量电子贡献部分 $e_{33}(0)$ 的数值为 **-0.419569** C/m^2 ，总的压电张量 e_{33} 的数值为 **1.524318** C/m^2 ，与文献参考值 [5] **-0.47** C/m^2 和 **1.46** C/m^2 接近。

2.25 fixcell 固定基矢弛豫计算

本节将以 MoS_2 体系为例，介绍在 DS-PAW 中如何进行固定晶格弛豫计算。

2.25.1 MoS_2 固定基矢弛豫计算输入文件

输入文件包含参数文件 `relax.in` 和结构文件 `structure.as` ， `relax.in` 如下：

```
# task type
task = relax
#system related
sys.structure = structure.as
sys.symmetry = false
sys.functional = PBE
sys.spin = none

#scf related
cal.methods = 1
cal.smearing = 1
cal.ksamping = G
cal.cutoff = 500
cal.kpoints = [19, 19, 5]
#relax related
relax.freedom = all
relax.convergence =  0.05
relax.methods = CG
```

structure.as 文件参考如下：

```
Total number of atoms
6
Lattice  Fix_x Fix_y Fix_z
3.19031572 0.00000000 0.00000000   F T T
-1.59515786 2.76289446 0.00000000  F F T
0.00000000 0.00000000 14.87900448  T T T
Cartesian
S 0.00000000 1.84193052 12.72413785
S 1.59515943 0.92096386 5.28463561
S 0.00000000 1.84193052 9.59436887
S 1.59515943 0.92096386 2.15486663
Mo 1.59515943 0.92096386 11.15925336
Mo 0.00000000 1.84193052 3.71975112
```

structure.as 标签设置介绍：

固定晶胞维度进行弛豫计算需在 *structure.as* 文件中新增固定标签，类似于固定原子弛豫的设置（在原子坐标后添加 Fix 标签），固定基矢需在 *structure.as* 的第三行 **Lattice** 后添加 Fix 标签，本案例的标签对应固定晶胞的 c 边和 a 边的 y、z 方向、b 边的 z 方向。

2.25.2 run 程序运行

准备好输入文件之后，将 `relax.in` 和 `structure.as` 文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 *DS-PAW relax.in* 。

2.25.3 analysis 计算结果分析

根据上述的输入文件，计算完成之后将会得到 `DS-PAW.log` 、`relax.json` 这 2 个文件。

relax.json： 计算完成之后的 **json** 数据文件；

将 *relax.json* 拖入 Device Studio 查看结构，或直接查看 **paw_tmp** 目录下的 relax.tmp 文件，可得弛豫结束后的结构数据如下：

```
Total number of atoms
6
Lattice
   3.19696732   0.00000000   0.00000000
  -1.59848077   2.76865753   0.00000000
   0.00000000   0.00000000  14.87900448
 Direct
 Mo   0.66666701   0.33333316   0.74999995
 Mo   0.33333340   0.66666675   0.24999997
 S   0.33333340   0.66666666   0.85535854
 S   0.66666686   0.33333303   0.35535875
 S   0.33333367   0.66666699   0.64464148
 S   0.66666708   0.33333333   0.14464130
```

通过对比可得弛豫前 a = b = **3.19031572** ，弛豫后变成了 a = b = **3.19696732** ，而 c = **14.87900448** 不变。

2.26 thermal 声子热力学性质计算

本节将以 Si 体系为例，介绍在 DS-PAW 中如何进行声子热力学性质计算。

2.26.1 Si 声子热力学性质计算输入文件

输入文件包含参数文件 `phonon-thermal.in` 和结构文件 `structure.as`，`phonon-thermal.in` 如下：

```
# task type
task = phonon
#system related
sys.structure = structure.as
sys.symmetry = true
sys.functional = PBE
sys.spin = none

#scf related
cal.methods = 1
cal.smearing = 1
cal.ksamping = G
cal.kpoints = [5, 5, 5]
cal.cutoffFactor = 1.5
scf.convergence = 1.0e-7
#phonon related
phonon.structureSize = [2,2,2]
phonon.type =dos
phonon.qpoints = [31,31,31]
phonon.method = dfpt

phonon.thermal=true
phonon.thermalRange = [0,1000,10]
```

phonon-thermal.in 输入参数介绍：

- `phonon.thermal`：该参数为控制热力学计算的开关；
- `phonon.thermalRange`：该参数可指定热力学计算的温度范围及数据存储间隔；

structure.as 文件参考如下：

```
Total number of atoms
2
Lattice
0.00 2.75 2.75
2.75 0.00 2.75
2.75 2.75 0.00
Direct
Si -0.125000000 -0.125000000 -0.125000000
Si 0.125000000 0.125000000 0.125000000
```

2.26.2 run 程序运行

准备好输入文件之后，将 `phonon-thermal.in` 和 `structure.as` 文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 *DS-PAW phonon-thermal.in* 。

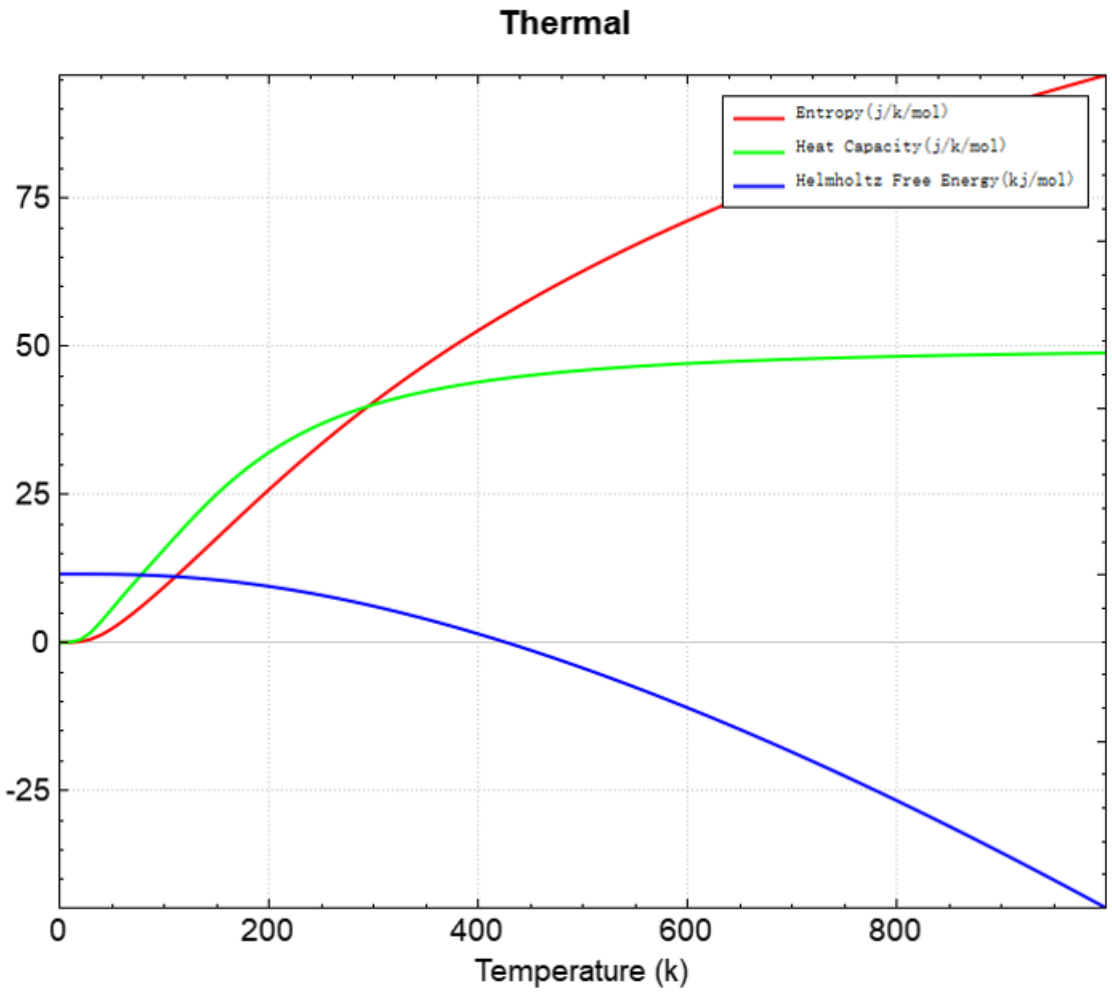
2.26.3 analysis 计算结果分析

根据上述的输入文件，计算完成之后将会得到 `DS-PAW.log` 、`phonon.json` 这 2 个文件。

phonon.json：计算完成之后生成的 json 文件，打开热力学计算的开关，生成的 `phonon.json` 文件中会写入 **ThermalInfo** 数据，具体解析见输出文件格式说明部分。

使用 **Device Studio** 可直接对 `phonon.json` 文件处理出图，其操作步骤为：Simulator-->DS-PAW-->Analysis Plot，选择 `phonon.json` 即可，可根据作图要求自定义设置面板参数。

分析热力学数据，可得熵、热容、亥姆霍兹自由能随温度变化的曲线如下所示，与 phonony git 仓库展示的结果一致：https://phonopy.github.io/phonopy/examples#thermal-properties



2.27 solid state NEB 计算

本节将以 HfZrO 体系为例，介绍在 DS-PAW 中如何放开晶胞弛豫进行 solid state NEB 计算。

2.27.1 $HfZrO$ Solid state NEB 计算输入文件

输入文件包含参数文件 `ssneb.in` 和结构文件 `structure.as` ， `ssneb.in` 如下：

```
task = neb

sys.structure = structure.as
sys.functional  = LDA
sys.spin = none
sys.symmetry = false

cal.ksamping = G
cal.kpoints = [10,10,10]
cal.cutoff = 500
cal.methods = 1
cal.smearing = 1
cal.sigma = 0.05

scf.mixType = Broyden
scf.mixBeta = 0.4
scf.convergence = 1e-6
scf.max = 300

neb.springK = 5
neb.images = 6
neb.iniFin = true
neb.method = QM2
neb.convergence = 0.01
neb.max = 500
neb.freedom = all

io.wave = false
io.charge = false
```

ssneb.in 输入参数介绍：

- `neb.freedom`：该参数控制过渡态弛豫的维度，设置为 all 对应弛豫晶胞大小；
- `neb.method`：该参数指定过渡态搜寻的方法，当弛豫晶胞大小时该参数仅限于 QM2 和 FIRE;

structure.as 需提供多个，初态结构 `structure00.as` 参考如下

```
Total number of atoms
12
Lattice
 5.00209138 0.00000009 0.00000004
```

(续下页)

(接上页)

```
 0.00000009 5.00209143 -0.00000004
 0.00000004 -0.00000004 5.07896990
Cartesian
Hf 2.50104558 2.50104575 0.00000000
Hf 0.00000000 0.00000000 0.00000000
O 3.75156841 1.25052303 1.47285183
O 3.75156857 3.75156869 1.04735062
O 1.25052293 1.25052297 3.60611823
O 1.25052286 3.75156867 4.03161932
O 1.25052287 3.75156860 1.47285187
O 1.25052275 1.25052294 1.04735054
O 3.75156850 1.25052287 4.03161945
O 3.75156850 3.75156869 3.60611821
Zr 2.50104577 0.00000000 2.53948497
Zr 0.00000000 2.50104594 2.53948491
```

末态结构 `structure07.as` 参考如下

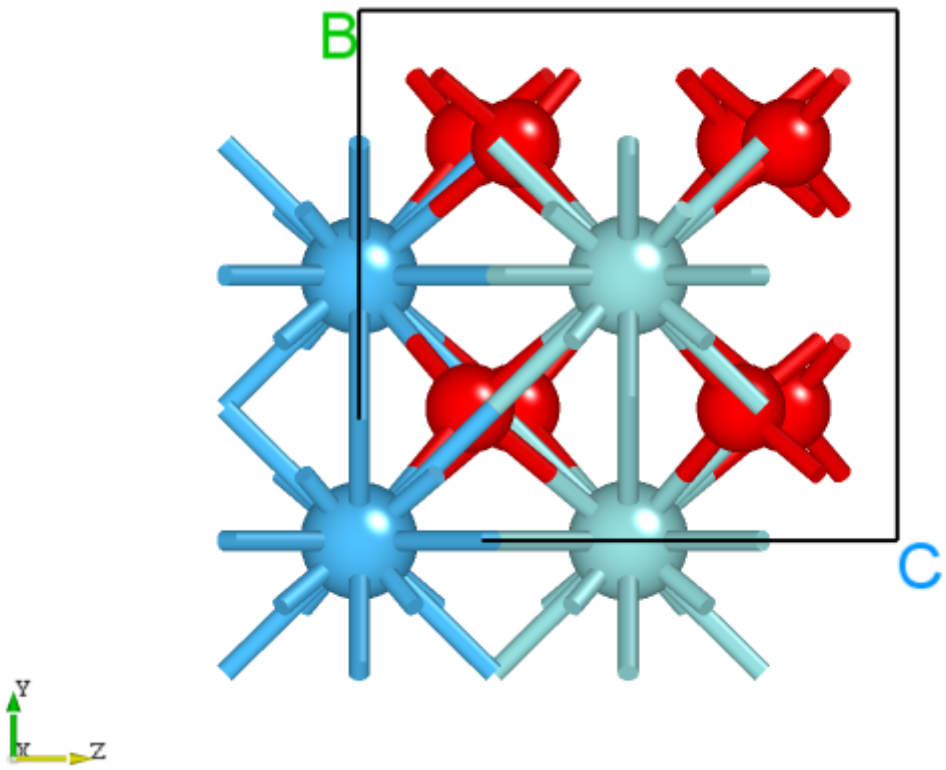
```
Total number of atoms
12
Lattice
 4.98221520 -0.00002552 0.00036684
 -0.00002562 4.99587652 0.00005905
 0.00039053 0.00006126 5.18258321
Cartesian
Hf 2.30823006 2.49975412 0.04967381
Hf 0.00919001 0.00195723 0.38722458
O 4.03365086 0.66419181 2.12958714
O 4.00001549 3.18954023 0.89210846
O 0.95871628 1.24120307 4.04442128
O 0.94984693 3.74053908 4.19050825
O 1.35895285 3.73907584 1.57483409
O 1.36804279 1.24264997 1.42944278
O 3.29999107 0.69159253 4.72728663
O 3.26626721 3.16200890 3.48972595
Zr 2.31915914 0.00841995 2.97686955
Zr 4.98082249 2.50639160 2.64290889
```

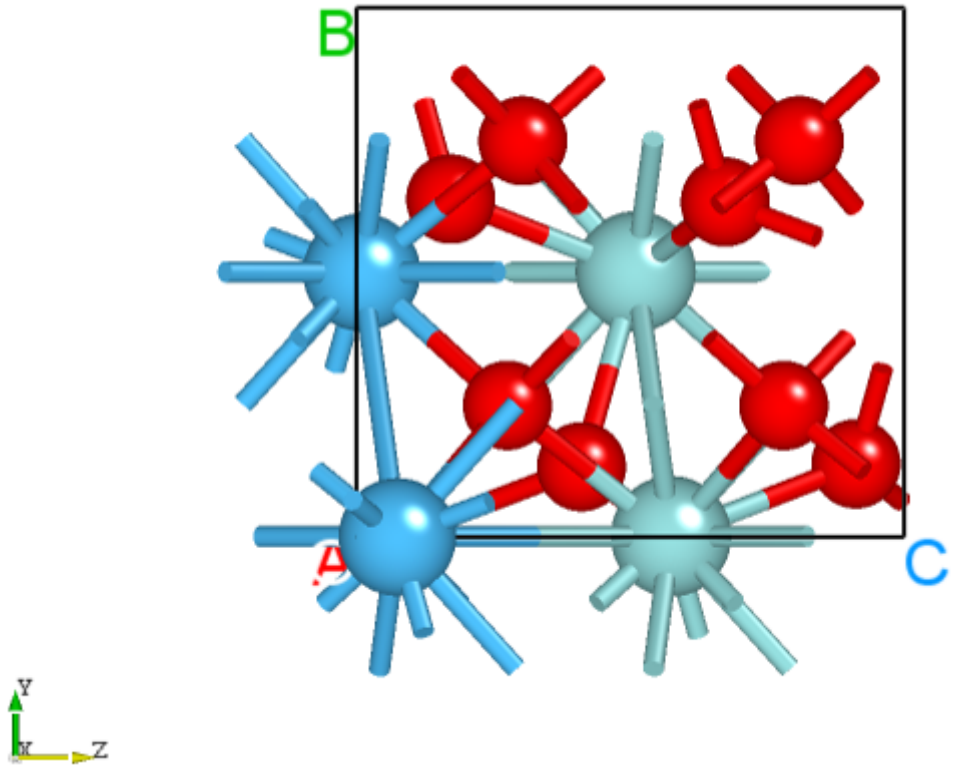
初末态构型在 **Device Studio** 中显示如下：

初态 T 相构型

末态 F 相构型

备注： 在 `neb.freedom = all` 时，`neb.method` 只能设置为 **QM2** 或 **FIRE**


B
C
Y
Z


B
A
C
Y
Z

2.27.2 run 程序运行

准备好输入文件之后，将 `ssneb.in` 和 `structure.as` 文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 *DS-PAW ssneb.in* 。

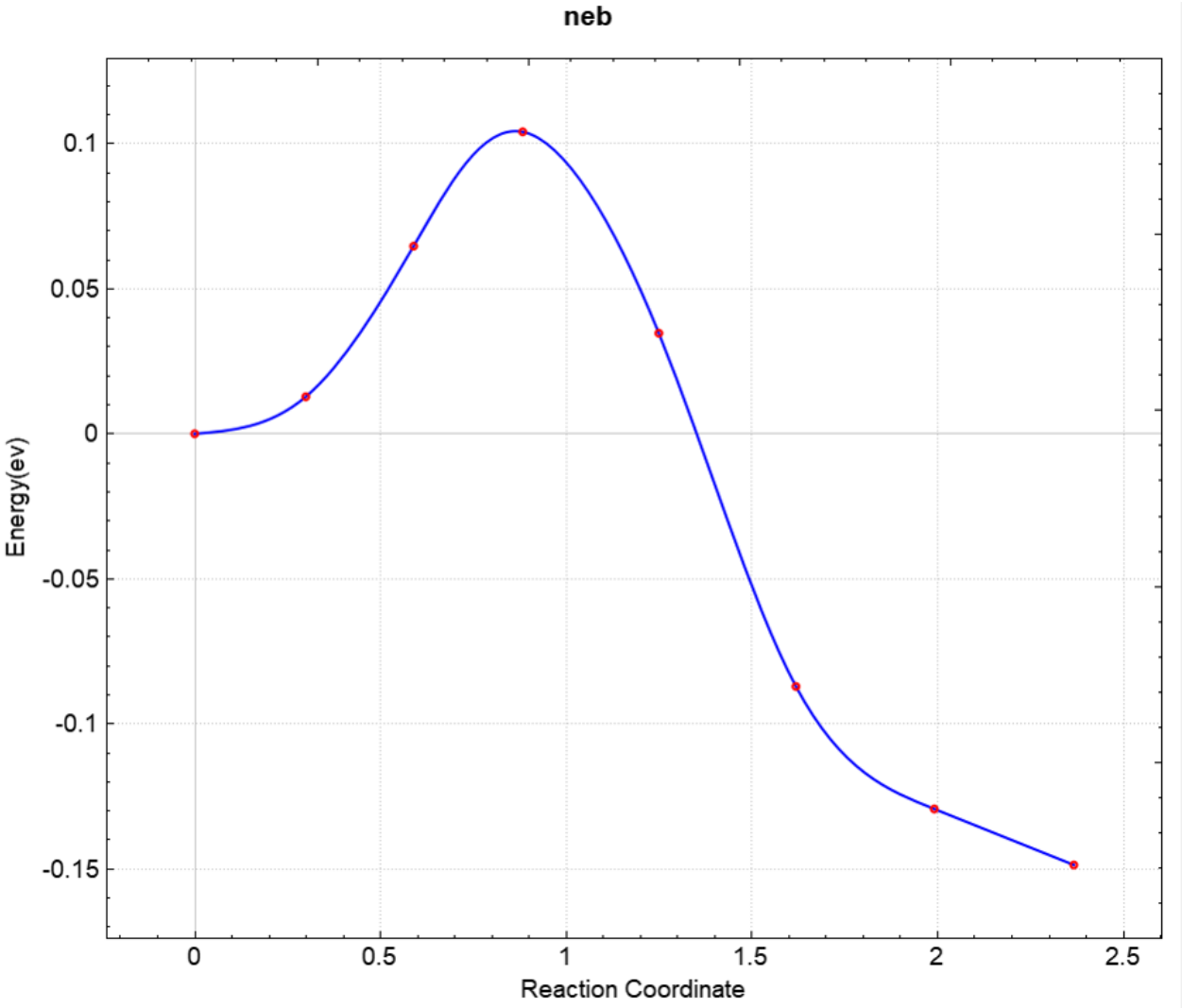
2.27.3 analysis 计算结果分析

根据上述的输入文件，计算完成之后将会得到 `DS-PAW.log` 、`neb.json` 这 2 个文件。

neb.json： 过渡态计算完成之后的 **json** 数据文件；此时反应坐标及能量变化等数据被保存在 `neb.json` 中，具体的数据结构详见数据结构解析部分；

使用 **Device Studio** 可直接对 `neb.json` 文件处理出图，其操作步骤为：Simulator-->DS-PAW-->Analysis Plot，选择 `neb.json` 即可，可根据作图要求自定义设置面板参数。

得到的反应势垒曲线如下所示：



2.28 PBEsol 泛函计算

本节将以 Ta_2O_5 体系为例，介绍在 DS-PAW 中如何使用 PBEsol 泛函进行弛豫计算。

2.28.1 Ta_2O_5 PBEsol 泛函计算输入文件

输入文件包含参数文件 `relax.in` 和结构文件 `structure.as` ，`relax.in` 如下：

```
task = relax

sys.structure = structure.as
sys.functional  = PBESOL
sys.spin = none
sys.symmetry = true

cal.ksamping = MP
cal.kpoints = [4,4,8]
cal.cutoff = 520
cal.smearing = 1
cal.sigma = 0.01

scf.convergence = 1e-5
scf.max = 60

relax.max = 60
relax.freedom = all
relax.convergence = 0.01
relax.methods = CG
```

relax.in 输入参数介绍：

- `sys.functional`：该参数控制使用泛函的类型，此例选择的泛函为 PBESOL；

结构文件 `structure.as` 参考如下

```
Total number of atoms
14
Lattice
 6.25000000 0.00000000 0.00000000
 0.00000000 7.40000000 0.00000000
 0.00000000 0.00000000 3.83000000
Cartesian
O 0.00000000 0.00000000 0.00000000
O 3.12500000 3.70000000 0.00000000
O 2.43375000 1.03970000 0.00000000
O 3.81625000 6.36030000 0.00000000
O 0.69125000 4.73970000 0.00000000
O 5.55875000 2.66030000 0.00000000
O 1.25625000 2.56114000 1.91500000
```

(续下页)

(接上页)

```
O 4.99375000 4.83886000 1.91500000
O 1.86875000 6.26114000 1.91500000
O 4.38125000 1.13886000 1.91500000
Ta 1.40250000 2.82754000 0.00000000
Ta 4.84750000 4.57246000 0.00000000
Ta 1.72250000 6.52754000 0.00000000
Ta 4.52750000 0.87246000 0.00000000
```

2.28.2 run 程序运行

准备好输入文件之后，将 `relax.in` 和 `structure.as` 文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 *DS-PAW relax.in* 。

2.28.3 analysis 计算结果分析

根据上述的输入文件，计算完成之后将会得到 `DS-PAW.log` 、`relax.json` 这 2 个文件及 paw_tmp 文件夹。

relax.json： 弛豫计算结束得到的结构文件

将 *relax.json* 拖入 Device Studio 查看结构，或直接查看 **paw_tmp** 目录下的 relax.tmp 文件，可得弛豫结束后晶胞常数如下表所示，与文献报道结果 [6] 一致。

	a (Å)	b (Å)	c (Å)
PBEsol - this work	6.190	7.315	3.786
PBEsol - ref	6.197	7.333	3.793

2.29 ref 参考文献

[1] Rêgo C R C, Oliveira L N, Tereshchuk P, et al. Comparative study of van der Waals corrections to the bulk properties of graphite[J]. Journal of Physics: Condensed Matter, 2015, 27(41): 415502.

[2] Clima S, Wouters D J, Adelmann C, et al. Identification of the ferroelectric switching process and dopant-dependent switching properties in orthorhombic HfO2: A first principles insight[J]. Applied Physics Letters, 2014, 104(9): 092906.

[3] Chen M, Weinert M. Layer k-projection and unfolding electronic bands at interfaces[J]. Physical Review B, 2018, 98(24): 245421.

[4] Gajdoš M, Hummer K, Kresse G, et al. Linear optical properties in the projector-augmented wave methodology[J]. Physical Review B, 2006, 73(4): 045112.

[5] Bernardini F, Fiorentini V, Vanderbilt D. Spontaneous polarization and piezoelectric constants of III-V nitrides[J]. Physical Review B, 1997, 56(16): R10024.

[6] Valencia-Balvín C, Pérez-Walton S, Osorio-Guillén J M. First principles calculations of the electronic and dielectric properties of λ-Ta2O5[J]. TecnoLógicas, 2018, 21(43): 43-52.

[7] Haynes W M, Lide D R, Bruno T J. CRC handbook of chemistry and physics[M]. CRC press, 2016.

[8] Zhang Z, Guo Y, Robertson J. Role of the third metal oxide in In–Ga–Zn–O4 amorphous oxide semiconductors: Alternatives to gallium[J]. Journal of Applied Physics, 2020, 128(21): 215704.

CHAPTER 3

应用案例

本章将介绍 DS-PAW 的各种应用实例，具体包括：如何计算磁矩、如何计算反铁磁材料等；用户通过以下应用教程，将可以更深入了解 DS-PAW 软件的使用。

3.1 O 原子的磁矩计算

本节中我们将以单个氧原子体系为例来介绍磁性体系的计算。

3.1.1 O 原子自洽计算之文件准备

由于本次计算的是单个氧原子的磁矩，因此并不需要进行结构弛豫计算，我们直接从自洽计算开始，准备参数文件 `scf.in` 和结构文件 `structure.as` ，`scf.in` 如下：

```
task = scf
sys.symmetry = false
sys.structure = structure.as
sys.spin = collinear
cal.smearing = 1
cal.sigma = 0.01
cal.kpoints = [1, 1, 1]
```

本次计算的输入文件中以下几个参数需要重点关注：

- `sys.symmetry`： 本次计算中我们将在关闭对称性的情况下进行计算，因为在 DS-PAW 中可以通过对称性来减少程序的计算量，但是往往通过对称性优化之后，一些较为细节的东西将会不是特别准确，因此本次计算中将关闭对称性；
- `sys.spin`： 指定体系磁性，因为我们知道单个氧原子的磁矩是 **2μB** ，这里我们将设置体系的磁性为 **collinear** 也就是 **共线自旋**；

- `cal.kpoints`：在前面的章节中我们介绍过对于没有周期性的维度，**K 点**可以设置为 1；

structure.as 文件参考如下：

```
Total number of atoms
1
Lattice
7.50000000 0.00000000 0.00000000
0.00000000 8.00000000 0.00000000
0.00000000 0.00000000 8.90000000
Cartesian
O  0.00000000 0.00000000 0.00000000
```

结构文件中我们使用笛卡尔坐标，因此我们设置第七行中坐标类型为 `Cartesian`；为了使结构也尽量的减少对称性，手动晶格改为 **[7.5，8，8.9]** 格子。

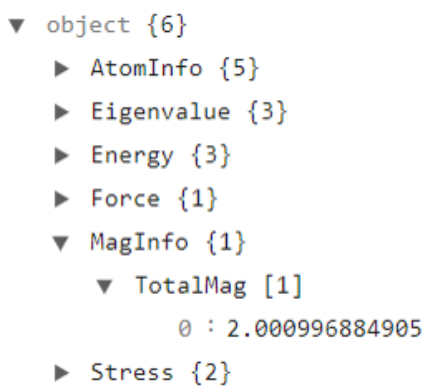
3.1.2 run 程序运行

准备好输入文件之后，将 `scf.in` 和 `structure.as` 文件上传到安装了 DS-PAW 的环境上，运行 *DS-PAW scf.in* 命令。

3.1.3 analysis 计算结果分析

根据上述的输入文件，计算完成之后将会得到 `DS-PAW.log`、`system.json` 这 2 个文件。

使用文本编辑器或者在线显示 **json** 格式的网页中打开 `system.json` 文件，具体数据如下：



在 `system.json` 的 **Eigenvalue - Spin - Occupation** 部分可得向上自旋的电子占据数为 4，向下自旋的电子占据数为 2，同时在 **MagInfo** 部分可得整个体系带有两个 **2μB** 的磁矩。

3.2 NiO 体系的反铁磁计算

本节中我们将以 NiO 体系为例介绍如何设置计算反铁磁计算。

3.2.1 NiO 体系自洽计算

本次案例中同样省去了结构弛豫这个过程，用户重现此案例时，可以自行尝试下进行结构弛豫计算，准备参数文件 `scf.in` 和结构文件 `structure.as` ， `scf.in` 如下：

```
task = scf
sys.structure = structure.as
sys.spin = collinear
cal.smearing = 4
cal.kpoints = [8, 8, 8]
cal.cutoff = 500
```

本次计算的输入文件中以下几个参数需要重点关注：

- `cal.smearing`： 本次计算中采用 **四面体加布洛赫**的方法，当使用该方法时 *sigma* 将强制被设置为 **0** ；
- `sys.spin`： 指定体系磁性，**NiO** 为反铁磁材料因此我们需要设置磁性为 *collinear* 也就是 **共线自旋**；
- `cal.cutoff`： 这里我们设置计算中平面波的截断为 **500** ；

`structure.as` 文件参考如下：

```
Total number of atoms
4
Lattice
4.16840000 2.08420000 2.08420000
2.08420000 4.16840000 2.08420000
2.08420000 2.08420000 4.16840000
Cartesian Mag
Ni 1.04210000 1.04210000 1.04210000   2.0
Ni 5.21050000 5.21050000 5.21050000   -2.0
O 3.12630000 3.12630000 3.12630000    0
O 7.29470000 7.29470000 7.29470000    0
```

可以注意到，在结构文件的第七行的 `Cartesian` 后面加入 **Mag** 标签，设置了这个标签之后就可以设置每个原子的磁矩，由于我们需要体现反铁磁（整个体系不显示磁矩，单个原子有磁矩），我们建立了四个原子的晶胞，一个 Ni 原子设置磁矩为 **-2** ，另一个 Ni 原子设置磁矩为 **2** ，两个 O 原子设置磁矩为 **0** 。

备注:

1. **Mag** 标签可设置体系中各原子的磁矩。线性自旋计算中添加每个原子的总磁矩即可；自旋轨道耦合计算中需添加 x, y, z 方向上的磁矩，添加标签为 **Mag_x** , **Mag_y**, **Mag_z**，在对应的原子坐标后添加三个方向上磁矩即可。以 NiO 体系为例，若进行自旋轨道耦合计算，磁矩设置应如下：

```
Total number of atoms
4
Lattice
4.16840000 2.08420000 2.08420000
2.08420000 4.16840000 2.08420000
2.08420000 2.08420000 4.16840000
Cartesian Mag_x Mag_y Mag_z
Ni 1.04210000 1.04210000 1.04210000   0.0 0.0 2.0
Ni 5.21050000 5.21050000 5.21050000   0.0 0.0 -2.0
O 3.12630000 3.12630000 3.12630000    0.0 0.0 0.0
O 7.29470000 7.29470000 7.29470000    0.0 0.0 0.0
```

3.2.2 run 程序运行

准备好输入文件之后，将 `scf.in` 和 `structure.as` 文件上传到安装了 DS-PAW 的环境上，运行 *DS-PAW scf.in* 命令。

3.2.3 analysis 自洽计算结果分析

根据上述的输入文件，计算完成之后将会得到 `DS-PAW.log`、`system.json` 这 2 个文件。在 `system.json` 中的 **MagInfo** 部分可得自洽完成之后总的磁矩为 **1e-08**，几乎为 0。

3.2.4 NiO 体系态密度计算

之后准备进行态密度计算，准备参数文件 `pdos.in` 结构文件 `structure.as` 和自洽的二进制电荷密度文件 `rho.bin`，此时 **pdos.in** 如下：

```
task = dos
sys.structure = structure.as
sys.spin = collinear
cal.iniCharge = ./rho.bin
cal.smearing = 4
cal.kpoints = [16, 16, 16]
cal.cutoff = 500
dos.range = [-10, 10]
dos.EfShift = true
dos.resolution = 0.05
dos.project = true
```

pdos.in 输入参数介绍：

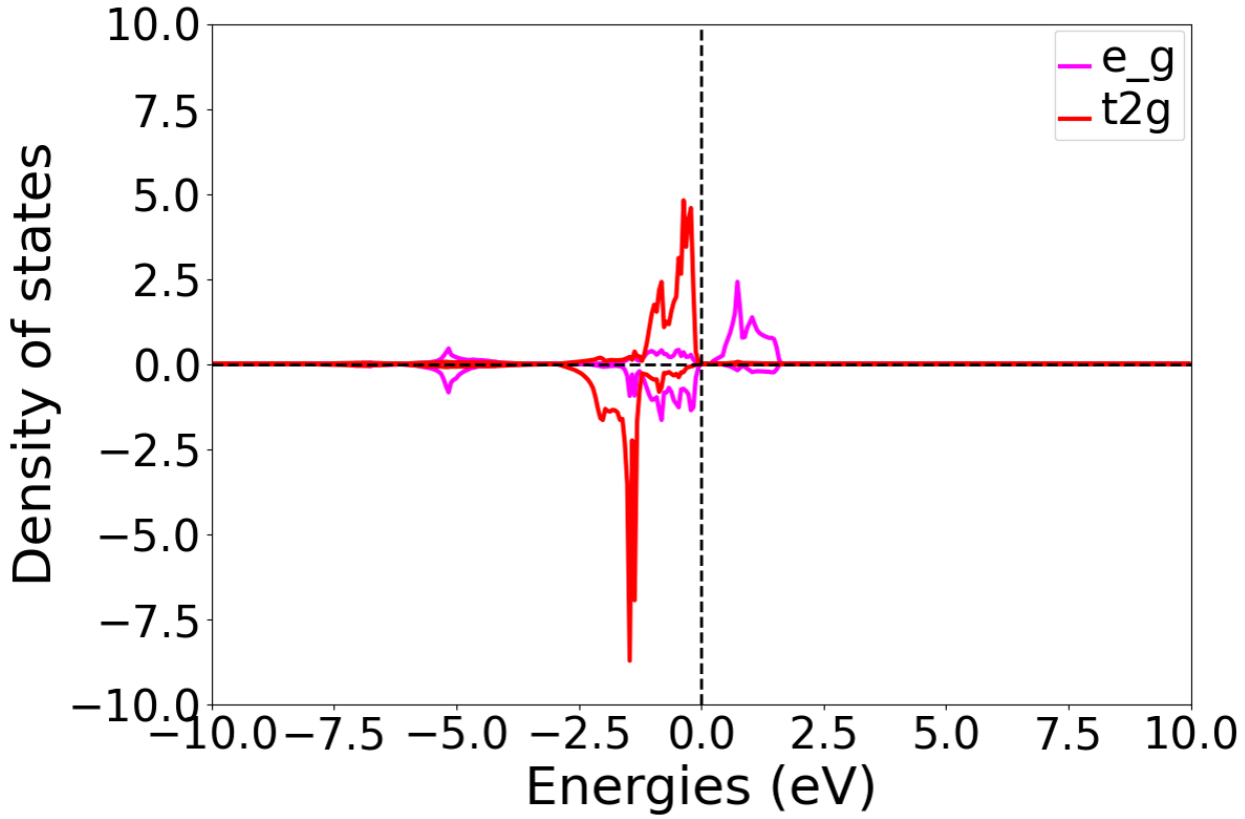
- `dos.range`： 表示能量计算区间为-10 到 10；
- `dos.EfShift`：表示能量计算区间是否按照自洽的费米能级平移，此例中 dos 的能量计算区间会在得到费米能级后进行平移；
- `dos.resolution`： 表示在能量计算区间内打点的间隔精度。此例能量范围为 20 eV，间隔精度为 0.05，对应打点数为 400；
- `dos.project`： 控制态密度的投影计算，本次计算打开了态密度的投影。

3.2.5 run 程序运行

将新建的 `pdos.in` 文件上传 **NiO** 的自洽文件夹上，运行 *DS-PAW pdos.in* 命令。

3.2.6 dos 态密度计算结果分析

根据上述的输入文件，计算完成之后将会得到 `DS-PAW.log` 、`dos.json` 这 2 个文件。使用 python 工具对 `dos.json` 进行数据处理画出 NiO 的 t2g 和 eg 轨道，具体操作见辅助工具使用教程，可得不加 U 得到的态密度图如下所示：



3.2.7 NiO 体系 DFT+U 的态密度计算

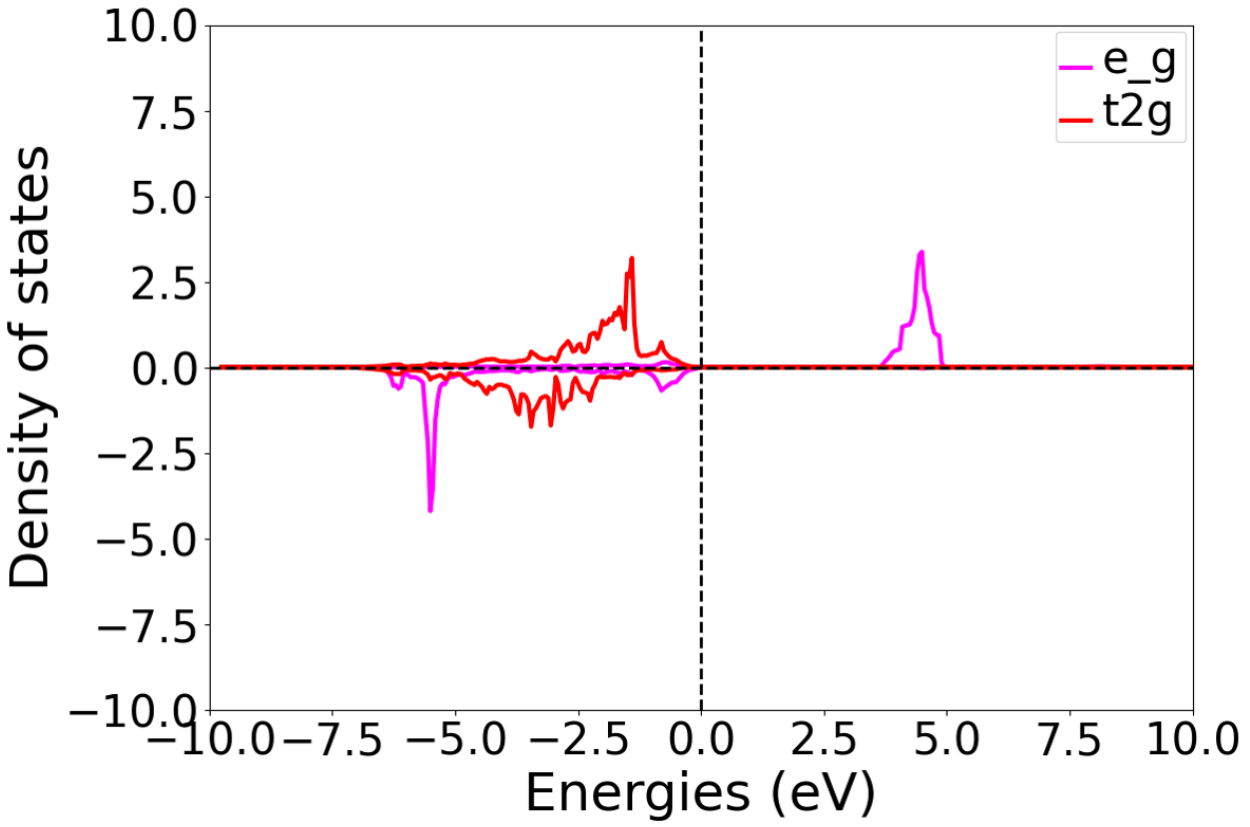
NiO 体系 DFT+U 的态密度计算的计算流程与前面的 NiO 体系的态密度计算的流程一致，我们只需要在前面的 NiO 体系的自洽和态密度的计算中都加入 DFT+U 相关的参数即可：

```
#correction related
corr.dftu=true
corr.dftuElements =[Ni]
corr.dftuOrbital=[d]
corr.dftuU = [8]
corr.dftuJ = [0.95]
```

本次计算的输入文件中以下几个参数需要重点关注：

- `corr.dftu` 设置是否打开 DFT+U 的开关，本例子中设置为 true；
- `corr.dftuElements` 设置需要加 U 的元素，本例中为 Ni；
- `corr.dftuOrbital` 设置需要加 U 的轨道，本例中设置为 d 轨道；
- `corr.dftuU` 设置具体的 U 值，本例中设置为 8；
- `corr.dftuJ` 设置具体的 J 值，本例中设置为 0.95；

自洽和态密度计算完成之后，得到 DFT+U 之后的态密度图如下：



备注：

1. **DFT+U 可以设置多个元素对应轨道的 U 值，例如设置 Ni 的 d 轨道加 U 值为 8，J 值为 0.95；O 的 p 轨道加 U 值为 1，J 值为 0，此时设置如下：corr.dftuElements =[Ni,O] corr.dftuOrbital=[d,p] corr.dftuU = [8,1] corr.dftuJ = [0.95,0]。**

3.3 $AuAl$ slab 模型功函数计算

本节中我们将以 Au 和 Al 的异质结体系为例介绍如何进行功函数计算。

3.3.1 $AuAl$ slab 模型自洽计算之文件准备

本次案例中同样省去了结构弛豫这个过程，用户重现此案例时，可以自行尝试下进行结构弛豫计算，准备参数文件 `scf.in` 和结构文件 `structure.as` ， `scf.in` 如下：

```
task = scf
sys.structure = structure.as
sys.spin = collinear
cal.smearing = 4
cal.kpoints = [8, 8, 1]
cal.cutoff = 400

io.potential=true
potential.type = hartree

#correction related
corr.dipol = true
corr.dipolDirection = c
```

本次计算的输入文件中以下几个参数需要重点关注：

- `io.potential` 为自洽中计算势函数的开关；
- `potential.type` 控制势函数保存的类型，计算功函数的时候需要静电势的数据，这里我们设置 potential.type = hartree；
- `corr.dipol` 为偶极修正的开关；本例中设置为 true；
- `corr.dipolDirection` 本例中设置偶极修正的方向为晶格矢量的 c 方向。

structure.as 文件参考如下：

```
Total number of atoms
8
Lattice
 4.06384898 0.00000000 0.00000000
 0.00000000 4.06384898 0.00000000
 0.00000000 0.00000000 20.00000000
Cartesian
Au 1.01596223 1.01596223 0.00000000
Au 3.04788672 3.04788672 0.00000000
Au 3.04788672 1.01596224 2.03914999
Au 1.01596224 3.04788672 2.03914999
Al 1.01596224 1.01596224 4.07109999
Al 3.04788673 3.04788673 4.07109999
Al 3.04788673 1.01596224 6.09585000
Al 1.01596224 3.04788673 6.09585000
```

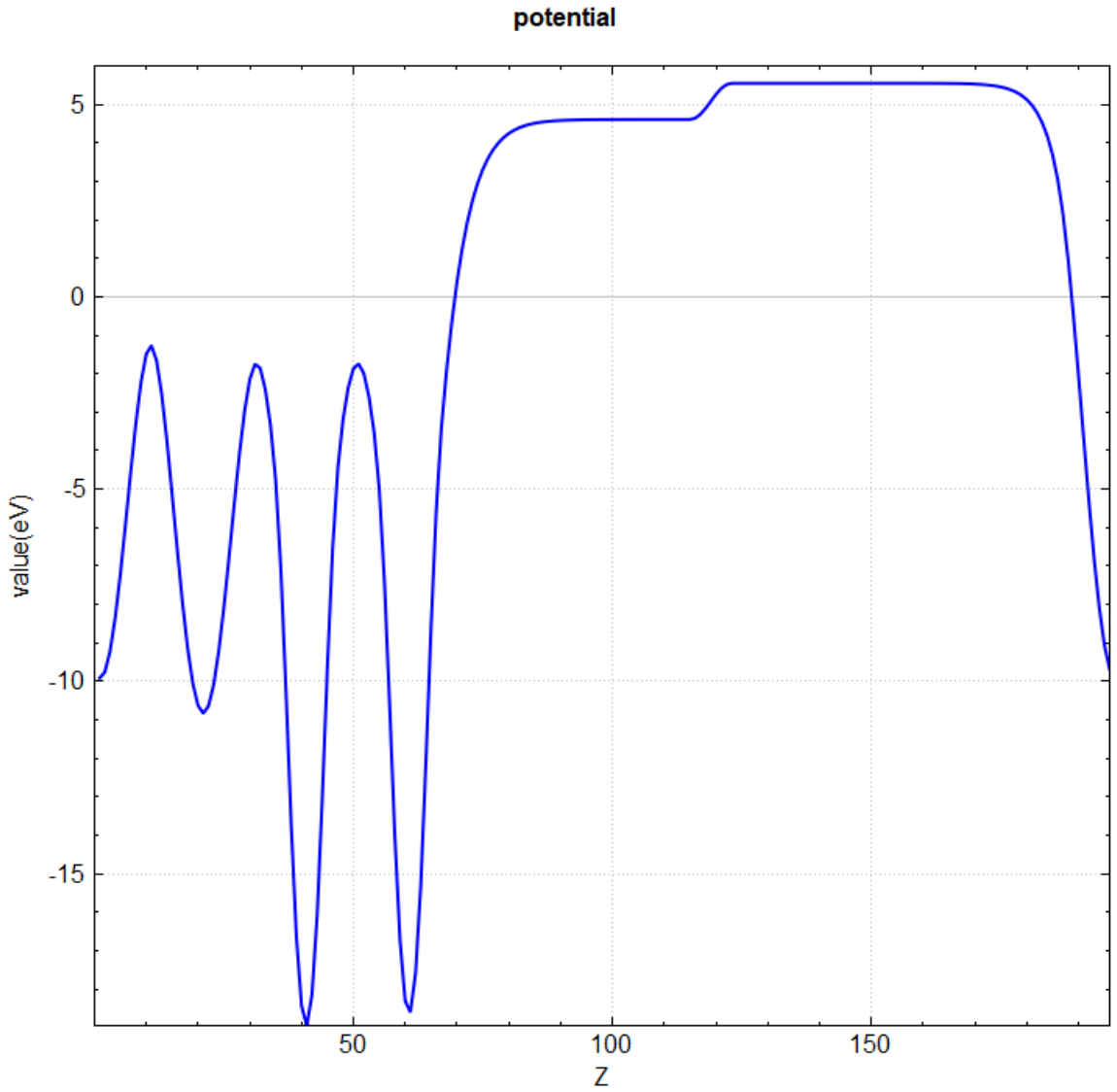
3.3.2 run 程序运行

准备好输入文件之后，将 `scf.in` 和 `structure.as` 文件上传到安装了 DS-PAW 的环境上，运行 DS-PAW scf.in 命令。

3.3.3 workfunction 功函数数据分析

根据上述的输入文件，计算完成之后将会得到 `DS-PAW.log` 、`system.json` 、`potential.json` 等多个文件。我们将对 `potential.json` 文件进行数据处理既可以得到功函数。

使用 **Device Studio** 可直接对 `potential.json` 文件处理出图，其操作步骤为：Simulator-->DS-PAW-->Analysis Plot，选择 `potential.json` 即可，可根据作图要求自定义设置面板参数。DS 处理得到的势函数曲线如下所示：



通过势函数的面内平均图，我们可以得到 Au 和 Al 的真空电势分别为 5.5 ev 和 4.6 eV

从 system.json 中读取费米能级为 0.113 eV

根据公式 $w = -e - E_F$ 得到 Au 和 Al 构成异质结之后，Au 的功函数为 **5.387** eV，Al 的功函数为 **4.487** eV。文献参考值 [7]：Au 的功函数在 **5.10-5.47** eV 区间，Al 的功函数在 **4.06-4.26** eV 区间。

CHAPTER 4

赝势说明

DS-PAW 目前可支持 `.paw`、`.potcar`、`.pawpsp` 3 种格式的赝势使用，用户可通过参数 `sys.pseudoType` 指定使用赝势的类型。

4.1 hzw 内部 paw 赝势

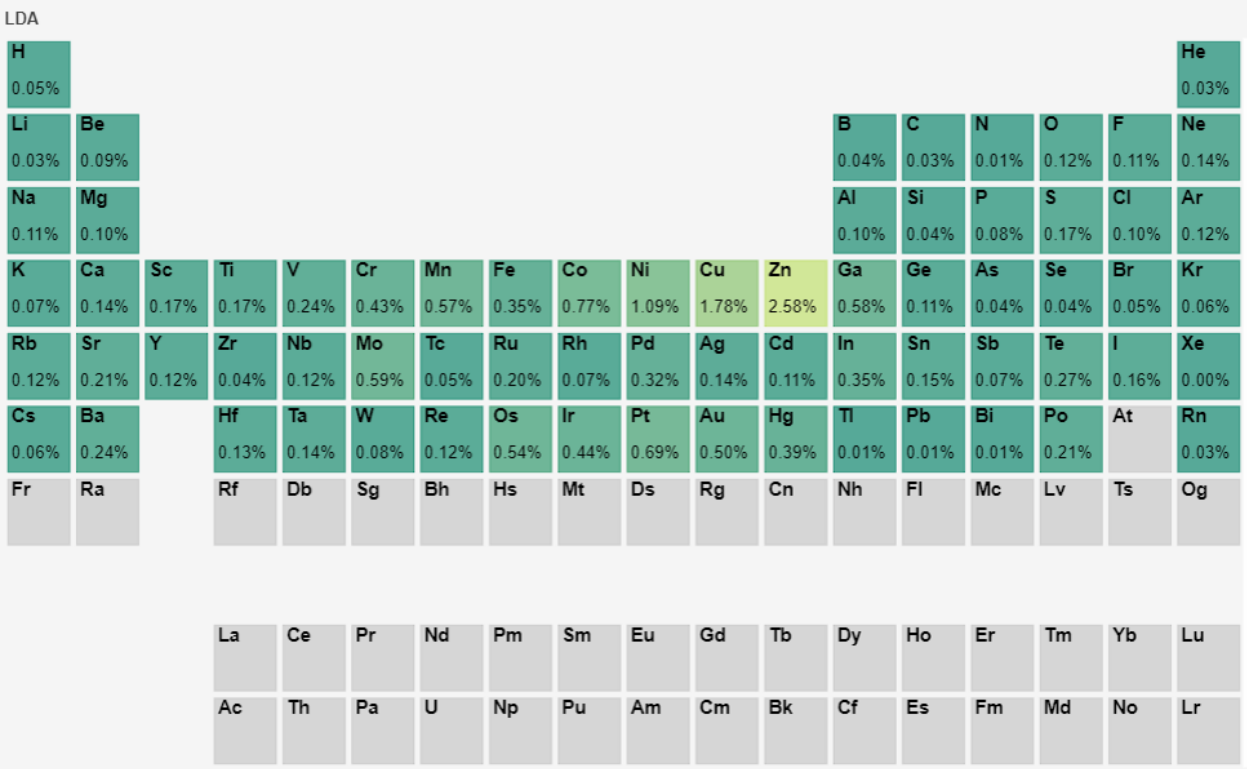
DS-PAW 默认使用的是 **hzw** 赝势（.paw），对应参数 `sys.pseudoType` 为 **-1** ，此时 DS-PAW 会从安装路径 **/pseudopotential** 读取赝势文件，目前 **hzw** 赝势库包含元素共 72 种，为元素周期表中 **1-86** 号元素（镧系元素目前只支持元素镧）。

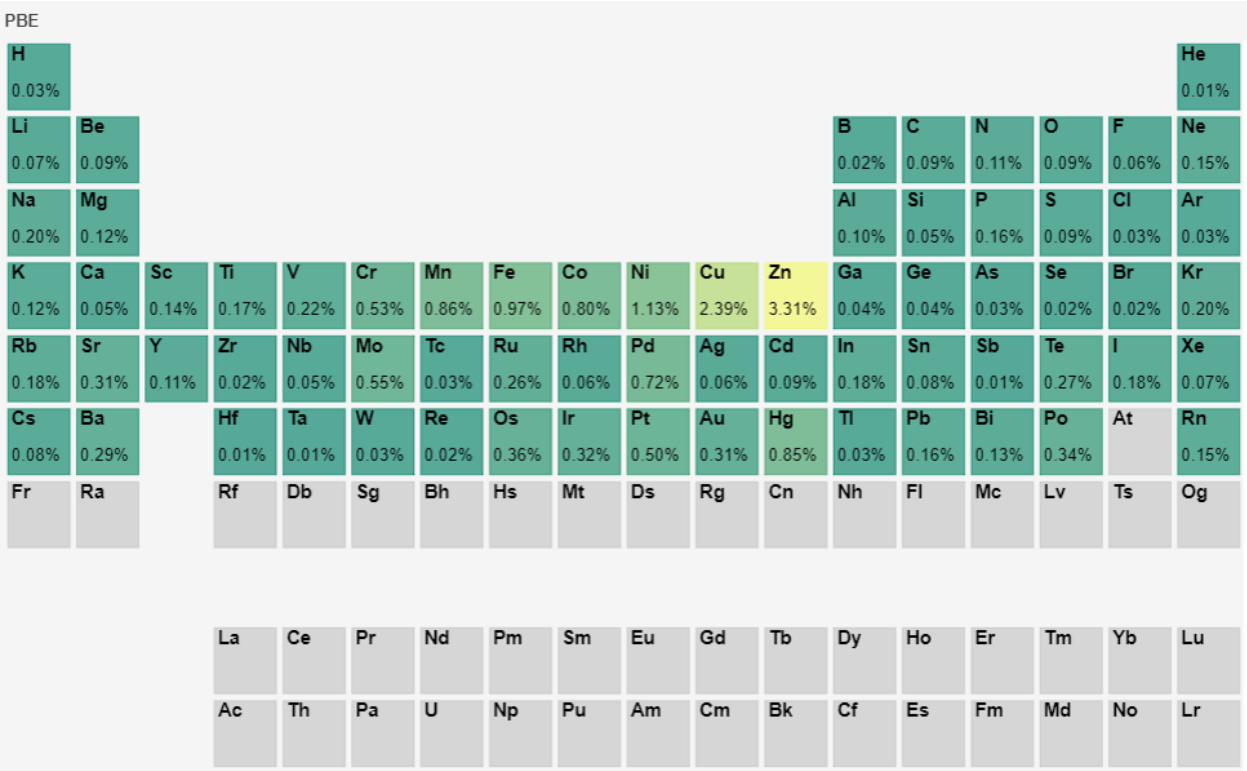
关于 **hzw** 赝势的准确性说明：快速入门与应用案例从多个功能出发进行计算，其计算结果与文献都能很好的吻合，这验证了 **hzw** 赝势在各个功能计算中都能展现较高的准确性。

此外，针对赝势库中的 **72** 种元素所对应的单质，另进行了状态方程拟合获取稳态晶胞体积的计算，测试对象包括 **72** 种元素对应 LDA 和 PBE 泛函赝势，共计 **144** 个赝势文件。测试所得体积与量子化学软件 **WIEN2k** 计算所得结果进行比较，两者计算误差在元素周期表中展示如下：

WIEN2k 数据来源：https://github.com/abinit/pseudo_dojo **因 WIEN2k 网站未提供 La、At 计算数据，上表未展示 La、At 对比数据**

通过对比可得，状态方程拟合得到稳态体积与 **WIEN2k** 软件结果基本一致，其中误差最大的为元素 Zn，对应的 LDA 和 PBE 赝势的误差分别为 **2.58%** 和 **3.31%** ，对这两种元素的赝势优化正在进行中。其余元素的误差基本控制在 **0.1%** 以内，该测试进一步验证了赝势库的整体准确性。


LDA
H 0.05%
He 0.03%
Li 0.03%
Be 0.09%
B 0.04%
C 0.03%
N 0.01%
O 0.12%
F 0.11%
Ne 0.14%
Na 0.11%
Mg 0.10%
Al 0.10%
Si 0.04%
P 0.08%
S 0.17%
Cl 0.10%
Ar 0.12%
K 0.07%
Ca 0.14%
Sc 0.17%
Ti 0.17%
V 0.24%
Cr 0.43%
Mn 0.57%
Fe 0.35%
Co 0.77%
Ni 1.09%
Cu 1.78%
Zn 2.58%
Ga 0.58%
Ge 0.11%
As 0.04%
Se 0.04%
Br 0.05%
Kr 0.06%
Rb 0.12%
Sr 0.21%
Y 0.12%
Zr 0.04%
Nb 0.12%
Mo 0.59%
Tc 0.05%
Ru 0.20%
Rh 0.07%
Pd 0.32%
Ag 0.14%
Cd 0.11%
In 0.35%
Sn 0.15%
Sb 0.07%
Te 0.27%
I 0.16%
Xe 0.00%
Cs 0.06%
Ba 0.24%
Hf 0.13%
Ta 0.14%
W 0.08%
Re 0.12%
Os 0.54%
Ir 0.44%
Pt 0.69%
Au 0.50%
Hg 0.39%
Tl 0.01%
Pb 0.01%
Bi 0.01%
Po 0.21%
At
Rn 0.03%
Fr
Ra
Rf
Db
Sg
Bh
Hs
Mt
Ds
Rg
Cn
Nh
Fl
Mc
Lv
Ts
Og
La
Ce
Pr
Nd
Pm
Sm
Eu
Gd
Tb
Dy
Ho
Er
Tm
Yb
Lu
Ac
Th
Pa
U
Np
Pu
Am
Cm
Bk
Cf
Es
Fm
Md
No
Lr


PBE
H 0.03%
He 0.01%
Li 0.07%
Be 0.09%
B 0.02%
C 0.09%
N 0.11%
O 0.09%
F 0.06%
Ne 0.15%
Na 0.20%
Mg 0.12%
Al 0.10%
Si 0.05%
P 0.16%
S 0.09%
Cl 0.03%
Ar 0.03%
K 0.12%
Ca 0.05%
Sc 0.14%
Ti 0.17%
V 0.22%
Cr 0.53%
Mn 0.86%
Fe 0.97%
Co 0.80%
Ni 1.13%
Cu 2.39%
Zn 3.31%
Ga 0.04%
Ge 0.04%
As 0.03%
Se 0.02%
Br 0.02%
Kr 0.20%
Rb 0.18%
Sr 0.31%
Y 0.11%
Zr 0.02%
Nb 0.05%
Mo 0.55%
Tc 0.03%
Ru 0.26%
Rh 0.06%
Pd 0.72%
Ag 0.06%
Cd 0.09%
In 0.18%
Sn 0.08%
Sb 0.01%
Te 0.27%
I 0.18%
Xe 0.07%
Cs 0.08%
Ba 0.29%
Hf 0.01%
Ta 0.01%
W 0.03%
Re 0.02%
Os 0.36%
Ir 0.32%
Pt 0.50%
Au 0.31%
Hg 0.85%
Tl 0.03%
Pb 0.16%
Bi 0.13%
Po 0.34%
At
Rn 0.15%
Fr
Ra
Rf
Db
Sg
Bh
Hs
Mt
Ds
Rg
Cn
Nh
Fl
Mc
Lv
Ts
Og
La
Ce
Pr
Nd
Pm
Sm
Eu
Gd
Tb
Dy
Ho
Er
Tm
Yb
Lu
Ac
Th
Pa
U
Np
Pu
Am
Cm
Bk
Cf
Es
Fm
Md
No
Lr

4.2 VASP 赝势

DS-PAW 提供了外部 potcar 格式赝势（.potcar）的使用接口，对应参数 `sys.pseudoType` 为 **10** ，此时 DS-PAW 会从默认路径 **./** 读取赝势文件。**由于版权限制，DS-PAW 只提供 VASP 赝势的使用接口，赝势文件需用户自行准备**。在使用时需要用户把赝势文件名做相应的修改，如果使用硅元素的 LDA 赝势，对应的 POTCAR 需要改成 Si_LDA.potcar，放置到指定目录（可以通过 sys.pseudoPath 设置）。

4.3 gbrv 赝势

DS-PAW 提供了外部 gbrv 格式赝势（.pawpsp）的使用接口，对应参数 `sys.pseudoType` 为 **11** ，此时 DS-PAW 会从默认路径 **./** 读取赝势文件。gbrv 赝势是一套可以免费使用的赝势库，总共提供 64 种元素的赝势文件，获取网站 http://www.physics.rutgers.edu/gbrv/，下载时需要注意，DS-PAW 支持的格式是 PAW format for Abinit。在使用时需要用户把赝势文件名做相应的修改，如果使用硅元素的 LDA 赝势，对应的赝势文件需要改成 Si_LDA.pawpsp，放置到指定目录（可以通过 sys.pseudoPath 设置）。

4.4 compare 赝势对比

4.4.1 Si 体系能带计算

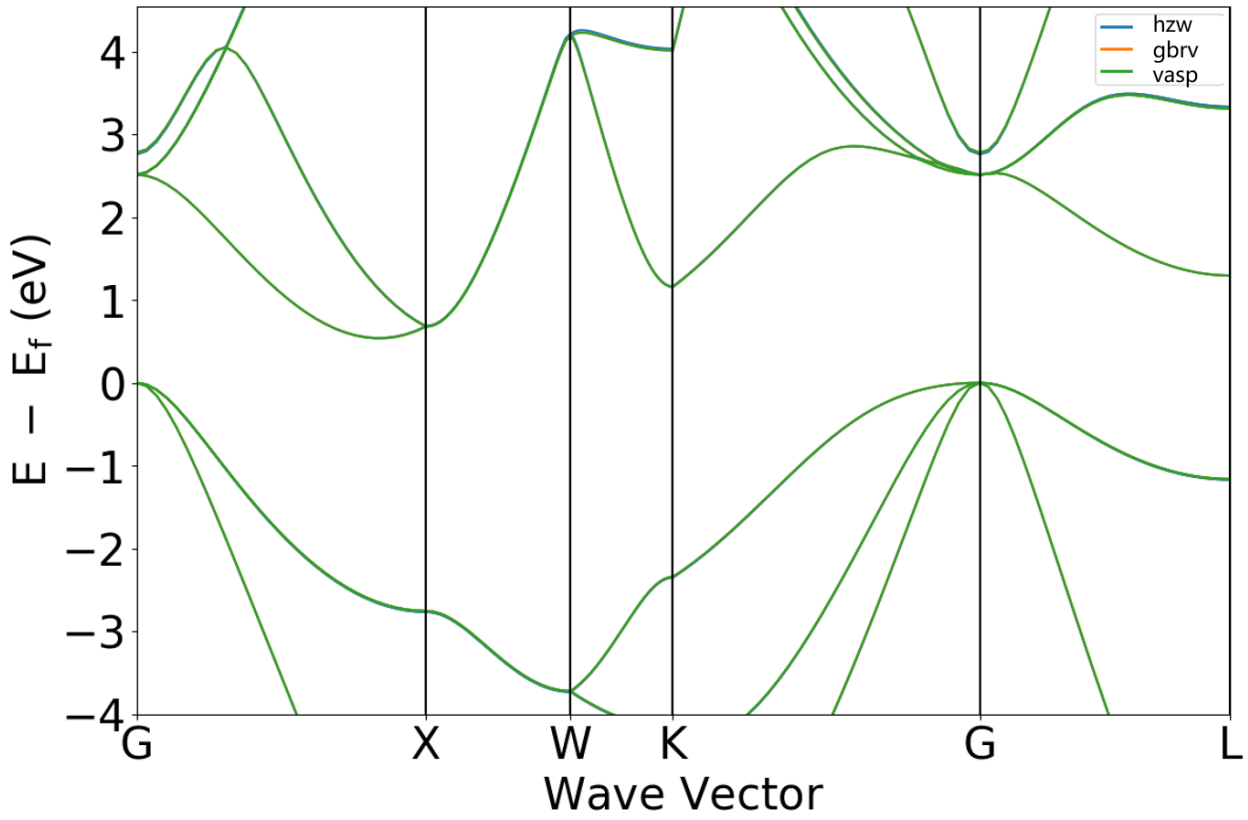
为对比三种赝势的计算结果，本节以 Si 体系为例，分别使用以上三种赝势进行了能带计算，下图为能带对比图，该对比图说明三种赝势在描述 Si 的能带时，体系出较高的一致性，验证了 hzw 赝势的准确性。

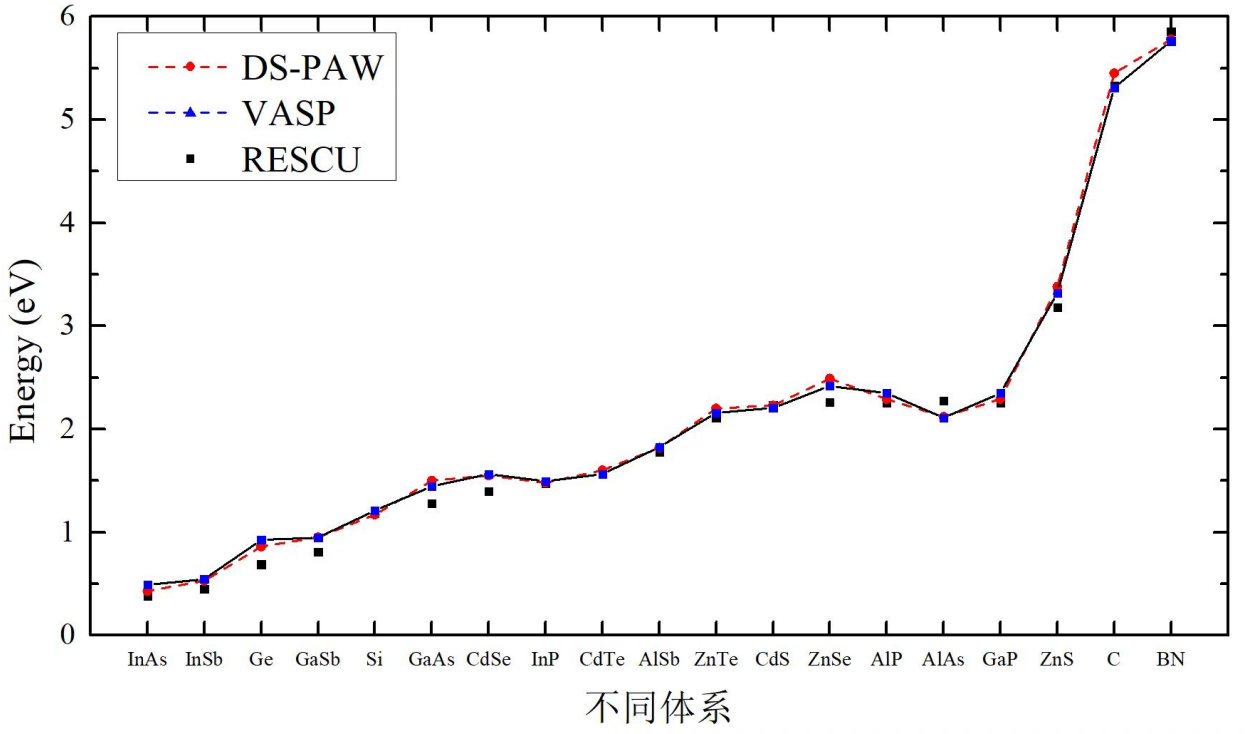
数据来源：该图计算结果由鸿之微合作者提供

4.4.2 multi 多体系带隙计算

本节以多个体系为例，使用 **DS-PAW** 对这些体系进行 HSE06 能带计算，并得到了多个体系的带隙值，将 **DS-PAW** 计算得到的带隙值与文献提供的 **VASP** 及 **RESCU** 软件的计算数据进行对比，得到下图，该图进一步验证了 hzw 赝势的准确性。

数据来源：https://doi.org/10.1103/PhysRevB.97.075139


hzw
gbrv
vasp
E − Ef (eV)
4
3
2
1
0
−1
−2
−3
−4
G
X
W
K
G
L
Wave Vector


DS-PAW
VASP
RESCU
Energy (eV)
6
5
4
3
2
1
0
InAs
InSb
Ge
GaSb
Si
GaAs
CdSe
InP
CdTe
AlSb
ZnTe
CdS
ZnSe
AlP
AlAs
GaP
ZnS
C
BN
不同体系

CHAPTER 5

参数说明

5.1 parameter 参数列表

- *task*

- *sys.pseudoType*
- *sys.pseudoPath*
- *sys.structure*
- *sys.symmetry*
- *sys.symmetryAccuracy*
- *sys.functional*
- *sys.spin*
- *sys.soi*
- *sys.electron*
- *sys.hybrid*
- *sys.hybridType*
- *sys.hybridAlpha*
- *sys.hybridOmega*

- *cal.iniCharge*
- *cal.iniWave*

- *cal.cutoffFactor*
- *cal.cutoff*
- *cal.methods*
- *cal.smearing*
- *cal.sigma*
- *cal.kpoints*
- *cal.ksamping*
- *cal.toalBands*

- *io.charge*
- *io.elf*
- *io.potential*
- *io.wave*
- *io.band*
- *io.dos*
- *io.optical*
- *io.bader*
- *io.polarization*
- *io.magProject*
- *io.outStep*

- *scf.max*
- *scf.min*
- *scf.mixBeta*
- *scf.mixType*
- *scf.convergence*

- *relax.max*
- *relax.freedom*
- *relax.methods*
- *relax.convergenceType*
- *relax.convergence*
- *relax.stepRange*

- *dos.range*
- *dos.resolution*

- *dos.project*
- *dos.EfShift*

- *band.kpointsLabel*
- *band.kpointsCoord*
- *band.kpointsNumber*
- *band.project*
- *band.unfolding*
- *band.primitiveUVW*

- *potential.type*

- *corr.chargedSystem*
- *corr.dipol*
- *corr.dipolDirection*
- *corr.dftu*
- *corr.dftuElements*
- *corr.dftuOrbital*
- *corr.dftuU*
- *corr.dftuJ*
- *corr.VDW*
- *corr.VDWType*
- *corr.dipolEfield*
- *corr.dipolPosition*

- *pcharge.bandIndex*
- *pcharge.kpointsIndex*
- *pcharge.sumK*

- *neb.springK*
- *neb.images*
- *neb.iniFin*
- *neb.method*
- *neb.convergenceType*
- *neb.convergence*
- *neb.stepRange*

- *neb.max*
- *neb.freedom*

- *frequency.dispOrder*
- *frequency.dispRange*

- *phonon.structureSize*
- *phonon.method*
- *phonon.type*
- *phonon.isDisplacement*
- *phonon.fdDisplacement*
- *phonon.iniPhonon*
- *phonon.qsamping*
- *phonon.qpoints*
- *phonon.qpointsLabel*
- *phonon.qpointsCoord*
- *phonon.qpointsNumber*
- *phonon.primitiveUVW*
- *phonon.dosRange*
- *phonon.dosResolution*
- *phonon.dosSigma*
- *phonon.dfptEpsilon*
- *phonon.nac*
- *phonon.thermal*
- *phonon.thermalRange*

- *elastic.dispOrder*
- *elastic.dispRange*

- *aimd.ensemble*
- *aimd.thermostat*
- *aimd.iniTemp*
- *aimd.finTemp*
- *aimd.timeStep*
- *aimd.totalSteps*

5.2 detail 参数详细描述

参数名称: *task*

默认值: 无

可选值: `scf/relax/dos/band/potential/elf/pcharge/neb/frequency/phonon/elastic/aimd/epsilon/`

描述: *task* 参数表示计算的类型，必须设置；`scf/relax` 可以是从头计算（不需要设置 `cal.iniCharge` 和 `cal.iniWave` ）也可以导入电荷密度或波函数（设置 `cal.iniCharge` 和 `cal.iniWave` ）；`dos/band/potential/elf` 为后处理计算，必须要读取电荷密度，在导入电荷密度的同时可选择性的导入波函数（必须设置 **cal.iniCharge** ，选择性设置 **cal.iniWave** ）；当 **task=scf/realx** 时，默认输出 `system.json` ；当 **task=dos/band/potential/elf/pcharge** 时，默认输出对应的 json 文件 `dos.json/band.json/potential.json/elf.json/pcharge.json` ; **task=neb/frequency/phonon/elastic/aimd/epsilon** 分别对应过渡态/频率/声子/弹性/分子动力学/介电常数计算，默认输出对应的 json 文件 `neb.json/frequency.json/phonon.json/elastic.json/aimd.json/epsilon.json`;

案例: task = scf

参数名称: *sys.pseudoType*

默认值: -1

可选值: -1/10/11

描述: `sys.pseudoType` 参数为设置 **DS-PAW** 计算所需的赝势格式；-1 表示使用 hzw 赝势（*.paw*），目前 *DS-PAW* 支持 ***H He Li Be B C N O F Ne Na Mg Al Si P S Cl Ar K Ca Sc Ti V Cr Mn Fe Co Ni Cu Zn Ga Ge As Se Br Kr Rb Sr Y Zr Nb Mo Tc Ru Rh Pd Ag Cd In Sn Sb Te I Xe Cs Ba La Hf Ta W Re Os Ir Pt Au Hg Tl Pb Bi Po At Rn** 72 种元素的 hzw 赝势；10 表示外部 potcar 格式赝势 (*.potcar), 11* 表示外部 *pawpsp* 格式赝势 (.pawpsp)；

案例: sys.pseudoType = -1

参数名称: *sys.pseudoPath*

默认值: 当 `sys.pseudoType` =-1 时，无需设置该参数，程序只能从安装路径 **/pseudopotential** 读取赝势文件；`sys.pseudoType` = 10，默认值 **./** ；`sys.pseudoType` = 11，默认值 **./** ；

描述: `sys.pseudoPath` 参数为设置 **DS-PAW** 计算所需的赝势所在路径；一般无需自行设置，读取 hzw 赝势时会从默认存储路径读取，读取外部赝势时会默认从当前路径读取；

案例: sys.pseudoPath = ./

参数名称: *sys.structure*

默认值: atoms.as

可选值: XXX.as /XXX.json

描述: `sys.structure` 参数设置结构文件的路径，结构文件可以是.as 格式或.json 格式，支持绝对路径和相对路径；使用 DS-PAW 进行结构弛豫之后会生成 `relax.json` 文件，可以直接设置 sys.structure 为 **relax.json** 即可读取结构弛豫的文件进行计算；

案例: sys.structure = structure.as

参数名称: *sys.symmetry*

默认值: true

可选值: true/false

描述: `sys.symmetry` 该参数表示 DS-PAW 计算时是否进行对称性分析;

案例: sys.symmetry = false

参数名称: *sys.symmetryAccuracy*

默认值: 1.0e-5

可选值: real

描述: `sys.symmetryAccuracy` 该参数表示 DS-PAW 计算时对称性分析的精度;

案例: sys.symmetryAccuracy = 1.0e-6

参数名称: *sys.functional*

默认值: LDA

可选值: LDA/PBE/REVPBE/RPBE/PBESOL/vdw-optPBE/vdw-optB88/vdw-optB86b/vdw-DF/vdw-revPBE/vdw-DF2/vdw-revDF2

描述: `sys.functional` 参数指定 DS-PAW 的泛函类型，如果 **sys.functional=LDA** 则会去读取指定路径下的 LDA 赝势；vdw 开头的系列赝势对应泛函类的范德瓦尔斯校正方法;

案例: sys.functional = PBESOL

参数名称: *sys.spin*

默认值: none

可选值: none/collinear/non-collinear

描述: `sys.spin` 参数指定计算的自旋性质；**none** 表示没有自旋，**collinear** 表示共线自旋，**non-collinear** 表示一般自旋;

案例: sys.spin = collinear

参数名称: *sys.soi*

默认值: false

可选值: true/false

描述: `sys.soi` 表示是否考虑自旋轨道耦合效应；自旋轨道耦合效应需要在 sys.spin=non-collinear 时才会生效;

案例: sys.soi = true

参数名称: *sys.electron*

默认值: 所有价电子的总和

可选值: real

描述： sys.electron 参数指定价电子的总数；DS-PAW 通过引入背景电荷的方法计算带电体系；

案例： sys.electron = 12

参数名称: *sys.hybrid*

默认值： false

可选值： true/false

描述： sys.hybrid 参数指定是否使用杂化泛函；true 表示引入杂化泛函，false 表示不引入杂化泛函；

案例： sys.hybrid = true

参数名称: *sys.hybridType*

默认值： HSE06

可选值： PBE0/HSE03/HSE06/B3LYP

描述： sys.hybridType 参数指定杂化泛函的类型；该参数只有在 sys.hybrid = true 时生效；

案例： sys.hybridType = HSE06

参数名称: *sys.hybridAlpha*

默认值： 当 sys.hybridType = PBE0 时，默认值为 **0.25** ，当 sys.hybridType = HSE06 时，默认值为 **0.25** ，当 sys.hybridType = HSE03 时，默认值为 **0.25**

可选值： real

描述： sys.hybridAlpha 参数指定杂化泛函中精确的交换相关泛函的系数；该参数只有在 sys.hybrid = true 时生效；

案例： sys.hybridAlpha = 0.20

参数名称: *sys.hybridOmega*

默认值： 当 sys.hybridType = PBE0 时，默认值为 **0** ，当 sys.hybridType = HSE06 时，默认值为 **0.2** ，当 sys.hybridType = HSE03 时，默认值为 **0.3**

可选值： real

描述： sys.hybridOmega 参数指定杂化泛函的屏蔽系数；该参数只有在 sys.hybrid = true 时生效；

案例： sys.hybridOmega = 0.2

参数名称: *cal.iniCharge*

默认值: 无

可选值: 指定 rho.bin 文件路径

描述： cal.iniCharge 参数表示用户可以通过指定 DS-PAW 自洽或结构弛豫计算得到的电荷密度文件 **rho.bin** 的路径进行后续的计算；**task=scf/relax** 时，如果不需要读取上一次的电荷密度时则不设置 cal.iniCharge ，如果需要如果读取上一次的电荷密度时则设置 cal.iniCharge；当 task= dos/band/potential/elf 时必须设置 cal.iniCharge 指定 **rho.bin** 的路径；文件路径支持相对路径和绝对路径；

案例: cal.iniCharge = /home/exp/rho.bin

参数名称: *cal.iniWave*

默认值: 无

可选值: 指定 wave.bin 文件路径

描述: `cal.iniWave` 参数表示用户可以通过指定 DS-PAW 自洽或结构弛豫计算得到的波函数文件 **wave.bin** 的路径进行后续的计算；不设置此参数则表示不读取 **wave.bin** ； 文件路径支持相对路径和绝对路径；

案例: cal.iniWave = /home/exp/wave.bin

参数名称: *cal.cutoffFactor*

默认值: 1.0

可选值: real

描述: `cal.cutoffFactor` 表示截断能参数 cal.cutoff 的系数，当 **cal.cutoffFactor=1.5** 时，DS-PAW 计算时使用的截断能为 cal.cutoff*1.5。DS-PAW2022A 版本内部赝势皆已完成测试，cutoffFactor 设置默认值 1.0 已能满足大多计算要求；

案例: cal.cutoffFactor = 1.0

参数名称: *cal.cutoff*

默认值: 当前计算所用的元素赝势中截断能的最大值；

可选值: real

描述: `cal.cutoff` 参数表示 DS-PAW 软件计算时候使用的平面波基矢的截断能，可在安装路径 **/pseudopotential** 查看各赝势文件内置截断能 ecutoff 的大小，如从 O_PBE.paw 文件可读出 O_PBE 的 ecutoff 为 650 eV。

案例: cal.cutoff = 650

参数名称: *cal.methods*

默认值: 1

可选值: 1/2/3/4/5

描述: `cal.methods` 表示自洽电子部分优化的方法，1 表示 `BD(block Davidson)` 方法；2 表示 `RM(residual minimization)` 方法；3 表示 `RM(residual minimization)` 方法和 `BD(block Davidson)` 方法的组合；4 表示 `damped MD` (阻尼分子动力学) 方法；5 表示 `conjugated gradient` (共轭梯度方法)；其中 4 和 5 可以在杂化泛函中使用；

案例: cal.methods = 1

参数名称: *cal.smearing*

默认值: 1

可选值: 1/2/3/4

描述: `cal.smearing` 表示用何种方法来设置每个波函数的部分占有数 Gaussian smearing/Fermi-smearing/Methfessel-Paxton order 1/tetrahedron method with Blochl corrections；

案例: cal.smearing = 2

参数名称: *cal.sigma*

默认值: 0.2

可选值: real

描述: `cal.sigma` 表示使用有限温度方法设置部分占有数时的展宽;

案例: cal.sigma = 0.01

参数名称: *cal.kpoints*

默认值: [1,1,1]

可选值: 3*1 int array

描述: `cal.kpoints` 表示 DS-PAW 设置布里渊区 k 点网格取样大小;

案例: cal.kpoints = [9,9,9]

参数名称: *cal.ksamping*

默认值: MP

可选值: MP/G

描述: `cal.ksamping` 表示 DS-PAW 自动生成布里渊区 k 点网格的方法，`Monkhorst-Pack` 方法/ `Gamma centered` 方法;

案例: cal.ksamping = G

参数名称: *cal.totalBands*

默认值: 与体系价电子数目相关

可选值: int

描述: `cal.totalBands` 表示 DS-PAW 计算中所包含的总能带数目;

案例: cal.totalBands = 100

参数名称: *io.charge*

默认值: true

可选值: true/false

描述: 输出电荷密度的二进制文件 `rho.bin` 和 `rho.json` 文件; 当 io.charge=true 时，生成 `rho.bin` 和 `rho.json` 文件; 当 task=scf/realx 时该参数可以选择 **true/false** ; 当 task= dos/band/potential/elf 时，io.charge 强制为 **false** ;

案例: io.charge = true

参数名称: *io.elf*

默认值: false

可选值: false/true

描述: 输出 ELF 的数据结果；当 task=scf/realx 时该参数可以选择 **true/false** ； 当 task= dos/band/potential 时，io.elf 强制为 **false** ； 当 task=elf 时，io.elf 强制为 **true** ；

案例: io.elf = true

参数名称: *io.potential*

默认值: false

可选值: false/true

描述: 输出势函数的数据结果；当 task=scf/realx 时该参数可以选择 **true/false** ； 当 task= dos/band/elf 时，io.potential 强制为 **false** ； 当 task=potential 时，io.potential 强制为 **true** ； 当 io.potential=true 时，可以选择 `potential.type` 来设置输出势函数的类型；

案例: io.potential = true

参数名称: *io.wave*

默认值: false

可选值: false/true

描述: 输出波函数的二进制文件 `wave.bin` ；当 io.wave=true 时，生成 `wave.bin` 文件；当 task=scf/realx 时该参数可以选择 **true/false** ； 当 task= dos/band/potential/elf 时，io.wave 强制为 **false** ；

案例: io.wave = true

参数名称: *io.band*

默认值: false

可选值: false/true

描述: 在 task=scf 时是否直接计算能带的开关；当 io.band=true 时，所有能带计算参数都生效；

案例: io.band = true

参数名称: *io.dos*

默认值: false

可选值: false/true

描述: 在 task=scf 时是否直接计算态密度的开关；当 io.dos=true 时，所有态密度计算参数都生效；

案例: io.dos = true

参数名称: *io.optical*

默认值: false

可选值: false/true

描述： 输出光学性质计算文件 optical.json；当 io.optical=true 时，生成 optical.json 文件；当 task=scf 时该参数可以选择 true/false；

案例: io.optical = true

参数名称: *io.bader*

默认值: false

可选值: false/true

描述： 输出 bader 电荷文件 bader.json；当 io.bader=true 时，生成 bader.json 文件；当 task=scf 时该参数可以选择 true/false；

案例: io.bader = true

参数名称: *io.polarization*

默认值: false

可选值: false/true

描述： 输出铁电极化文件 polarization.json；当 io.polarization=true 时，生成 polarization.json 文件；当 task=scf 时该参数可以选择 true/false；

案例: io.polarization = true

参数名称: *io.magProject*

默认值: false

可选值: false/true

描述： 在磁矩计算中，控制在 system.json 文件中是否写入磁矩信息；

案例: io.magProject = true

参数名称: *io.outStep*

默认值: task = relax/neb 时默认值为 1，task = aimd 时默认值为 20

可选值: int

描述: 控制在结构弛豫计算、过渡态计算、分子动力学模拟计算中，多少个离子步在对应的 paw_tmp/relax.tmp、paw_tmp/neb.tmp、paw_tmp/aimd.tmp 文件中写一次结构信息；

案例: io.outStep = 50

参数名称: *scf.max*

默认值: 60

可选值: int

描述： `scf.max` 表示 DS-PAW 自洽计算时电子步的最大步数；

案例: scf.max = 100

参数名称: *scf.min*

默认值: 2

可选值: int

描述: `scf.min` 表示 DS-PAW 自洽计算时电子步的最少步数;

案例: scf.min = 5

参数名称: *scf.mixBeta*

默认值: 0.4

可选值: real

描述: `scf.mixBeta` 表示 DS-PAW 自洽计算时电子混合算法的 Beta 值;

案例: scf.mixBeta = 0.2

参数名称: *scf.mixType*

默认值: Broyden

可选值: Broyden/Kerker

描述: `scf.mixType` 表示 DS-PAW 自洽计算时电子混合算法的类型，目前支持 **Broyden 方法**和 **Kerker 方法**;

案例: scf.mixType = Broyden

参数名称: *scf.convergence*

默认值: 1.0e-4

可选值: real

描述: `scf.convergence` 表示 DS-PAW 自洽计算时，能量的收敛判据;

案例: scf.convergence = 1.0e-5

参数名称: *relax.max*

默认值: 60

可选值: int

描述: `relax.max` 表示 DS-PAW 结构弛豫时，最大的离子步步数;

案例: relax.max = 300

参数名称: *relax.freedom*

默认值: atom

可选值: atom/all/volume

描述: `relax.freedom` 表示 DS-PAW 结构弛豫的自由度，atom 表示只弛豫原子; all 表示弛豫晶格常数和原子; volume 表示只弛豫晶格;

案例: relax.freedom = atom

参数名称: *relax.methods*

默认值: CG

可选值: CG/DMD/QN

描述: `relax.methods` 表示 DS-PAW 结构驰豫的方法，CG 表示共轭梯度法; DMD 表示阻尼分子动力学法; QN 表示准牛顿方法；

案例: relax.methods = CG

参数名称: *relax.convergenceType*

默认值: force

可选值: force/energy

描述: `relax.convergenceType` 表示驰豫计算中收敛标准的选择，可以选择受力或以能量为收敛标准；

案例: relax.convergenceType = energy

参数名称: *relax.convergence*

默认值: 0.05/1e-4

可选值: real

描述: `relax.convergence` 表示 DS-PAW 结构驰豫时，原子受力或能量的收敛判据；选择力为收敛标准时默认值为 0.05，选择能量为收敛标准时默认值为 1e-4；

案例: relax.convergence = 0.01

参数名称: *relax.stepRange*

默认值: 0.5

可选值: real

描述: `relax.stepRange` 表示结构驰豫时，结构驰豫的中的缩放常数；

案例: relax.stepRange = 0.2

参数名称: *dos.range*

默认值: [-10,10]

可选值: 2*1 array

描述: `dos.range` 表示当 task=dos 时，态密度计算能量的区间；

案例: dos.range = [-15,15]

参数名称: *dos.resolution*

默认值: 0.05

可选值: real

描述: `dos.resolution` 表示当 task=dos 时，态密度计算能量间隔精度;

案例: dos.resolution = 0.1

参数名称: *dos.project*

默认值: false

可选值: false/true

描述: `dos.project` 参数控制着投影态密度;当 task=dos 时,dos.project 为 **false/true** ;若打开投影, dos.project= ture , 此时 `dos.json` 中将会保存投影态密度的信息；若不打开投影，dos.project = false ；

案例: dos.project = true

参数名称: *dos.EfShift*

默认值: false

可选值: false/true

描述: `dos.EfShift` 参数表示 `dos.range` 的两个能量是否按照自洽的费米能级平移；当 dos.EfShift=fasle, 则 `dos.range` 的能量不按照费米能级平移；当 dos.EfShift=true，则 `dos.range` 的能量按照费米能级平移;

案例: dos.EfShift = true

参数名称: *band.kpointsLabel*

默认值: 无

可选值: n*1 string array

描述: 该参数只有在 task=band 时才能生效; `band.kpointsLabel` 为能带计算时高对称点标签, band.kpointsLabel 数据大小是 band.kpointsCoord 数据大小的 **1/3** ； 比 band.kpointsNumber 数据大小多 **1** ；

案例: band.kpointsLabel = [G,M,K,G]

参数名称: *band.kpointsCoord*

默认值: 无

可选值: 3n*1 real array

描述: 该参数只有在 task=band 时才能生效; **band.kpointsCoord** 为能带计算时高对称点的分数坐标, band.kpointsCoord 数据大小是 band.kpointsLabel 数据大小的 **3 倍**;

案例: band.kpointsCoord = [0, 0, 0, 0.5, 0.5, 0.5, 0, 0, 0.5, 0, 0, 0]

参数名称: *band.kpointsNumber*

默认值: 无

可选值: (n-1)*1 int array

描述: 该参数只有在 task=band 时才能生效; **band.kpointsNumber** 为能带每相邻两个高对称点的间隔, `band.kpointsNumber` 比 `band.kpointsLabel` 数据大小少 **1** ;

案例: band.kpointsNumber = [5, 5, 5]

参数名称: *band.project*

默认值: false

可选值: false/true

描述: `band.project` 参数控制着投影能带；当 task= band 时, band.project 为 **false/true** ；若打开投影，band.project= ture , 此时 `band.json` 中将会保存投影态密度的信息；若不打开投影，band.project = false ；

案例: band.project = true

参数名称: *band.unfolding*

默认值: false

可选值: false/true

描述: `band.unfolding` 参数是能带去折叠的开关；当 task= band 时, band.unfolding 生效（io.band = true 不生效），若设置 band.unfolding= ture , 此时 `band.json` 中将会保存反折叠能带数据；

案例: task = band, band.unfolding = true

参数名称: *band.primitiveUVW*

默认值: 无

可选值: 9*1 real array

描述: `band.primitiveUVW` 能带去折叠计算时，超胞的晶格常数乘上 UVW 系数等于原胞的晶格矢量；

案例: band.primitiveUVW = [0.0, 0.5, 0.5, 0.5, 0.0, 0.5, 0.5, 0.5, 0.0]

参数名称: *potential.type*

默认值: total

可选值: total/hartree/all

描述: `potential.type` 只在 task=scf/realx 且 io.potential=true、或 task=potential 时生效；当 potential.type 为 hartree 时，`potental.json` 中输出静电势，当为 total 时，potental.json 中输出总的局域势即离子势、静电势和交换关联势的和，当为 all 时，`potental.json` 中同时输出两个势；

案例: potential.type = all

参数名称: *corr.chargedSystem*

默认值: false

可选值: false/true

描述: `corr.chargedSystem` 表示当计算带电体系时，可以设置该参数修正带电块体体系的能量；

案例: corr.chargedSystem = true

参数名称: *corr.dipol*

默认值: false

可选值: false/true

描述: `corr.dipol` 表示引入人为的势场 (偶极修正) 来修正真空电势不平整的问题;

案例: corr.dipol = true

参数名称: *corr.dipolDirection*

默认值: 无

可选值: a/b/c

描述: `corr.dipolDirection` 表示偶极修正的方向，a/b/c 分别表示三个晶格常数的方向;

案例: corr.dipolDirection = c

参数名称: *corr.dipolPosition*

默认值: 无

可选值: 3*1 real array

描述: `corr.dipolPosition` 表示偶极子在晶胞的相对位置;

案例: corr.dipolPosition = [0.5, 0.5, 0.5]

参数名称: *corr.dipolEfield*

默认值: 0

可选值: real

描述: `corr.dipolEfield` 表示外加电场的大小，单位为 eV/Å，该参数只在 `corr.dipol = true` 和设置 `corr.dipolDirection` 的情况下生效;

案例: corr.dipolEfield = 0.05

参数名称: *corr.dftu*

默认值: false

可选值: false/true

描述: `corr.dftu` 表示是否引入 hubbard U 来处理强关联体系;

案例: corr.dftu = true

参数名称: *corr.dftuElements*

默认值: 无

可选值: n*1 string array

描述: `corr.dftuElements` 表示设置需要加 U 的元素;

案例: corr.dftuElements = [Ni,O]

参数名称: *corr.dftuOrbital*

默认值: 无

可选值: n*1 string array

描述: `corr.dftuOrbital` 表示设置选中元素上需要加 U 的轨道;

案例: corr.dftuOrbital = [d,s]

参数名称: *corr.dftuU*

默认值: 无

可选值: n*1 real array

描述: `corr.dftuU` 表示设置选中元素选中轨道上需要加 U 值的大小;

案例: corr.dftuU = [8,1]

参数名称: *corr.dftuJ*

默认值: 无

可选值: n*1 real array

描述: `corr.dftuJ` 表示设置选中元素选中轨道上需要加 J 值的大小;

案例: corr.dftuJ = [0.95,0]

参数名称: *corr.VDW*

默认值: false

可选值: false/true

描述: `corr.VDW` 表示是否引入范德瓦尔斯修正;

案例: corr.VDW = true

参数名称: *corr.VDWType*

默认值: D2G

可选值: D2G/D3G/D3BJ

描述: `corr.VDWType` 表示使用哪种范德瓦尔斯修正, D2G 表示 DFT-D2 of Grimme 方法; D3G 表示 DFT-D3 of Grimme 方法; D3BJ 表示 DFT-D3 with Becke-Jonson damping 方法;

案例: corr.VDWType = D3G

参数名称: *pcharge.bandIndex*

默认值: 无

可选值: n*1 int array

描述： `pcharge.bandIndex` 表示部分电荷密度计算时能带的序号；

案例: pcharge.bandIndex = [1,3,4]

参数名称: *pcharge.kpointsIndex*

默认值: 无

可选值: n*1 int array

描述： `pcharge.kpointsIndex` 表示部分电荷密度计算时 K 点的序号；

案例: pcharge.kpointsIndex = [12,14]

参数名称: *pcharge.sumK*

默认值: false

可选值: false/true

描述： `pcharge.sumK` 表示计算部分电荷密度之后保存数据是否将所有 K 点，不同能带的数据相加；

案例: pcharge.sumK = true

参数名称: *neb.springK*

默认值: 5

可选值: real

描述： `neb.springK` 表示过渡态计算中弹簧系数 K；

案例: neb.springK = 7

参数名称: *neb.images*

默认值: 无

可选值: int

描述： `neb.images` 表示过渡态计算中的中间结构的数目；

案例: neb.images = 5

参数名称: *neb.iniFin*

默认值: false

可选值: true/false

描述： `neb.iniFin` 表示过渡态计算中初态结构和末态结构是否进行自洽计算，true 表示进行自洽计算；

案例: neb.iniFin = true

参数名称: *neb.method*

默认值: QN

可选值: LBFGS/CG/QM/QN/QM2/FIRE

描述: `neb.method` 表示过渡态计算中使用的算法;

案例: neb.method = QN

参数名称: *neb.freedom*

默认值: atom

可选值: atom/all

描述: `neb.freedom` 表示过渡态计算中弛豫的自由度，可以选择只弛豫原子，也可放开晶胞进行弛豫;

案例: neb.freedom = all

参数名称: *neb.convergenceType*

默认值: force

可选值: force/energy

描述: `neb.convergenceType` 表示过渡态计算中收敛标准的选择，可以选择受力或以能量为收敛标准;

案例: neb.convergenceType = energy

参数名称: *neb.convergence*

默认值: 0.1/1e-4

可选值: real

描述: `neb.convergence` 表示过渡态计算中受力或能量的收敛标准；选择力为收敛标准时默认值为 0.1，选择能量为收敛标准时默认值为 1e-4;

案例: neb.convergence = 0.01

参数名称: *neb.stepRange*

默认值: 0.1

可选值: int

描述: `neb.stepRange` 表示过渡态计算中结构弛豫的步长;

案例: neb.stepRange = 0.01

参数名称: *neb.max*

默认值: 60

可选值: int

描述: `neb.max` 表示过渡态计算中结构弛豫的最大步数;

案例: neb.max = 300

参数名称: *frequency.dispOrder*

默认值: 1

可选值: 1/2

描述: `frequency.dispOrder` 表示频率计算时原子振动的方式，1 对应中心差分法，有两种振动方式，2 对应四种振动方式；

案例: frequency.dispOrder = 2

参数名称: *frequency.dispRange*

默认值: 0.01

可选值: real

描述: `frequency.dispRange` 表示频率计算时的原子位移；

案例: frequency.dispRange = 0.05

参数名称: *phonon.structureSize*

默认值: [1,1,1]

可选值: 3*1 int array

描述: `phonon.structureSize` 表示声子计算时超胞的大小；

案例: phonon.structureSize = [2,2,2]

参数名称: *phonon.method*

默认值: fd

可选值: fd/dfpt

描述: `phonon.method` 表示声子计算的方式；fd 为有限位移法；dfpt 为密度泛函微扰理论方法；

案例: phonon.method = dfpt

参数名称: *phonon.type*

默认值: phonon

可选值: phonon/band/dos/bandDos

描述: `phonon.type` 表示声子计算哪些性质：phonon 对应计算力常数矩阵或 force set；band 对应计算声子能带；dos 对应计算声子态密度；bandDos 对应计算声子能带及声子态密度；

案例: phonon.type = bandDos

参数名称: *phonon.isDisplacement*

默认值: true

可选值: true/false

描述: `phonon.isDisplacement` 表示 fd 方法计算声子计算时是否进行位移；

案例: phonon.isDisplacement = true

参数名称: *phonon.fdDisplacement*

默认值: 0.01

可选值: real

描述: `phonon.fdDisplacement` 表示 fd 方法计算声子计算时进行位移的大小;

案例: phonon.fdDisplacement = 0.05

参数名称: *phonon.iniPhonon*

默认值: 无

可选值: 指定 phonon.json 的路径

描述: `phonon.iniPhonon` 表示声子能带或态密度计算时读取力常数矩阵或 force set 时的路径;

案例: phonon.iniPhonon = ./phonon.json

参数名称: *phonon.qsamping*

默认值: MP

可选值: MP/G

描述: `phonon.qsamping` 表示计算声子时布里渊区 q 点采样方法，`Monkhorst-Pack` 方法/ `Gamma centered` 方法;

案例: phonon.qsamping = G

参数名称: *phonon.qpoints*

默认值: [1,1,1]

可选值: 3*1 int array

描述: `phonon.qpoints` 表示声子计算时 Q 空间网格取样大小;

案例: phonon.qpoints = [9,9,9]

参数名称: *phonon.qpointsLabel*

默认值: 无

可选值: n*1 string array

描述: `phonon.qpointsLabel` 表示声子能带计算时高对称点标签;

案例: phonon.qpointsLabel = [G,M,K,G]

参数名称: *phonon.qpointsCoord*

默认值: 无

可选值: 3n*1 real array

描述： phonon.qpointsCoord 表示声子能带计算时高对称点坐标；

案例: phonon.qpointsCoord = [0, 0, 0, 0.5, 0.5, 0.5, 0, 0, 0.5, 0, 0, 0]

参数名称: *phonon.qpointsNumber*

默认值: 51

可选值: int

描述： phonon.qpointsNumber 表示声子能带相邻两个高对称点的间隔；

案例: phonon.qpointsNumber = 100

参数名称: *phonon.primitiveUVW*

默认值: [1,0,0,0,1,0,0,0,1]

可选值: 9*1 real array

描述： phonon.primitiveUVW 声子能带计算时，超胞的晶格常数乘上 UVW 系数等于原胞的晶格矢量；

案例: phonon.primitiveUVW = [1,0,0,0,1,0,0,0,1]

参数名称: *phonon.dosRange*

默认值: [0, 40]

可选值: 2*1 real array

描述： phonon.dosRange 表示声子态密度计算能量的区间；

案例: phonon.dosRange = [-15,15]

参数名称: *phonon.dosResolution*

默认值: 0.1

可选值: real

描述： phonon.dosResolution 表示声子态密度计算能量间隔精度；

案例: phonon.dosResolution = 0.01

参数名称: *phonon.dosSigma*

默认值: 0.1

可选值: real

描述： phonon.dosSigma 表示声子态密度计算时的展宽；

案例: phonon.dosSigma = 0.1

参数名称: *phonon.dfptEpsilon*

默认值: false

可选值: false/true

描述: `phonon.dfptEpsilon` 是 phonon.method = dfpt 时控制介电常数计算的开关;

案例: phonon.dfptEpsilon = true

参数名称: *phonon.nac*

默认值: phonon.dfptEpsilon = true 时默认为 true

可选值: false/true

描述: 当 phonon.dfptEpsilon = true 时，若计算能带和态密度，phonon.nac 作为是否使用 non-analytical term correction 的开关;

案例: phonon.nac = false

参数名称: *phonon.thermal*

默认值: false

可选值: false/true

描述: `phonon.thermal` 是 task=phonon 且 phonon.type=dos 或 phonon.type=bandDos 时控制热力学性质计算的开关;

案例: phonon.thermal = true

参数名称: *phonon.thermalRange*

默认值: [0,1200,10]

可选值: 3*1 real array

描述: `phonon.thermalRange` [min_T, max_T, ΔT] 表示热力学性质计算时温度的选取范围以及数据存储间隔;

案例: phonon.thermalRange = [0,1000,10]

参数名称: *elastic.dispOrder*

默认值: 1

可选值: 1/2

描述: `elastic.dispOrder` 表示弹性常数计算时原子振动的方式，1 对应中心差分法，有两种振动方式，2 对应四种振动方式;

案例: elastic.dispOrder = 1

参数名称: *elastic.dispRange*

默认值: 0.01

可选值: real

描述: `elastic.dispRange` 表示弹性常数计算时的原子位移;

案例: elastic.dispRange = 0.05

参数名称: *aimd.ensemble*

默认值: NVE

可选值: NVE/NVT/TS

描述: `aimd.ensemble` 表示分子动力学模拟时选用的系综；TS 对应高温退火过程；

案例: aimd.ensemble = NVE

参数名称: *aimd.thermostat*

默认值: none

可选值: none/andersen/noseHoover

描述: `aimd.thermostat` 表示分子动力学模拟时选用的热浴方法；

案例: aimd.thermostat = andersen

参数名称: *aimd.iniTemp*

默认值: 无

可选值: real

描述: `aimd.iniTemp` 表示分子动力学模拟时的初始温度；

案例: aimd.iniTemp = 1000

参数名称: *aimd.finTemp*

默认值: aimd.iniTemp

可选值: real

描述: `aimd.finTemp` 表示分子动力学模拟时的末态温度；

案例: aimd.finTemp = 1000

参数名称: *aimd.timeStep*

默认值: 无

可选值: real

描述: `aimd.timeStep` 表示分子动力学模拟时的时间步长；

案例: aimd.timeStep = 1

参数名称: *aimd.totalSteps*

默认值: 无

可选值: real

描述: `aimd.totalSteps` 表示分子动力学模拟的总步数；

案例: aimd.totalSteps = 10000

CHAPTER 6

输出文件格式说明

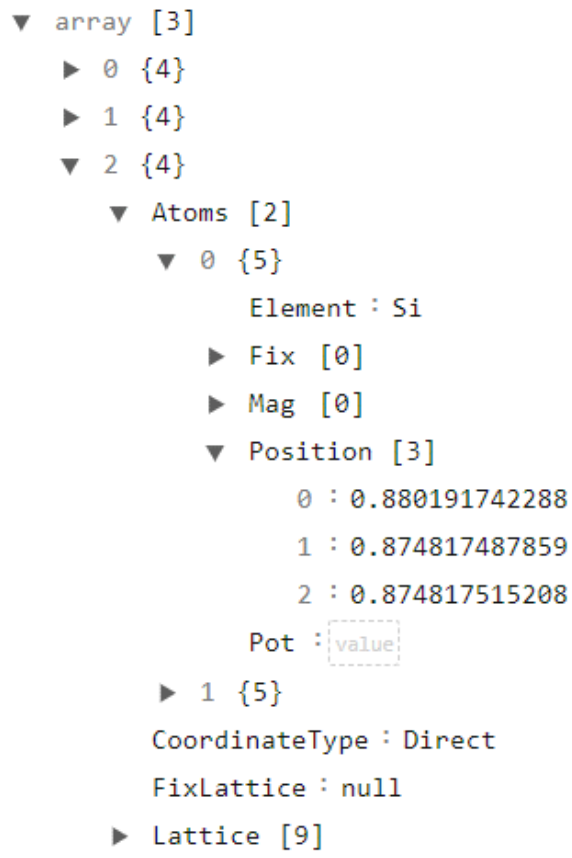
6.1 relax.json

relax.json 为 *task = relax* 时的输出文件，当 *task* 类型为其他时，该文件不输出。

（1）*relax.json* 中包含 **1*N** 个结构体；

（2）每个结构体保存着结构弛豫中每个离子步优化之后的结构信息，具体包含 3 部分信息:

- *Atoms* 中保存着每个原子的信息，每个 Atoms 中包含 5 部分信息：
  - *Element* 为原子的元素名;
  - *Fix* 为原子是否固定;
  - *Mag* 为原子的初始磁矩;
  - *Position* 为原子的对应的坐标；
  - *Pot* 为原子的赝势信息；
- *CoordinateType* 为坐标类型：**Direct** 为分数坐标, **Cartesian** 为笛卡尔坐标；
- *FixLattice* 为晶胞向量是否固定；
- *Lattice* 为晶格常数；



6.2 system.json

除 *task = frequency* 、*task = elastic* 、*task = epsilon* 外，其余 *task* 计算输出 *system.json* 文件。

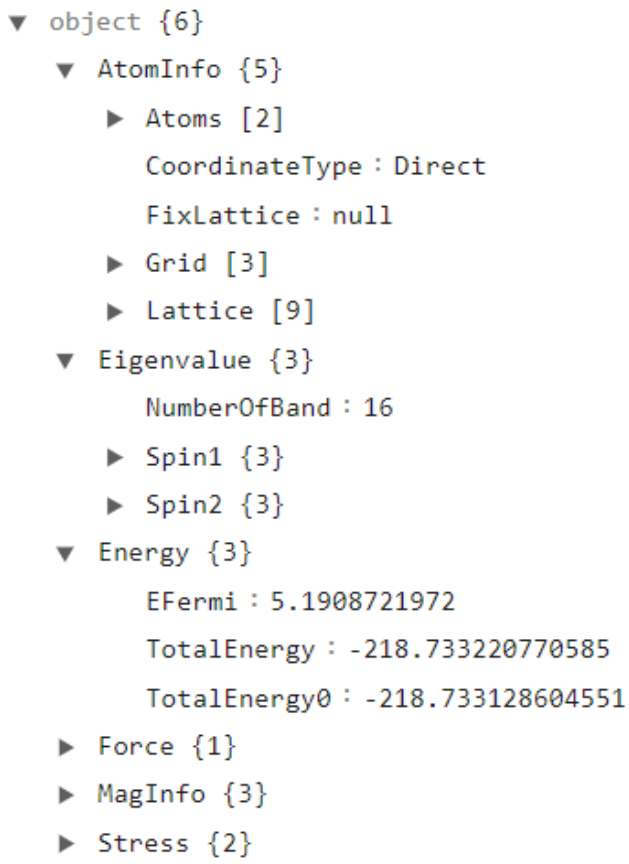
（1）*system.json* 中包含 6 个结构体;

（2）**AtomInfo** 中包含 5 部分信息:

- **Atoms** 中保存着每个原子的信息；每个 Atoms 中包含 2 部分信息：
  - *Element* 为对应原子的元素名;
  - *Position* 为对应原子的对应的坐标;
- *CoordinateType* 为坐标类型：**Direct** 为分数坐标, **Cartesian** 为笛卡尔坐标;
- *FixLattice* 为是否固定晶胞的开关，在 structure.as 文件的第三行添加 Fix 标签控制;
- *Grid* 为格点数目，绘制 3 维格点数据的时候需要使用
- *Lattice* 为晶格常数;

（3）**Eigenvalue** 中包含 3 部分信息:

- *NumberOfBand* 为总能带数;
- **Spin1** 和 **Spin2** (若考虑自旋) 保存着能量本征值，其数据格式相同；以 **Spin1** 为例，该结构包含 3 部分信息：
  - *BandEnergies* 中保存着每条能带每个 **K 点**对应的能量值;


▼ object {6}
▼ AtomInfo {5}
▶ Atoms [2]
CoordinateType : Direct
FixLattice : null
▶ Grid [3]
▶ Lattice [9]
▼ Eigenvalue {3}
NumberOfBand : 16
▶ Spin1 {3}
▶ Spin2 {3}
▼ Energy {3}
EFermi : 5.1908721972
TotalEnergy : -218.733220770585
TotalEnergy0 : -218.733128604551
▶ Force {1}
▶ MagInfo {3}
▶ Stress {2}

- *Kpoints* 中保存着每个 **K 点**的分数坐标；
- *Occupation* 中保存着每条能带每个 **K 点**对应的电子占据值；

（4）**Energy** 中包含 3 部分信息：

- *EFermi* 为自洽或最后一步结构弛豫的费米能级的值；
- *TotalEnergy* 为自洽或最后一步结构弛豫的总能的值；
- *TotalEnergy0* 为自洽或最后一步结构弛豫的 **sgmia** 趋于 **0** 时近似总能的值；

（5）**Force** 中包含 1 部分信息：

- *ForceOnAtoms* 中保存着每个原子对应 **XYZ** 三个方向上的力的值；

（6）*MagInfo* 中包含着磁矩的信息；

- *TotalMagOnAtomsX* 给出各原子 X 方向上的总磁矩，Y，Z 同理，该信息输出需在 input.in 文件中打开 io.magProject 开关；
- *TotalMagOnOrbitalX* 给出 X 方向磁矩分布在各轨道的分量，Y, Z 同理，该信息输出需在 input.in 文件中打开 io.magProject 开关；

（7）**Stress** 中包含 2 部分信息：

- *Direction* 中保存了 **Total stress** 的分量符号；
- *Total* 中保存了在每个 Direction 分量上 **stress** 的值；

6.3 dos.json

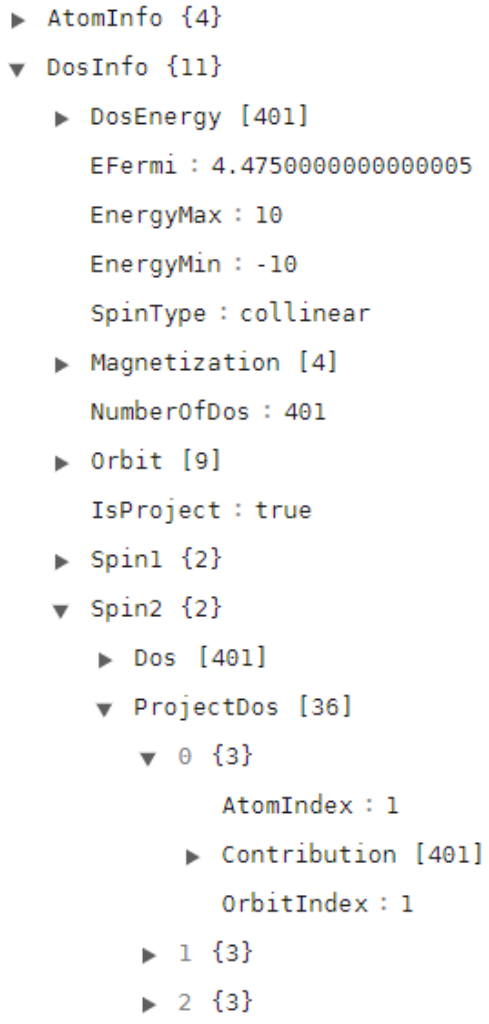
dos.json 为 task = dos 时的输出文件，当 task 类型为其他时，该文件不输出。

（1）*dos.json* 中包含 2 个结构体；

（2）*AtomInfo* 同 *system.json* 中的 AtomInfo 数据；

（3）**DosInfo** 中包含 11 部分信息：

- *DosEnergy* 为所计算态密度的能量点的数据；
- *EFermi* 为费米能级值；
- *EnergyMax* 为所计算态密度的最大能量值；
- *EnergyMin* 为所计算态密度的最小能量值；
- *SpinType* 为计算时所设置的磁性类型；
- *Magnetization* 为投影到每个原子上的磁矩分量；
- *NumberOfDos* 为所计算的态密度的所有能量点的数目；
- *Orbit* 为投影轨道信息；
- *IsProject* 为态密度计算时是否打开了投影功能；
- *Spin1* 和 *Spin2* 保存着态密度相关的数据，其数据格式相同；*Spin1* 中包含 2 部分信息：
  - *Dos* 为总的态密度的数据；
  - *ProjectDos* 包含原子数 × 轨道数目大小的结构体，每个结构体中包含着 3 部分信息：

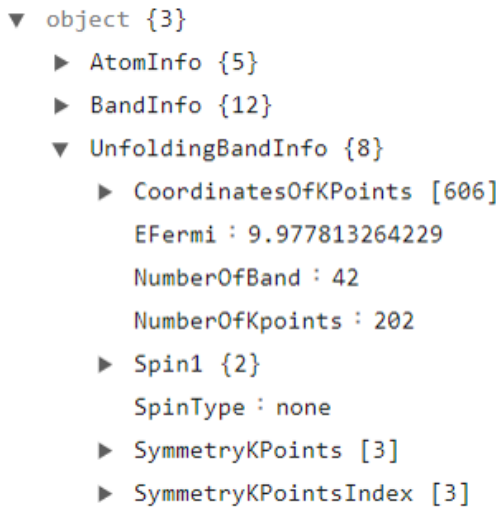

▶ AtomInfo {4}
▼ DosInfo {11}
▶ DosEnergy [401]
EFermi : 4.4750000000000005
EnergyMax : 10
EnergyMin : -10
SpinType : collinear
▶ Magnetization [4]
NumberOfDos : 401
▶ Orbit [9]
IsProject : true
▶ Spin1 {2}
▼ Spin2 {2}
▶ Dos [401]
▼ ProjectDos [36]
▼ 0 {3}
AtomIndex : 1
▶ Contribution [401]
OrbitIndex : 1
▶ 1 {3}
▶ 2 {3}

- *AtomIndex* 为当前 **Contribution** 对应的元素序号，该序号与 AtomInfo 中的 **Atoms** 的序号相同；
- *Contribution* 为 **AtomIndex** 对应原子和 **OrbitIndex** 对应轨道在每个能量点下的态密度贡献值；
- *OrbitIndex* 为当前 **Contribution** 对应的轨道序号，该序号与 DosInfo 中的 Orbit 的序号相同；

备注：注意：当只有 Spin1 的时候，Spin1 为总的信息当 Spin1 和 Spin2 同时存在的时候，Spin1 为自旋向上的信息，Spin2 为自旋向下的信息。

6.4 band.json

band.json 为 *task* = *band* 时的输出文件，当 *task* 类型为其他时，该文件不输出。



（1）此例为能带反折叠计算的 *band.json* 文件，包含 3 个结构体；

（2）*AtomInfo* 同 *system.json* 中的 **AtomInfo** 数据；

（3）**BandInfo** 中包含 12 部分信息：

- *BandGap* 为带隙值；
- *CBM* 为导带位置；
- *VBM* 为价带位置；
- *CoordinatesOfKPoints* 为能带计算时 K 点的分数坐标；
- *EFermi* 为费米能级值；
- *SpinType* 为计算时所设置的磁性类型；
- *SymmetryKPoints* 为能带计算时高对称 K 点的字母表示；
- *SymmetryKPointsIndex* 为能带计算时高对称 K 点的起始序号；

- *IsProject* 为能带计算时是否打开了投影功能；
- *NumberOfband* 为所计算的能带的所有能带的数目；
- *Orbit* 为投影轨道信息；

• *Spin1* 和 *Spin2* 保存着能带相关的数据，其数据格式相同, *Spin1* 中包含 2 部分信息：

– *Band* 为总的能带的数据；

– *ProjectBand* 包含原子数 × 轨道数目大小的结构体，每个结构体中包含着 3 部分信息：

– *AtomIndex* 为当前 **Contribution** 对应的元素序号，该序号与 AtomInfo 中的 **Atoms** 的序号相同；

– *Contribution* 为 **AtomIndex** 对应原子和 **OrbitIndex** 对应轨道在每个能量点下的能带贡献值；

– *OrbitIndex* 为当前 **Contribution** 对应的轨道序号，该序号与 BandInfo 中的 **Orbit** 的序号相同；

（4）**UnfoldingBandInfo** 中包含 8 部分信息，除 *Spin* 部分结构与 **BandInfo** 不同，其余数据结构均一致：

• *Spin1* 保存着反折叠能带的相关的数据，*Spin1* 中包含 2 部分信息：

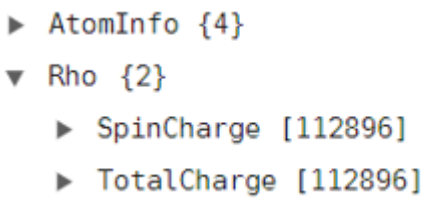
– *UnfoldingBand* 为总的能带数据；

– *Weight* 为能带的权重值；

备注：注意：当只有 Spin1 的时候，Spin1 为总的信息当 Spin1 和 Spin2 同时存在的时候，Spin1 为自旋向上的信息，Spin2 为自旋向下的信息。

6.5 rho.json

rho.json 为 *task = scf* 以及 *task =relax* 时的输出文件，当 *task* 类型为其他时，该文件不输出。



（1）*rho.json* 中包含 2 个结构体；

（2）**AtomInfo** 同 *system.json* 中的 **AtomInfo** 数据；

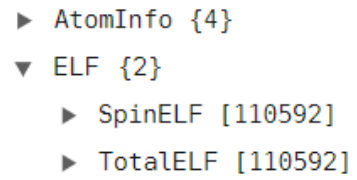
（3）**Rho** 中包含信息：

• *TotalCharge* 和 *SpinCharge* ； *TotalCharge* 和 *SpinCharge* 数据格式相同，都表示电荷密度其数据大小为 AtomInfo 中 3 个 Grid 的乘积；

备注：注意：在当只有 TotalCharge 的时候，TotalCharge 为总的电荷密度；当 TotalCharge 和 SpinCharge 同时存在的时候，TotalCharge 为总的电荷密度，SpinCharge 为自旋电荷密度；

6.6 elf.json

elf.json 可为 *task = elf* 和 *task = scf* 且 *io.elf* 设置为 *true* 以及 *task = relax* 且 *io.elf* 设置为 *true* 这 *3* 种 *task* 下的输出文件，当 *task* 类型为其他时，该文件不输出。



（1）*elf.json* 中包含 2 个结构体；

（2）**AtomInfo** 同 *system.json* 中的 AtomInfo 数据；

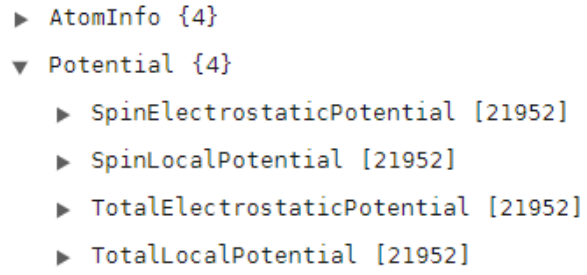
（3）**elf** 中包含信息：

- *TotalELF* 和 *SpinELF* ； *TotalELF* 和 *SpinELF* 数据格式相同，都表示电子局域密度的数据，其数据大小为 AtomInfo 中 3 个 Grid 的乘积；

备注：注意：在当只有 TotalELF 的时候，TotalELF 为总的局域电荷密度；当 TotalELF 和 SpinELF 同时存在的时候，TotalELF 为总的局域电荷密度，SpinELF 为自旋局域电荷密度；

6.7 potential.json

potential.json 可为 *task = potential* 和 *task = scf* 且 *io.potential* 设置为 *true* 以及 *task = relax* 且 *io.potential* 设置为 *true* 这 *3* 种 *task* 下的输出文件，当 *task* 类型为其他时，该文件不输出。



（1）*potential.json* 中包含 2 个结构体；

（2）**AtomInfo** 同 *system.json* 中的 AtomInfo 数据；

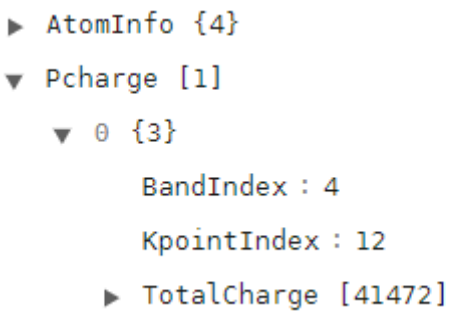
（3）**potential** 中包含 4 部分信息：

- *TotalElectrostaticPotential* 和 *SpinElectrostaticPotential* ；*TotalElectrostaticPotential* 和 *SpinElectrostaticPotential* 数据格式相同，都表示静电势函数的数据，其数据大小为 AtomInfo 中 **3 个 Grid** 的乘积；
- *TotalLocalPotential* 和 *SpinLocalPotential* ； *TotalLocalPotential* 和 *SpinLocalPotential* 数据格式相同，都表示局域势函数的数据，其数据大小为 AtomInfo 中 **3 个 Grid** 的乘积；

备注：注意：potential.type = hartree/total/all，分别保存 ElectrostaticPotential/LocalPotential/ElectrostaticPotential & LocalPotential 三种类型的势函数，对应静电势、局域势、静电势和局域势

6.8 pcharge.json

pcharge.json 为 *task* = *pcharge* 时的输出文件，当 *task* 类型为其他时，该文件不输出。



（1）*pcharge.json* 中包含 2 个结构体；

（2）**AtomInfo** 同 *system.json* 中的 **AtomInfo** 数据；

（3）**Pcharge** 在本例中包含 1 部分信息，为第 4 条能带的电荷密度，若计算 N 条能带，Pcharge 会写出 N 部分信息：

- *BandIndex* 和 *KpointIndex* 分别对应计算能带的序号及计算所用 K 点；
- *TotalCharge* 表示电荷密度的数据，其数据大小为 AtomInfo 中 **3 个 Grid** 的乘积；

备注：注意：当 pcharge.sumK = true 时，Pcharge 只包含一部分信息，为所有计算的能带的电荷密度之和。

6.9 optical.json

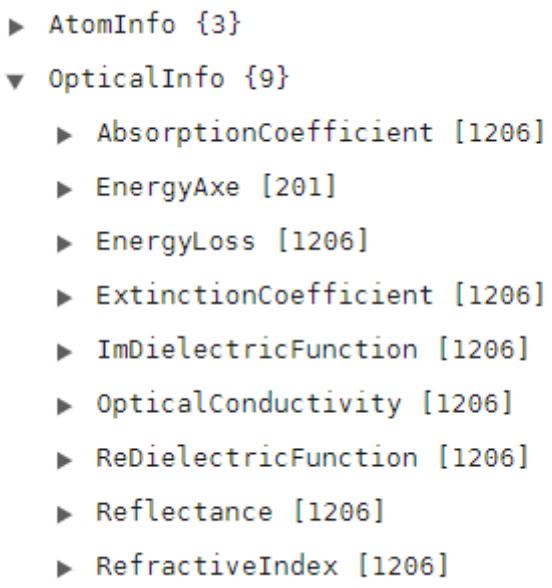
optical.json 为 *task* = *scf* 且 *io.optical* = *true* 时的输出文件，当 *task* 类型为其他时，该文件不输出。

（1）*optical.json* 中包含 2 个结构体；

（2）**AtomInfo** 同 *system.json* 中的 **AtomInfo** 数据；

（3）**OpticalInfo** 中包含 9 部分信息：

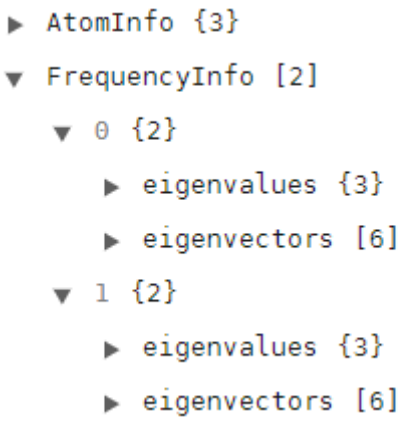
- *AbsorptionCoefficient* 、*ExtinctionCoefficient* 、*OpticalConductivity* 、*Reflectance* 、*RefractiveIndex* 数据格式相同，分别对应吸收系数、消光系数、光电导率、反射率和折射率，其数据大小为 计算能量间隔点数*6；
- *ImDielectricFunction* 和 *ReDielectricFunction* 数据格式相同，分别对应介电函数的虚部和实部，其数据大小为 计算能量间隔点数*6；



- *EnergyAxe* 和 *EnergyLoss* 分别对应能量计算范围，能量损失系数。

6.10 frequency.json

frequency.json 为 *task* = *frequency* 时的输出文件，当 *task* 类型为其他时，该文件不输出。



（1）*frequency.json* 中包含 2 个结构体；

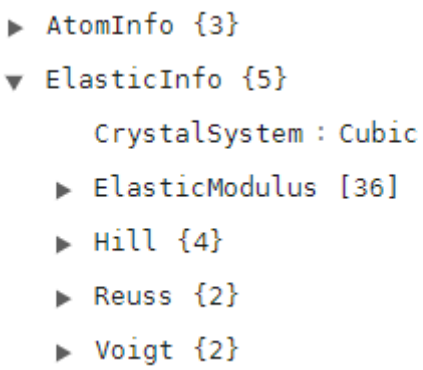
（2）**AtomInfo** 同 *system.json* 中的 **AtomInfo** 数据；

（3）**FrequencyInfo** 中包含 2 部分信息：

- 两部分数据对应两种类型的频率，分别为 `f` 和 `f/i` ；
- 每种频率的信息分为两部分，*eigenvalues* 和 *eigenvectors* ，对应频率的本征值和本征向量，本征向量给出 `x` 、 `y` 、 `z` 、 `dx`、`dy` 、`dz` 6 个方向上的分量。

6.11 elastic.json

elastic.json 为 *task* = *elastic* 时的输出文件，当 *task* 类型为其他时，该文件不输出。



（1）*elastic.json* 中包含 2 个结构体；

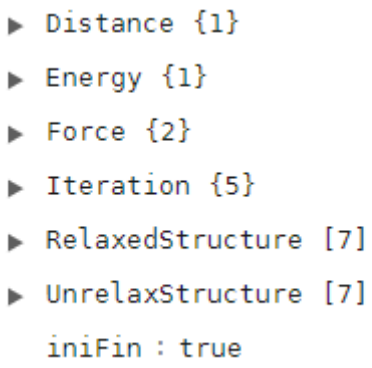
（2）**AtomInfo** 同 *system.json* 中的 **AtomInfo** 数据；

（3）**ElasticInfo** 中包含 5 部分信息：

- *CrystalSystem* 给出体系的晶体类型；
- *ElasticModulus* 数据大小为 36，给出弹性模量的数据。
- *Hill* 、*Reuss* 和 *Voigt* 分别给出三种模型下计算出的体积模量和剪切模量的数据，此外 *Hill* 模型还给出杨氏模量和泊松比。

6.12 neb.json

neb.json 为 *task* = *neb* 时的输出文件，当 *task* 类型为其他时，该文件不输出。



（1）*neb.json* 中包含 7 个结构体；

（2）*Distance* 给出每两个相邻构型（00 与 01、01 与 02...05 与 06）之间坐标的距离；

（3）*Energy* 体系总能，共有 **image 数量** +2 个数据

（4）*Force* 分为两部分，分别为最大受力和切向力，共有 **image 数量**个数据

（5）*Iteration* 给出所有插点结构在优化过程中的最大受力和总能数据，共有 **image 数量**个数据

（6）*RelaxedStructure* 给出所有构型在优化完成后的原子和晶胞信息

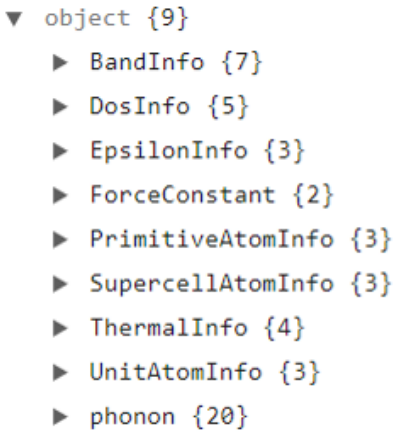
（7）*UnrelaxStructure* 给出所有构型优化前的原子和晶胞信息

（8）*iniFin* 表示是否对初态和末态结构进行自洽计算。

备注：注意：当 iniFin = False 时，Distance 部分不提供 00 与 01 及 05 与 06 之间的坐标距离，Energy 部分数据总数等于 image 数量。

6.13 phonon.json

phonon.json 为 *task* = *phonon* 时的输出文件，当 *task* 类型为其他时，该文件不输出。



（1）此例中 phonon.method = dfpt, phonon.type = bandDos，*phonon.json* 中包含 8 个结构体；

（2）**BandInfo** 包含 7 部分信息，结构与 *band.json* 文件中 *BandInfo* 部分相似，

- *CoordinatesOfQPoints* 为计算声子能带时 Q 点的分数坐标；
- *NumberOfBand* 为计算声子能带时所有能带数目；
- *NumberOfQPoints* 为计算声子能带时高对称 Q 点的字母表示；
- *Spin1* 为计算声子能带的相关数据；
- *SpinType* 为计算声子能带时设置的自旋类型；
- *SymmetryQPoints* 为声子能带计算时高对称 Q 点的字母表示；
- *SymmetryQPointsIndex* 为声子能带计算时高对称 Q 点的起始序号；

（3）**DosInfo** 包含 6 部分信息，结构与 *dos.json* 文件中 *DosInfo* 部分相似，

- *DosEnergy* 为计算声子态密度时能量点的数据；

- *EnergyMax* 为计算声子态密度的最大能量值；
- *EnergyMin* 为计算声子态密度的最小能量值；
- *NumberOfDos* 为计算声子态密度时所有能量点的数目；
- *Spin1* 为计算声子态密度的相关数据；
- *SpinType* 为计算声子态密度时设置的自旋类型；

（4）**EpsilonInfo** 包含 3 部分信息，给出介电常数相关数据，详细解析请见第（10）条。

（5）**ForceConstant** 分为 2 部分，*ConstantIndex* 给出力学常数数组的维度，*ConstantValue* 为力学常数的数据，数据大小为 Index 中各维度之乘积。

（6）**PrimitiveAtomInfo** 分为 3 部分，给出原胞原子信息。

（7）**SupercellAtomInfo** 分为 3 部分，给出超胞原子信息。

（8）**UnitAtomInfo** 分为 3 部分，给出单胞原子信息。

（9）**phonon** 分为 20 部分

- *dfptEpsilon* 为介电常数的计算开关；
- *dosRange* 为计算声子态密度时能量的范围；
- *dosResolution* 为计算声子态密度的能量间隔；
- *dosSigma* 为计算声子态密度时的展宽；
- *eigenVectors* 为本征向量的开关；
- *fdDisplacement* 为计算声子时原子位移的大小，该参数在 fd 方法时有效；
- *iniPhonon* 是否提供声子计算的 json 文件；
- *isDisplacement* 是否有位移；
- *method* 计算声子的方法；
- *nac* 为 nac 的开关；
- *primitiveUVW* 计算声子能带时是否将超胞还原成原胞；
- *qpoints* 为计算声子能带时 Q 点的大小；
- *qpointsCoord* 为计算声子能带时 Q 点的坐标；
- *qpointsLabel* 为计算声子能带时 Q 点的标签；
- *qpointsNumber* 为计算声子能带时 Q 点的数目；
- *qsamping* 为计算声子能带时 Q 点采样的方法；
- *structureSize* 为计算声子时扩胞参数；
- *thermal* 为热力学计算的开关；
- *thermalRange* 为热力学计算的温度范围；
- *type* 为声子计算的任务类型。

（10）当 task=phonon 时，若打开介电常数或热力学性质计算的开关，*phonon.json* 结构中会分别写入 *EpsilonInfo* 和 *ThermalInfo* 数据，如下图所示：

备注：注意：当 phonon.method = fd 时, phonon.json 文件会写入 Displacements 和 Forceset 两部分数据。

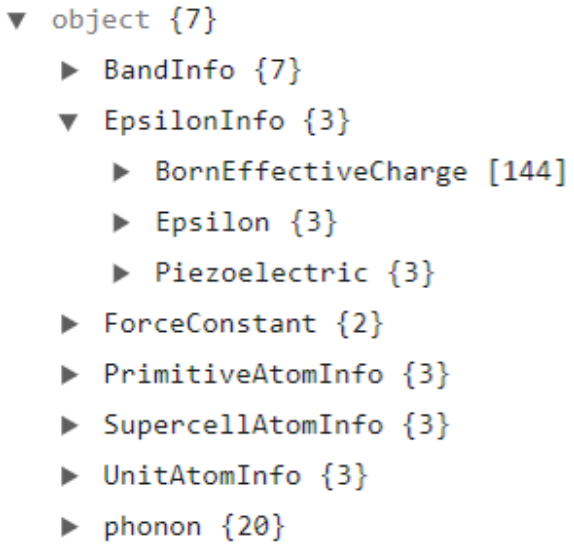


图 1: 其中 **EpsilonInfo** 分为 3 部分：

- *BornEfectiveCharge* 给出波恩有效电荷数据；
- *Epsilon* 给出介电常数数据；
- *Piezoelectric* 给出压电数据；

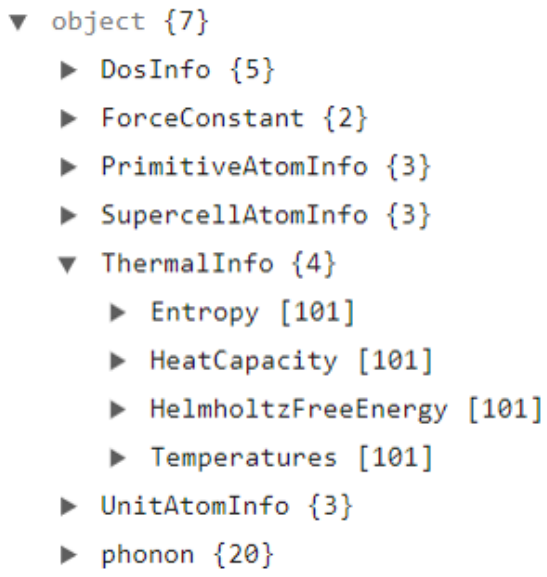
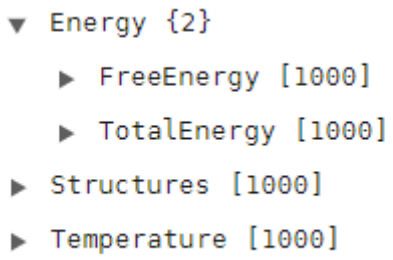


图 2: 其中 **ThermalInfo** 分为 4 部分：

- *Entropy* 给出熵的数据；
- *HeatCapacity* 给出热容数据；
- *HelmholtzFreeEnergy* 给出 Helmholtz 自由能数据；
- *Temperature* 给出温度数据；

6.14 aimd.json

aimd.json 为 *task* = *aimd* 时的输出文件，当 *task* 类型为其他时，该文件不输出。



（1）*aimd.json* 中包含 3 个结构体；

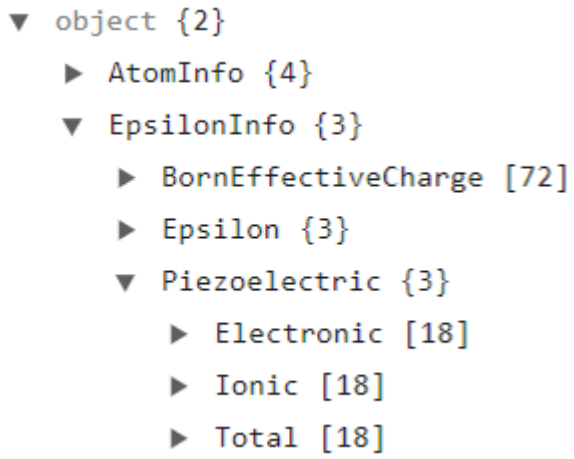
（2）**Energy** 包含两部分信息，分别为分子动力学模拟每一步的自由能数据和总能数据；数据大小取决于模拟总步数；

（3）**structure** 给出分子动力学模拟每一步的原子位置数据；

（4）**Temperature** 给出分子动力学模拟每一步的温度数据。

6.15 epsilon.json

epsilon.json 为 *task* = *epsilon* 时的输出文件，当 *task* 类型为其他时，该文件不输出。



（1）*epsilon.json* 中包含 **2** 个结构体；

（2）*AtomInfo* 同 *system.json* 中的 AtomInfo 数据；

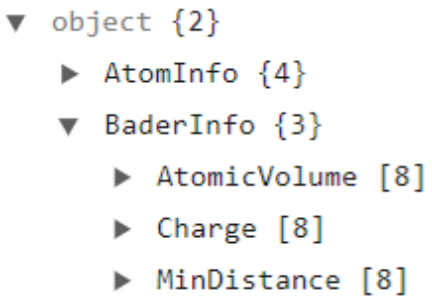
（3）*EpsilonInfo* 包含 3 部分数据：

- *BornEffectiveCharge* 为波恩有效电荷数据；

- *Epsilon* 为介电常数数据，分为电子贡献、离子贡献和总的介电常数；
- *Piezoelectric* 为压电张量的数据，分为电子贡献、离子贡献和总的压电张量。

6.16 bader.json

bader.json 为 *task* = *scf* 且 *io.bader* = *true* 时的输出文件，当 *task* 类型为其他时，该文件不输出。



（1）*bader.json* 中包含 **2** 个结构体；

（2）*AtomInfo* 同 *system.json* 中的 AtomInfo 数据；

（3）*BaderInfo* 包含 3 部分数据：

- *AtomicVolume* 为 bader 体积数据；
- *Charge* 为 bader 电荷数据；
- *MinDistance* 为最近原子距离。

6.17 polarization.json

polarization.json 为 *task* = *polarization* 时的输出文件，当 *task* 类型为其他时，该文件不输出。

（1）*polarization.json* 中包含 **5** 个结构体；

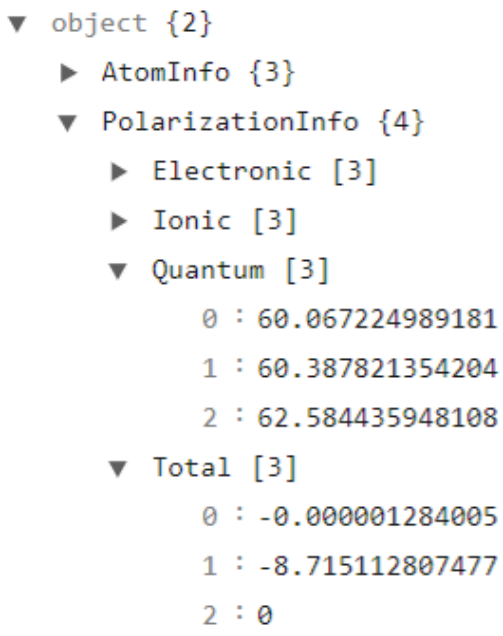
（2）*AtomInfo* 同 *system.json* 中的 AtomInfo 数据；

（3）*Electronic* 为电子贡献数据；

（4）*Ionic* 为离子贡献数据；

（5）*Quantum* 为极化量子数据；

（6）*Total* 为总的极化数据；


▼ object {2}
▶ AtomInfo {3}
▼ PolarizationInfo {4}
▶ Electronic [3]
▶ Ionic [3]
▼ Quantum [3]
0 : 60.067224989181
1 : 60.387821354204
2 : 62.584435948108
▼ Total [3]
0 : -0.000001284005
1 : -8.715112807477
2 : 0

CHAPTER 7

辅助工具使用教程

7.1 env 环境部署

依赖环境: **Python3.6 及以上**

- 依赖库安装：

(1) **Pymatgen 库**：使用 `pip install pymatgen` 安装；

(2) **Statsmodels 库**：使用 `pip install statsmodels` 安装；

(3) **Dspawpy 库**：使用 `pip install -i https://test.pypi.org/simple/ dspawpy==0.3.0` 安装；

7.2 rho 电荷密度数据处理

DS-PAW 使用 *pymatgen* 的画图程序来显示的二维数据，使用 *VESTA* 来显示三维数据。以应用教程中 Si 体系的 `band.json` 和 `dos.json` 为例子：

(1) 在计算目录中新建一个 *trans_rho.py* 文件，内容如下：

```
1 import json
2 from dspawpy.io import write_VESTA_format
3
4 with open("./rho.json","r") as file:
5 rho = json.load(file)
6
7 write_VESTA_format(rho["AtomInfo"], [rho["Rho"]["TotalCharge"]], "DS-PAW-rho.vesta")
```

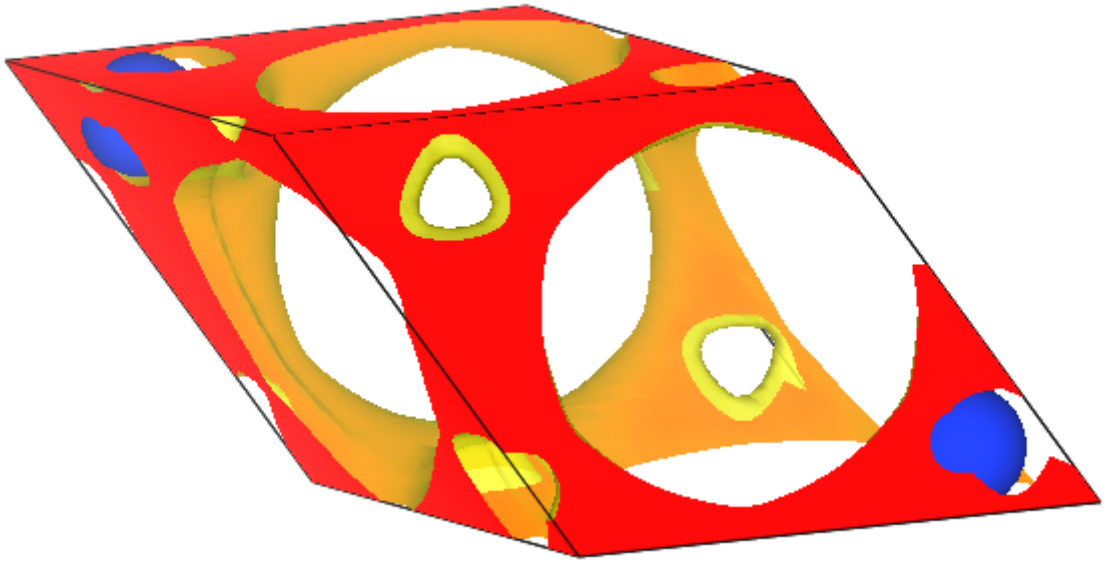
- 第 1，2 行为需要导入的 **python** 库函数；

- 第 4，5 行为需要读取的 `json` 文件；
- 第 7 行为执行 *write_VESTA_format* 函数，将数据保存为 VESTA 可识别的格式，其中 **rho["AtomInfo"]** 为读取 `rho.json` 中的晶格及坐标信息, **[rho["Rho"]["TotalCharge"]]** 为 `rho.json` 中总的电荷密度的三维数据, `DS-PAW-rho.vesta` 表示生成的 **vesta** 格式数据；

(2) 执行 *python trans_rho.py* ；

(3) 得到转换后的文件 `DS-PAW-rho.vesta` ，将其重命名为 `CHGCAR.vasp` 以便在 **VESTA** 显示；

(4) 将 `CHGCAR.vasp` 文件拖入 VESTA 中则可得到如下的电荷密度图



7.3 band 能带数据处理

以不开自旋时的 MoS2 体系的 `band.json` 为例子：

（1）**普通能带处理**：

创建 *bandplot.py* 文件，具体代码如下：

```
from pymatgen.electronic_structure.plotter import BSPlotter
from dspawpy.io import get_band_data

band_data = get_band_data("./band.json")

bsp = BSPlotter(bs=band_data)
plt = bsp.get_plot()
plt.savefig("bandplot.png",img_format="png")
plt.show()
```

- 第 1、2 行为需要导入的 **python** 库函数，这里用到了 pymatgen 中的 **BSPloter** 库，以及 *dspawpy.io* 中的 `get_band_data` 函数；
- 第 4 行为读取 `band.json` 文件构造 **band_data** 这个结构体；
- 第 6 行为调用 BSPloter 处理 **band_data** 数据；
- 第 7 行为使用 `bsp` 下面的 **get_plot** 进行画图；
- 第 8 行为保存图片的名称和格式，将数据保存为 png 格式命名为 **bandplot.png** ；
- 第 9 行为显示图片，如果用户运行的环境无法直接显示图片，用户可以将该行注释，之后打开 **bandplot.png** 文件查看。

(2) 执行 *python bandplot.py*

(3) 生成 `bandplot.png` 文件

知识点：

1. *Pymatgen* 中主要用到 *electronic_structure.plotter* 模块，基本上该模块下的 *80%* 的功能都能使用；
2. *dspawpy.io* 模块为 *DS-PAW* 的接口模块；
3. 使用 *get_band_data* 函数可以将 *DS-PAW* 计算得到的 *band.json* 文件转化为 *pymatgen* 支持的格式；
4. 使用 *BSPlotter* 模块获取到 *DS-PAW* 计算的 *band.json* 的数据；
5. 使用 *BSPlotter* 模块中 *get_plot* 函数绘制能带图；
6. 使用 *savefig* 函数可以将能带图以不同格式保存；
7. 使用 *show* 函数能够直接显示能带图，如果用户的机器上不支持图形界面，可以用 # 注释 *show* 函数；

执行代码可以得到以下能带图：

（2）**将能带投影到每一种元素然后分别作图，通过线条的粗细来表示该元素对该轨道的贡献**：

创建 *bandplot_elt_projected.py* 文件，具体代码如下：

```
1  from pymatgen.electronic_structure.plotter import BSPlotterProjected
2  from dspawpy.io import get_band_data
3
4  band_data = get_band_data("./band.json")
5
6  # 构建投影band数据实例
7  bsp = BSPlotterProjected(bs=band_data)
8  # 绘制按元素投影图片,返回matplotlib.pyplot
9  plt = bsp.get_elt_projected_plots()
10 plt.savefig("bandplot_elt_projected.png",img_format="png")
11 plt.show()
```

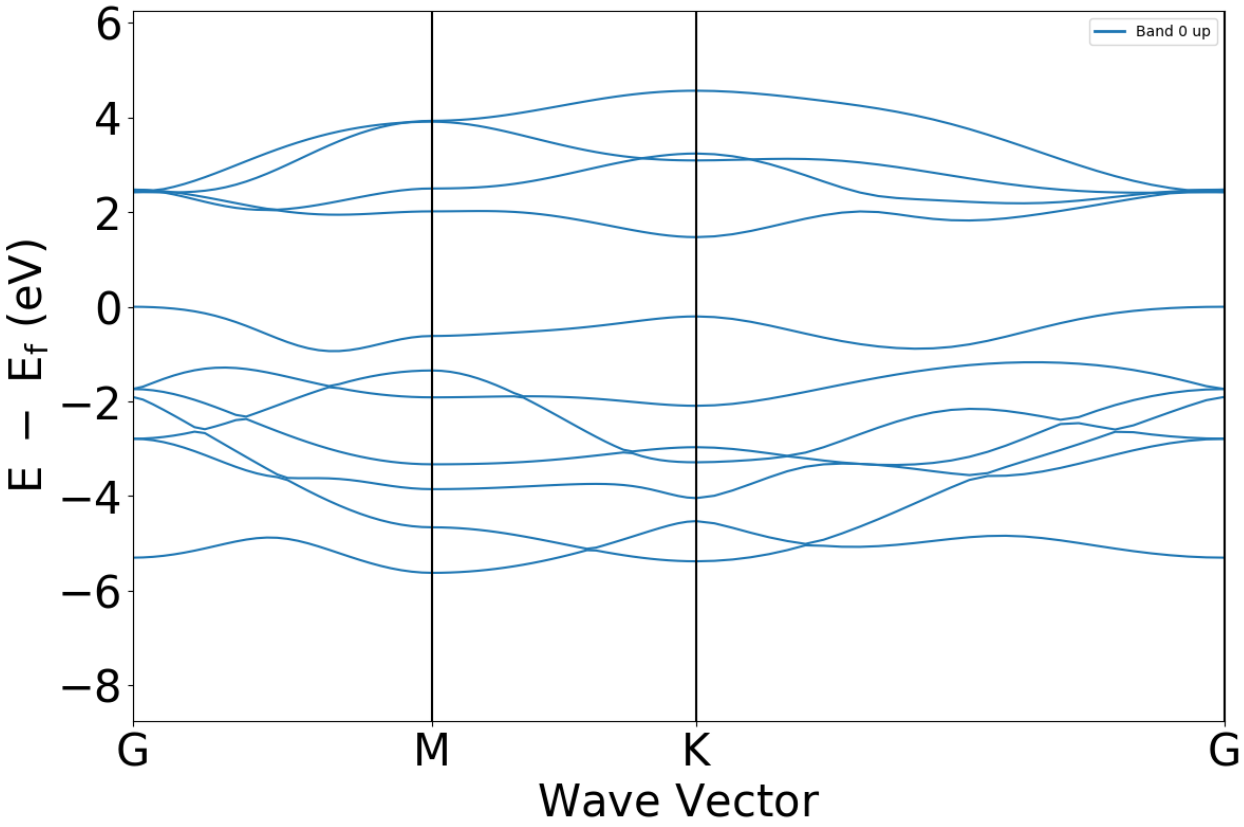
知识点：

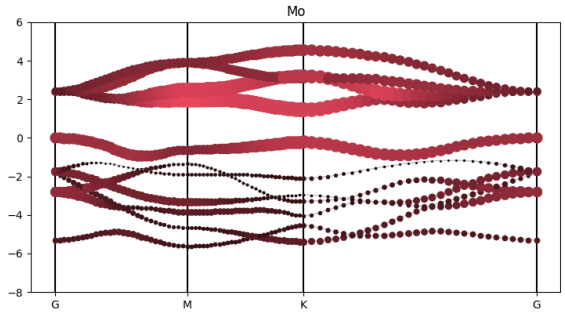
1. 用户如果需要绘制能带投影的数据，此时需要使用 *BSPlotterProjected* 模块；
2. 使用 *BSPlotterProjected* 模块中 *get_elt_projected_plots* 函数能够绘制每种元素对轨道贡献的能带图；

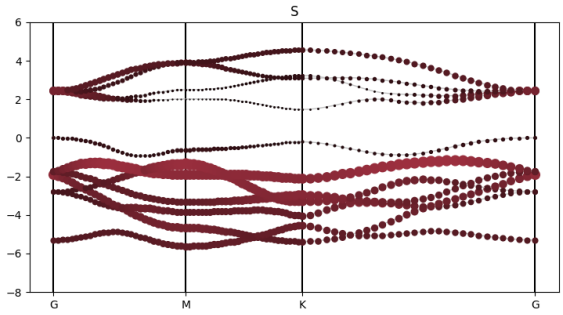
执行代码可以得到以下能带图：

（3）**将能带投影到每一种元素然后分别作图，通过不同的颜色来表示该元素对该轨道的贡献：**

创建 *bandplot_elt_projected_color.py* 文件，具体代码如下：


Band 0 up
6
4
2
0
−2
−4
−6
−8
E − E_f (eV)
G
M
K
G
Wave Vector


Mo
6
4
2
0
−2
−4
−6
−8
G
M
K
G

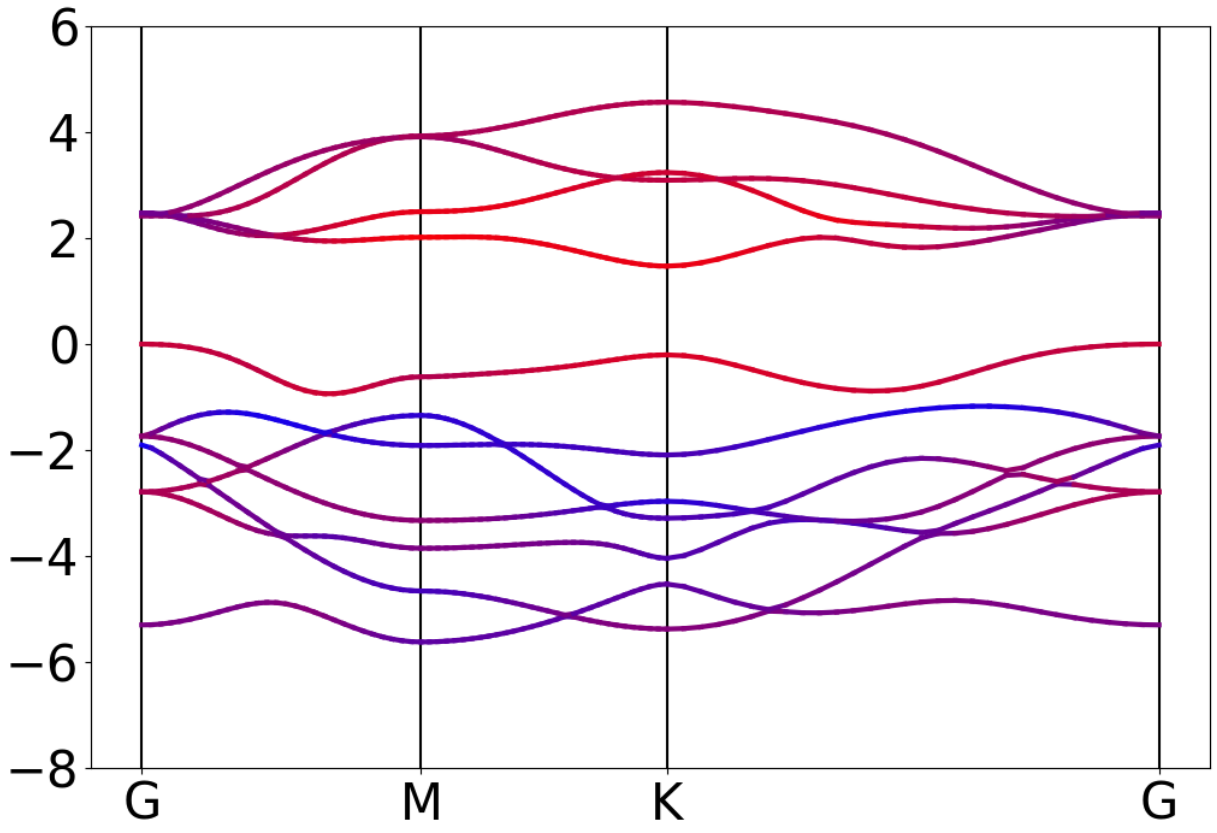

S
6
4
2
0
−2
−4
−6
−8
G
M
K
G

```python
from pymatgen.electronic_structure.plotter import BSPlotterProjected
from dspawpy.io import get_band_data

band_data = get_band_data("./data/band.json")

bsp = BSPlotterProjected(bs=band_data)
# 绘制投影到元素band图片,返回matplotlib.pyplot({'Mo':['d'],'S':['s']})
plt = bsp.get_elt_projected_plots_color()
plt.savefig("bandplot_elt_projected_color.png",img_format="png")
plt.show()
```

知识点：

使用 *BSPlotterProjected* 模块中 *bandplot_elt_projected_color* 函数能够绘制每种元素对轨道贡献的能带图，用红绿蓝三原色来表示 *spd* 轨道；

执行代码可以得到以下能带图：



（4）能带投影到不同元素的不同轨道（L）：

创建 *bandplot_projected.py* 文件，具体代码如下：

```python
from pymatgen.electronic_structure.plotter import BSPlotterProjected
from dspawpy.io import get_band_data

band_data = get_band_data("./data/band.json")
```

(续下页)

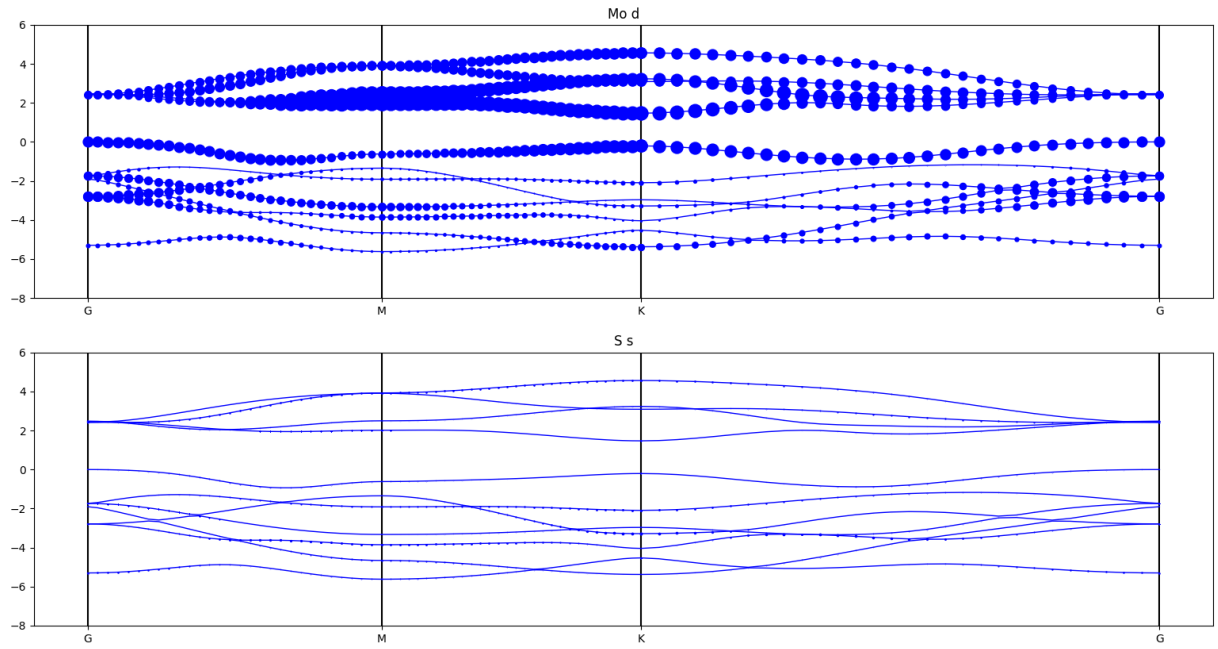
(接上页)

```

bsp = BSPlotterProjected(bs=band_data)

# 绘制投影到元素某些轨道band，返回matplotlib.pyplot
plt = bsp.get_projected_plots_dots({'Mo':['d'],'S':['s']})
plt.savefig("bandplot_projected.png",img_format="png")
plt.show()
```

知识点：

1. 使用 *BSPlotterProjected* 模块中 *get_projected_plots_dots* 可以让用户来自定义需要绘制的某种元素某种轨道 *(L)* 的能带图；
2. 例如 *get_projected_plots_dots ({'Mo':['d'],'S':['s']})* 就是绘制 *Mo* 的 *d* 轨道和 *S* 的 *s* 轨道；

执行代码可以得到以下能带图：



（5）将能带投影到不同原子的不同轨道 (M)：

创建 *bandplot_plots_dots_patom_pmorb.py* 文件，具体代码如下：

```
from pymatgen.electronic_structure.plotter import BSPlotterProjected
from dspawpy.io import get_band_data

band_data = get_band_data("./band.json")
bsp = BSPlotterProjected(bs=band_data)

plt = bsp.get_projected_plots_dots_patom_pmorb(dictio={'Mo':['px','py','pz']}, dictpa=
↪{'Mo':[1]})

plt.savefig("bandplot_projected_plots_dots_patom_pmorb.png",img_format="png")
plt.show()
```

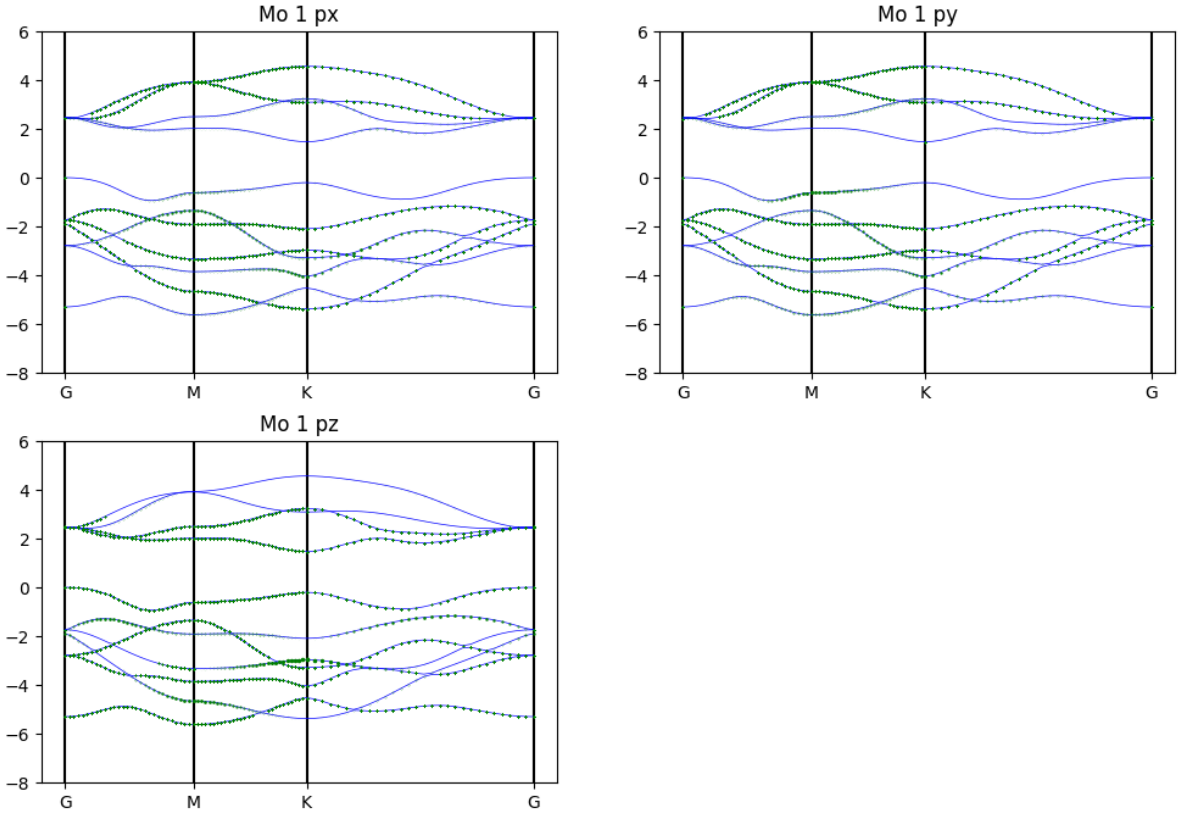
- 第 1、2 行为需要导入的 **python** 库函数，这里用到了 pymatgen 中的 **BSPloter** 库，以及 *dspawpy.io* 中的 `get_band_data` 函数；
- 第 4 行为读取 `band.json` 文件构造 **band_data** 这个结构体；
- 第 5 行为调用 **BSPlotterProjected** 处理 **band_data** 数据；
- 第 7 行为使用 `bsp` 下面的 **get_projected_plots_dots** 进行画图，用户可以根据自己的体系和需要绘制的投影轨道对参数进行修改；
- 第 9 行为保存图片的名称和格式，这里将数据保存为 png 格式命名为 **band-plot_projected_plots_dots_patom_pmorb.png**；
- 第 10 行为显示图片，如果用户运行的环境无法直接显示图片，用户可以将该行注释，之后打开 **png** 文件查看。

知识点：

1. 使用 *BSPlotterProjected* 模块中 *get_projected_plots_dots_patom_pmorb* 的自由度更高，可以让用户来自定义需要绘制的某种原子某种轨道 *(M)* 的能带图；
2. *dictpa* 指定原子，*dictio* 指定该原子的轨道；
3. *get_projected_plots_dots_patom_pmorb* 函数中还可以使用 *sum_atoms* 功能将多个原子的贡献相加，该功能用户可以自行根据使用方法进行尝试；

注意：get_projected_plots_dots_patom_pmorb 函数中序号是从 1 开始的；

执行代码可以得到以下能带图：



7.4 dos 态密度数据处理

以应用教程中的反铁磁 NiO 体系的 `dos.json` 为例子：

（1）**总的态密度**：

创建 *dosplot_total.py* 文件，具体代码如下：

```
1  from pymatgen.electronic_structure.plotter import DosPlotter
2  from dspawpy.io import get_dos_data
3
4  dos_data = get_dos_data("dos.json")
5  dos_plotter = DosPlotter(stack=False,zero_at_efermi=True)
6  dos_plotter.add_dos('total dos',dos=dos_data)
7
8  plt = dos_plotter.get_plot()
9  plt.savefig("dos_total.png",img_format="png")
10 plt.show()
```

- 第 1、2 行为需要导入的 **python** 库函数，这里用到了 pymatgen 中的 **BSPloter** 库，以及 `dspawpy.io` 中的 `get_dos_data` 函数；
- 第 4 行为读取 `dos.json` 文件构造 **dos_data** 这个结构体；
- 第 5 行为调用 `DosPlotter` 处理 **dos_data** 数据，括号中 **stack=False** 表示态密度作图方式为曲线，**stack=True** 表示作图方式为面积，**zero_at_efermi** 控制是否将零点设为费米能级；
- 第 6 行使用 `add_dos` 来定义要处理的对象 **dos** ，调用前边定义好的 dos 对象 `dos_data`
- 第 8 行使用 `dos_plotter` 下面的 get_plot 进行画图；
- 第 9 行为保存图片的名称和格式，这里将数据保存为 png 格式命名为 **dos_total.png** ；
- 第 10 行为显示图片，如果用户运行的环境无法直接显示图片，用户可以将第该行注释，之后打开 **png** 文件查看。

知识点：

1. 使用 *get_dos_data* 函数可以将 *DS-PAW* 计算得到的 *dos.json* 文件转化为 *pymatgen* 支持的格式；
2. 使用 *DosPlotter* 模块获取到 *DS-PAW* 计算的 *dos.sjon* 的数据；
3. *DosPlotter* 函数可以传递参数：*stack* 参数表示画态密度是否加阴影，*zero_at_efermi* 表示是否在态密度图中进行将费米能量置零，这里设置 *stack=False* , *zero_at_efermi=False* ；
4. 使用 *DosPlotter* 模块中 *add_dos* 获取态密度的数据；
5. *DosPlotter* 模块中 *et_plot* 函数绘制态密度图；

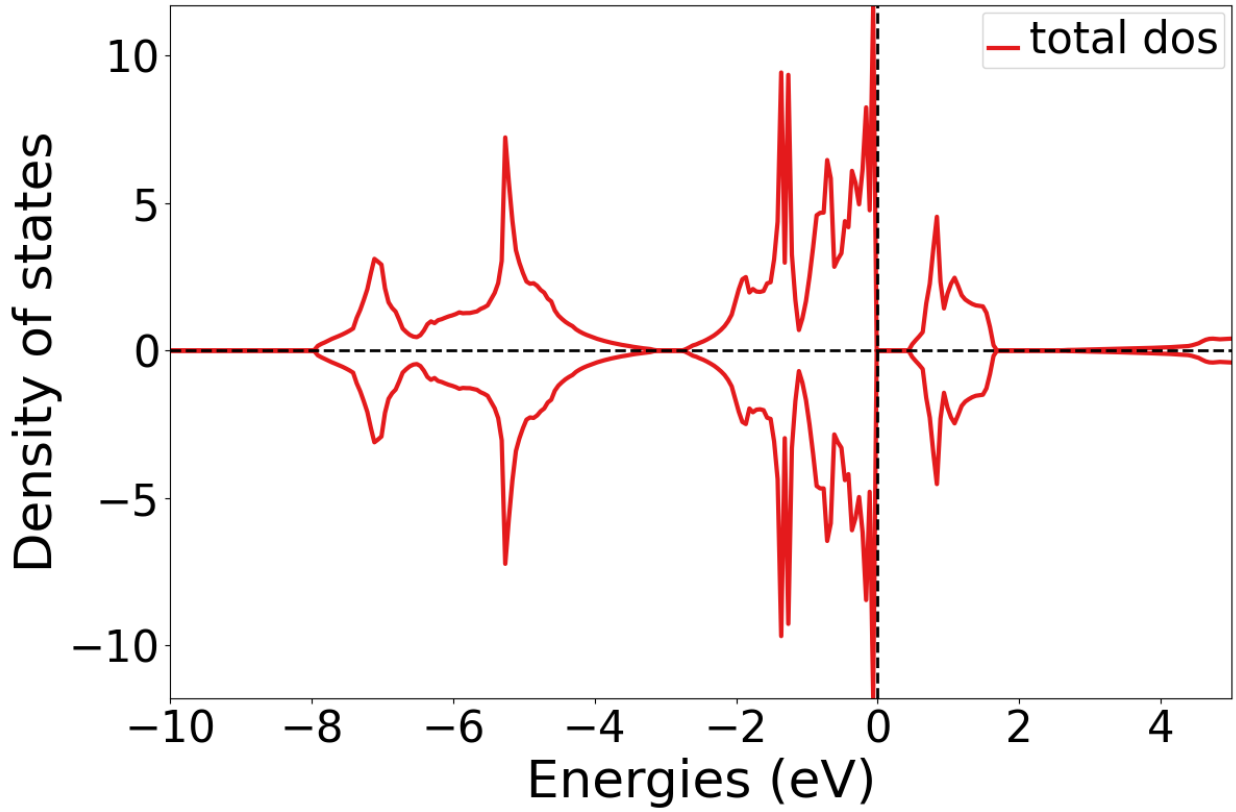
执行代码可以得到以下态密度图：

（2）**将态密度投影到不同的轨道上 (L)：**

创建 *dosplot_spd.py* 文件，具体代码如下：

```
1  from pymatgen.electronic_structure.plotter import DosPlotter
2  from dspawpy.io import get_dos_data
3
```

(续下页)



(接上页)

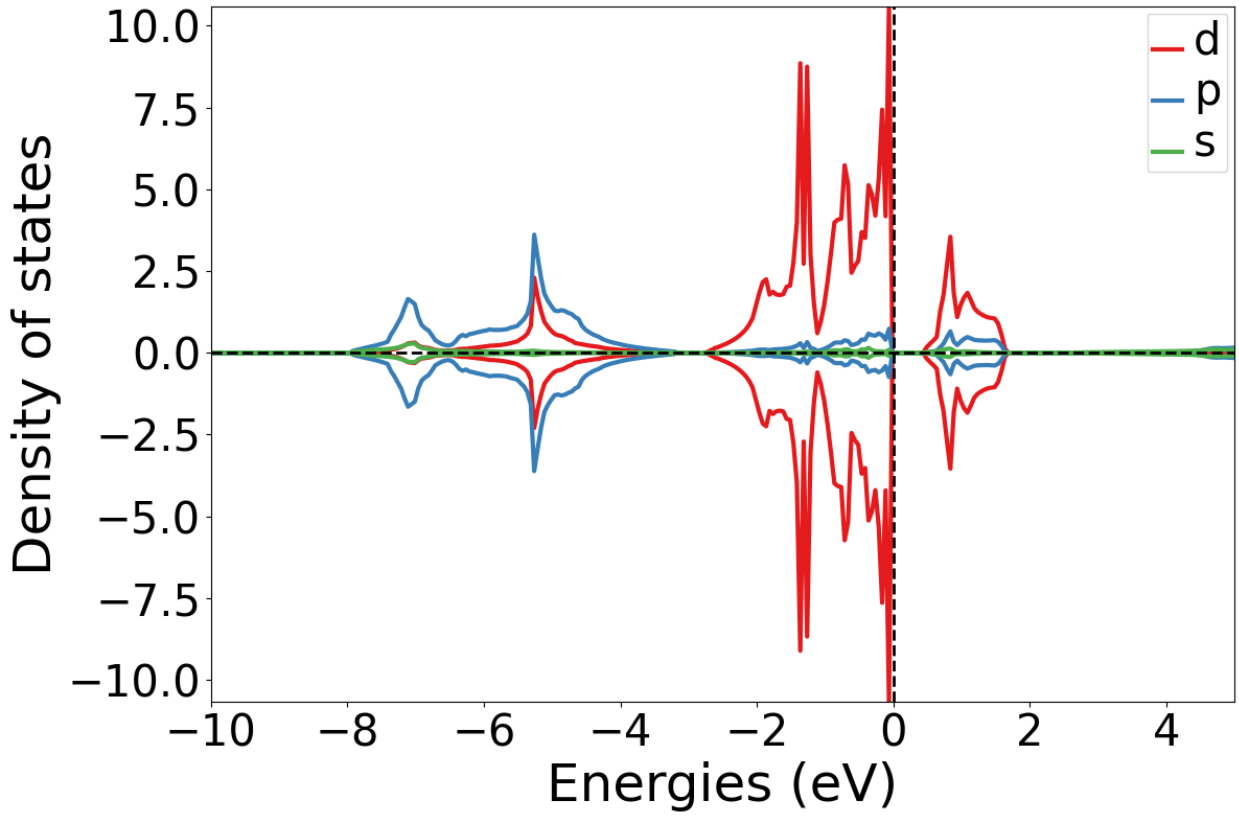
```
4  dos_data = get_dos_data("dos.json")
5  dos_plotter = DosPlotter(stack=False,zero_at_efermi=True)
6  dos_plotter.add_dos_dict(dos_data.get_spd_dos())
7
8  plt = dos_plotter.get_plot()
9  plt.savefig("dos_spd.png",img_format="png")
10 plt.show()
```

- 第 1、2 行是导入需要的 **python** 库函数，这里用到了 **pymatgen** 中的 **DosPlotter** 库，以及 *dspawpy.io* 中的 `get_dos_data` 函数；
- 第 4 行为读取 `dos.json` 文件构造 **dos_data** 这个结构体；
- 第 5 行为调用 BSPlotterProjected 处理 **dos_data** 数据，括号中 **stack=False** 表示态密度作图方式为曲线，**stack=True** 表示作图方式为面积，**zero_at_efermi** 控制是否将零点设为费米能级；
- 第 6 行使用 `add_dos_dict` 获取投影信息，对 `dos_data` 进行 *get_spd_dos* 操作，获得投影到轨道的信息；
- 第 7 行为保存图片的名称和格式，这里将数据保存为 png 格式命名为 **dos_spd.png** ；
- 第 8 行为显示图片，如果用户运行的环境无法直接显示图片，用户可以将该行注释，之后打开 **dos_spd.png** 文件查看。

知识点：

1. 使用 *DosPlotter* 模块中 *add_dos_dict* 函数获取投影态密度的数据，之后使用 *get_spd_dos* 将投影信息按照 *spd* 轨道投影输出；

执行代码可以得到以下态密度图：



（3）**将态密度投影到不同的元素上**：

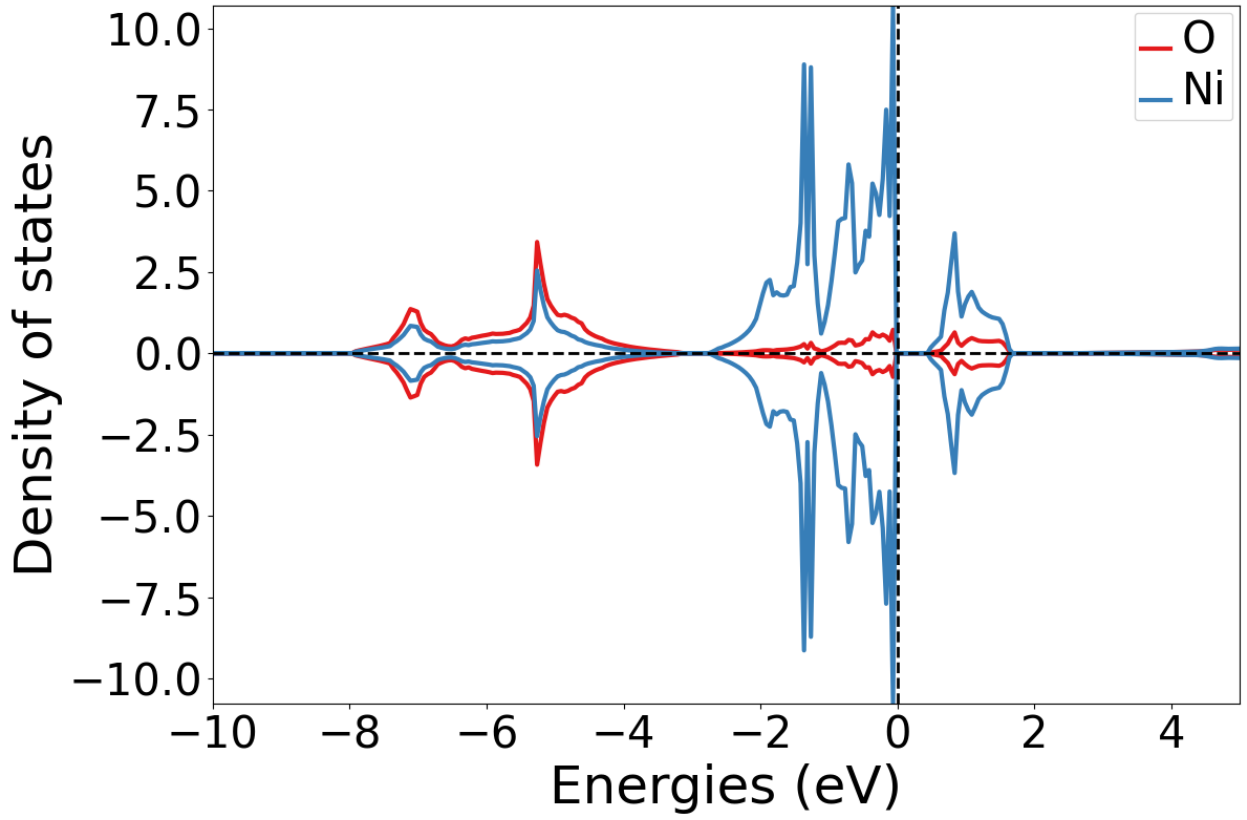
创建 *dosplot_element.py* 文件，具体代码如下：

```
# 导入pymatgen DosPlotter模块
from pymatgen.electronic_structure.plotter import DosPlotter
# 导入读取DS-PAW dos.json数据模块
from dspawpy.io import get_dos_data


dos_data = get_dos_data("dos.json")
dos_plotter = DosPlotter(stack=False,zero_at_efermi=True)

#按元素添加dos
dos_plotter.add_dos_dict(dos_data.get_element_dos())

#返回matplotlib.pyplot
plt = dos_plotter.get_plot()
plt.savefig("dos_element.png",img_format="png")
plt.show()
```

知识点：

1. 使用 *DosPlotter* 模块中 *add_dos_dict* 函数获取投影态密度的数据，之后使用 *get_element_dos* 将投影信息按照不同元素投影输出；

执行代码可以得到以下态密度图：



（4）**将态密度投影到不同的元素的不同轨道 (M) 上：**

创建 *dosplot_element_orbital.py* 文件，具体代码如下：

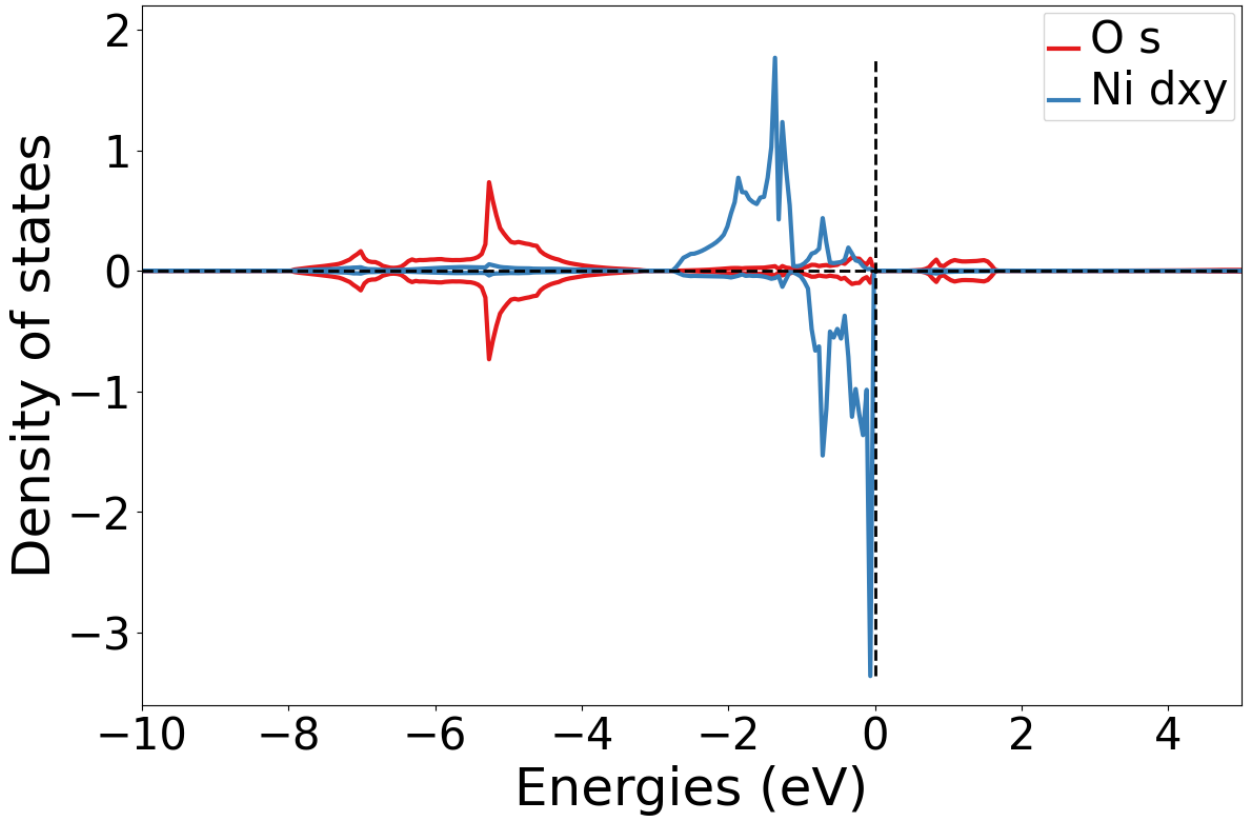
```
from pymatgen.electronic_structure.plotter import DosPlotter
from pymatgen.electronic_structure.core import Orbital
from dspawpy.io import get_dos_data

dos_data = get_dos_data("dos.json")
dos_plotter = DosPlotter(stack=False,zero_at_efermi=False)
dos_plotter.add_dos("Ni dxy",dos_data.get_site_orbital_dos(dos_data.structure[0],
↪Orbital(4)))
#dos_plotter.save_plot("dos_element_orbital.png",img_format="png")
dos_plotter.show()
```

知识点：

1. 使用 *get_site_orbital_dos* 函数提取 *dos* 数据中特定原子，特定轨道的贡献，这是获取第 *2* 个原子的第 *5* 个轨道的态密度 *dos_data.structure[0],Orbital(4)* ；
2. 之后使用 *DosPlotter* 模块中 *add_dos* 函数进行画图；

注意：get_site_orbital_dos 函数中序号是从 0 开始的；

执行代码可以得到以下态密度图：



（5）将态密度投影到不同的原子的 t2g 轨道和 eg 轨道上：

以应用案例 NiO 的反铁磁计算得到的 `dos.json` 为例，

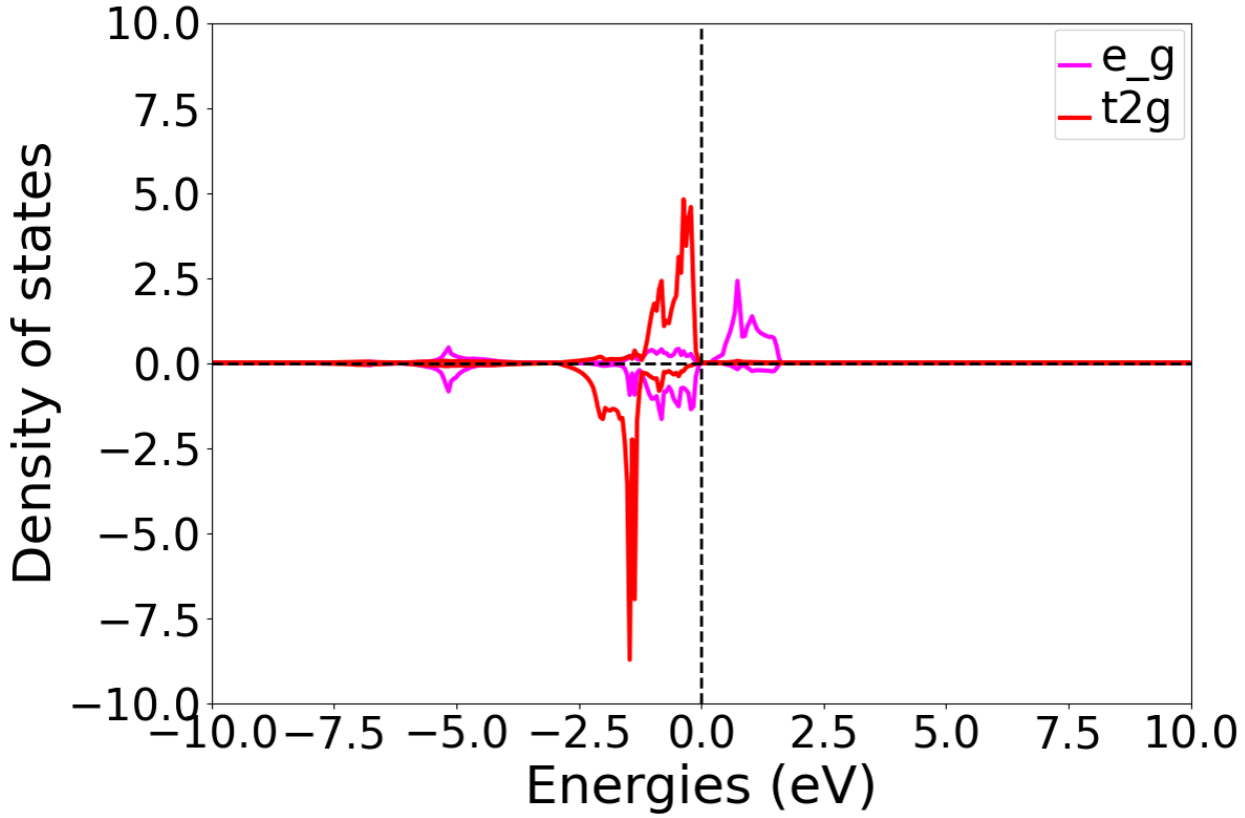
创建 *dosplot_t2g_eg.py* 文件，具体代码如下：

```
1 from pymatgen.electronic_structure.plotter import DosPlotter
2 from dspawpy.io import get_dos_data
3
4 dos_data = get_dos_data("./dos.json")
5 dos_plotter = DosPlotter(stack=False,zero_at_efermi=True)
6 dos_plotter.add_dos_dict(dos_data.get_site_t2g_eg_resolved_dos(dos_data.structure[1]))
7 dos_plotter.save_plot("dos_t2g_eg.png",img_format="png")
8 dos_plotter.show()
```

- 第 1，2 行为需要导入的 **python** 库函数，这里用到了 pymatgen 中的 **DosPlotter** 库，以及我们自己写的 `dspawpy.io` 中的 *get_dos_data* 函数；
- 第 4 行为读取 `dos.json` 文件构造 `dos_data` 结构体；
- 第 5 行为调用 DosPlotter 处理 dos_data 数据，括号中 **stack=False** 表示态密度作图方式为曲线，**stack=True** 表示作图方式为面积，*zero_at_efermi* 控制是否将零点设为 **费米能级**；
- 第 6 行使用 *add_dos_dict* 获取 **投影信息**，对 dos_data 进行 *get_site_t2g_eg_resolved_dos* 操作，获得 **投影到轨道**的信息；
- 第 7 行为保存图片的名称和格式，这里将数据保存为 png 格式命名为 **dos_t2g_eg.png** ；
- 第 8 行为显示图片，如果用户运行的环境无法直接显示图片，用户可以将该行注释了，之后打开 **dos_t2g_eg.png** 文件查看。

(2) 执行 *python dosplot_t2g_eg.py*；

(3) 生成 `dos_t2g_eg.png` 文件



知识点：

1. 使用 *get_site_t2g_eg_resolved_dos* 函数提取 *dos* 数据中特定原子的 *t2g* 和 *eg* 轨道的贡献，这是获取第 *1* 个原子的 *t2g* 和 *eg* 轨道的贡献；
2. 之后使用 *DosPlotter* 模块中 *add_dos_dict* 函数进行画图；

7.5 bandDos 能带和态密度共同显示

以应用教程中 Si 体系的 `band.json` 和 `dos.json` 为例：

（1）**将能带和态密度显示在一张图上：**

创建 *banddosplot.py* 文件，具体代码如下：

```
# 导入pymatgen BSDOSPlotter 画图模块
from pymatgen.electronic_structure.plotter import BSDOSPlotter
# 导入dspawpy 读取DS-PAW band.json,dos.json模块
from dspawpy.io import get_band_data,get_dos_data

```

(续下页)

(接上页)

```
# 读取dos.json, 参数dos.json文件路径
dos_data = get_dos_data("./data/dos.json")
# 读取band.json, 参数band.json文件路径
band_data = get_band_data("./data/band.json")

bdp = BSDOSPlotter()

# 绘制BandDos图，返回matplotlib.pyplot
plt = bdp.get_plot(bs=band_data,dos=dos_data)

# 保存png图片
plt.savefig("banddos.png",img_format="png")
# 显示图片
plt.show()
```

知识点：

1. 首先使用 *dspawpy.io* 中的 *get_band_data* 和 *get_dos_data* 模块获取 *band.json* 和 *dos.json* 的数据；
2. 之后使用 *BSDOSPlotter* 模块中的 *get_plots* 函数直接进行画图；
3. 如果需要将投影能带和投影态密度画在一起，可以参考前面单独画图的代码。

执行代码可以得到以下能带态密度图：

7.6 potential 势函数数据处理

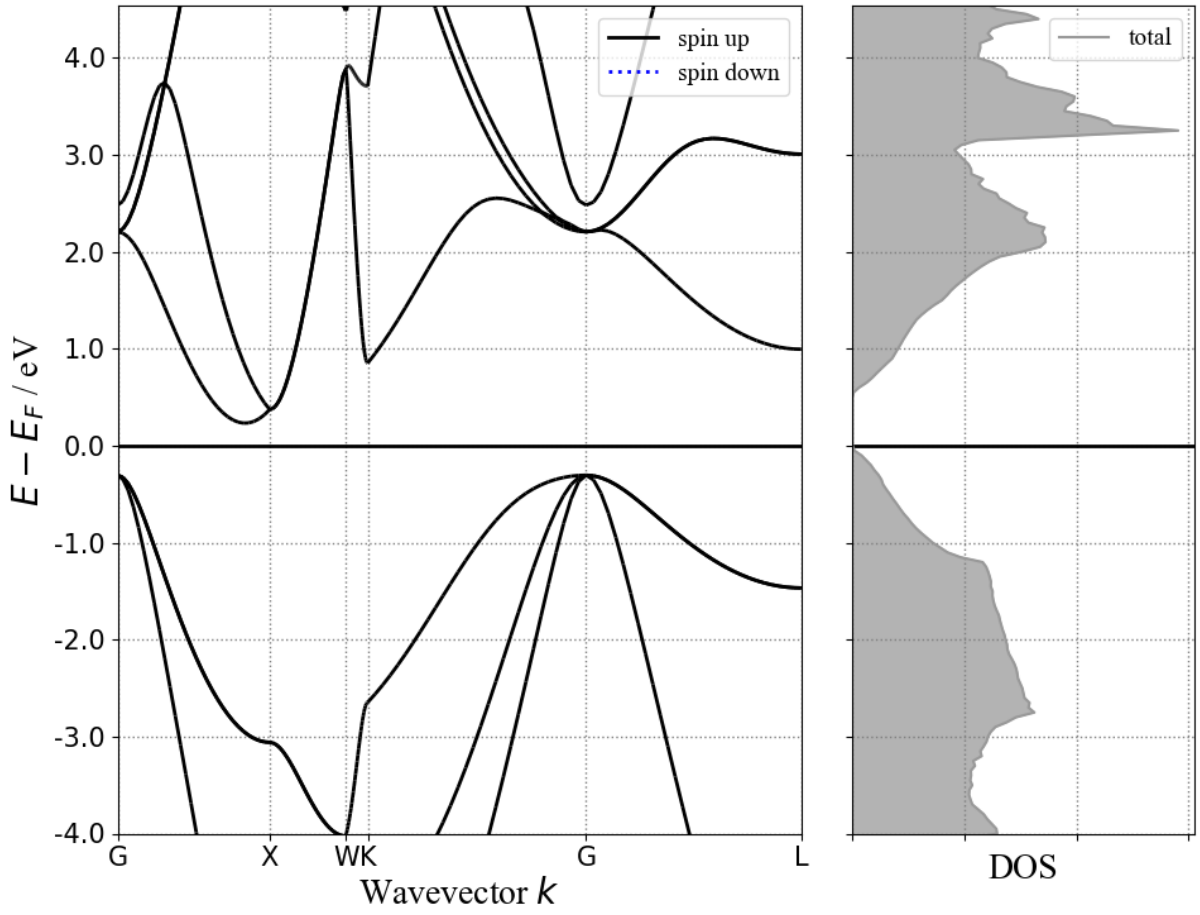
以 Si 的 slab 模型的 `potential.json` 为例：

（1）**势函数转化为 VESTA 能够支持的三维显示格式**：

创建 *trans_VESTA.py* 文件，具体代码如下：

```
import json
from dspawpy.io import write_VESTA_format

with open("./potential.json","r") as file:
      potential= json.load(file)

write_VESTA_format(potential["AtomInfo"], [potential["Potential"]["TotalElectrostaticPotential"], potential["Potential"]], "DS-PAW.vesta")
```

- 第 1，2 行为需要导入的 **python** 库函数；
- 第 4，5 行为需要读取的 **json** 文件，这里 `./potential.json` 指的时读取当前路径下的 `potential.json`，如果需要读取其他的 *json** 文件可以自行修改；
- 第 6 行为执行 *write_VESTA_format* 函数，将数据保存为 **VESTA** 可识别的格式，其中 `potential["AtomInfo"]` 为读取 `potential.json` 中的晶格及坐标信息，**potential["Potential"]["TotalElectrostaticPotential"]** 为 `potential.json` 中总的静电势的三维数据，**"DS-PAW- potential.vesta"** 表示将这些数据写入到 *DS-PAW- potential.vesta* 文件中；

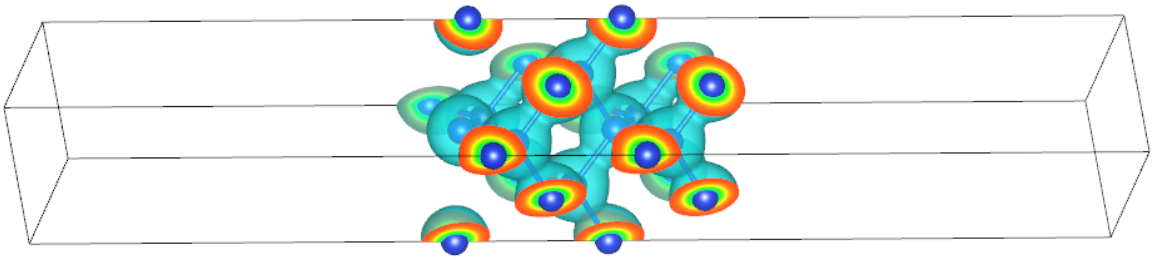

spin up
spin down
total
4.0
3.0
2.0
1.0
0.0
-1.0
-2.0
-3.0
-4.0
$E - E_F$ / eV
G
X
WK
G
L
Wavevector k
DOS

VESTA 展现的效果可以通过设置 VESTA 中 Isosurface level 数值来调节，具体步骤为 VESTA-properties-Isosurfaces-Isosurface level

知识点：

1. 使用 *with open* 语法加载需要处理的三维网格数据；
2. 使用 *write_VESTA_format* 函数可以将三维数据保存为 *VESTA* 格式的数据，其中 *potential["AtomInfo"]* 为晶格信息，*potential["Potential"]["TotalElectrostaticPotential"]* 总的静电势的数据，之后将数据命名为 *DS-PAW.vesta* ；
3. *write_VESTA_format* 支持输出多个网格数据，例如想要保存总的和自旋静电势，命令如下：*write_VESTA_format(potential["AtomInfo"], [potential["Potential"]["TotalElectrostaticPotential"], potential["Potential"]["SpinElectrostaticPotential"]], "DS-PAW.vesta")* ；
4. 处理部分电荷密度文件 *pcharge.json* 时，最后一行代码应可参考 *write_VESTA_format(pcharge["AtomInfo"], [pcharge["Pcharge"][0]["TotalCharge"]], "DS-PAW-pcharge.vesta")* 格式，其中 *[0]* 对应 *json* 文件中第 *1* 条待测能带电荷密度的三维数据，若需处理第 *2* 条能带该关键词应写作 *[1]*。

执行代码可以得到 VESTA 支持的数据，将数据拽入 VESTA 中得到以下势函数三维图：



（2）**将三维势函数进行面内平均：**

创建 *plot _planar_macroscopic.py* 文件，具体代码如下：

```
1  import json
2  import numpy as np
3  from dspawpy.plot import plot_potential_along_axis
4
5  with open("./potential.json","r") as file:
6      potential = json.load(file)
7
8  grid = potential["AtomInfo"]["Grid"]
9  pot= np.asarray(potential ["Potential"]["TotalElectrostaticPotential"]).reshape(grid,
   ↪order="F")
10
11 plt = plot_potential_along_axis(pot,axis=2,smooth=False)
12 plt.legend()
13 plt.show()
```

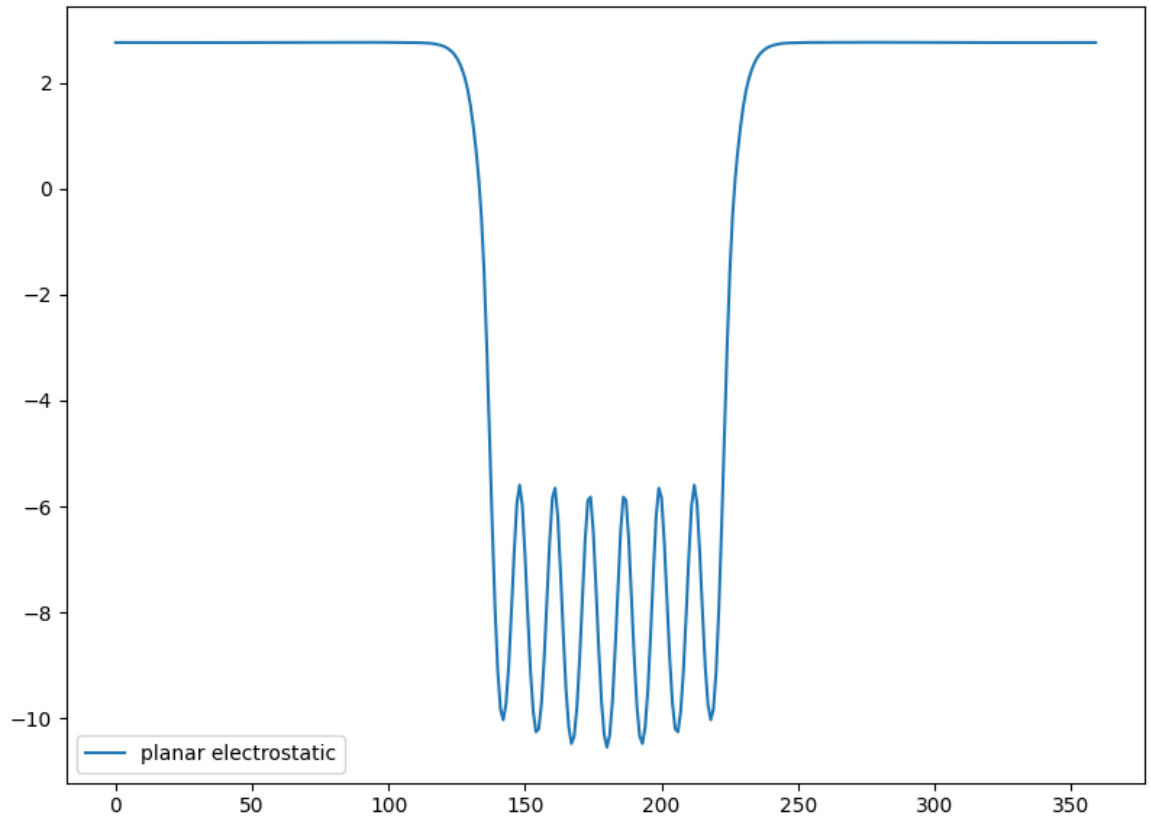
- 第 1,2,3 行为需要导入的 **python** 库函数，这里用到了 *dspawpy.io* 中的 `plot_potential_along_axis` 函数；
- 第 5，6 行为需要读取的 **json** 文件，这里 `./potential.json` 指的时读取当前路径下的 `potential.json` ，如果需要读取其他的 **json** 文件可以自行修改；

- 第 8 行为获取 grid，其中 **potential["AtomInfo"]["Grid"]** 为读取 `potential.json` 中的晶格及坐标信息;
- 第 9 行为执行 *pot* 函数，其中 **potential["Potential"]["TotalLocalPotential"]** 为读取 `potential.json` 中的局部势能信息,
- 第 11 行调用 *plot_potential_along_axis* 函数作图，**smooth=False** 则不作宏观平均图;
- 第 12 行为设置图例;
- 第 13 行为显示图片

知识点:

1. 进行面内平均及宏观平均时需要使用到 *numpy* 的库;
2. *DS-PAW* 软件中三维的数据都是按照一维的格式保存的，因此需要使用 *np.asarray* 函数将一维数据还原成 *3* 维数据;
3. 之后使用 *plot_potential_along_axis* 将某两个维度进行求平均，之后将剩下的维度数据进行绘图;
4. *plot_potential_along_axis* 的第一个变量为三维格点数据，第二个为沿着某个维度画图，将另外两个维度求平均，第三个参数为是否绘制宏观平均的曲线，第四个参数为宏观平均算法中的权重;

执行代码可以得到以下势函数图:



7.7 elf 电子局域密度数据处理

以快速入门 Si 电子局域密度计算计算得到的 `elf.json` 为例：

(1) 在计算目录中新建一个 *trans_elf.py* 文件，内容如下：

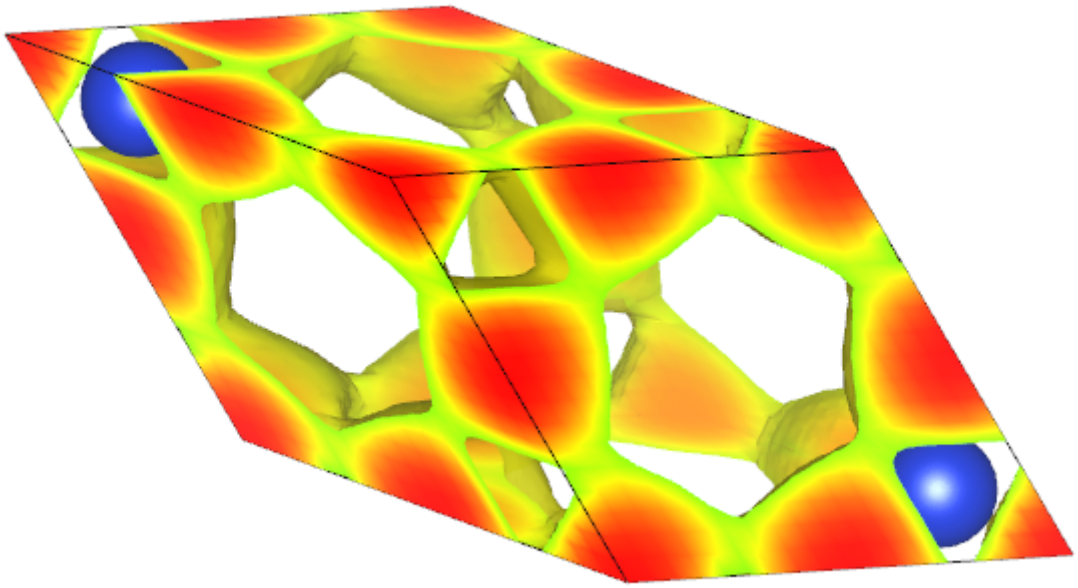
```
1  import json
2  from dspawpy.io import write_VESTA_format
3
4  with open("./elf.json","r") as file:
5        elf = json.load(file)
6
7  write_VESTA_format (elf ["AtomInfo"], [elf ["ELF"]["TotalELF"]], "DS-PAW-elf.vesta")
```

- 第 1，2 行为需要导入的 **python** 库函数
- 第 4，5 行为需要读取的 **json** 文件，这里 `./elf.json` 指的时读取当前路径下的 `elf.json` ，如果需要读取其他的 **json** 文件可以自行修改；
- 第 7 行为执行 *write_VESTA_format* 函数，将数据保存为 **VESTA** 可识别的格式，其中 **elf[”AtomInfo”]** 为读取 `elf.json` 中的晶格及坐标信息，**elf [”ELF”][”TotalELF”]** 为读取 `elf.json` 中总的局域电荷密度的三维数据, **DS-PAW- elf.vesta** 表示将这些数据写入到 **DS-PAW- elf.vesta** 文件中；

(2) 执行 *python trans_ elf.py* ；

(3) 得到转换后的文件 `DS-PAW-elf.vesta` ，将其重命名为 `elf.vasp` 以便在 **VESTA** 显示；

(4) 将 `elf.vasp` 文件拖入 **VESTA** 中则可得到如下的势能图，展现的效果可以通过设置 VESTA 中 *Isosurface level* 数值来调节，具体步骤为 **VESTA-properties-Isosurfaces-Isosurface level** 。



7.8 pcharge 部分电荷密度数据处理

以快速入门石墨烯部分电荷密度计算得到的 pcharge.json 为例：

(1) 在计算目录中新建一个 *trans_pcharge.py* 文件，内容如下：

```
import json
from dspawpy.io import write_VESTA_format

with open("./pcharge.json","r") as file:
      pcharge= json.load(file)

write_VESTA_format(pcharge["AtomInfo"], [pcharge["Pcharge"][0]["TotalCharge"]], "DS-PAW-pcharge.vesta")
```

- 第 1，2 行为需要导入的 **python** 库函数
- 第 4，5 行为需要读取的 **json** 文件，这里 ./pcharge.json 指的时读取当前路径下的 pcharge.json ，如果需要读取其他的 **json** 文件可以自行修改；
- 第 7 行为执行 *write_VESTA_format* 函数，将数据保存为 **VESTA** 可识别的格式，其中 **pcharge["AtomInfo"]** 为读取 pcharge.json 中的晶格及坐标信息，**pcharge["Pcharge"][0]["TotalCharge"]** 为 pcharge.json 中第 1 条待测能带电荷密度的三维数据，此例中为能带 4 的数据，若写作 **pcharge["Pcharge"][1]["TotalCharge"]** 则读取能带 5 的数据，**DS-PAW-pcharge.vesta** 表示将这些数据写入到 **DS-PAW-pcharge.vesta** 文件中；

(2) 执行 *python trans_pcharge.py* ；

(3) 得到转换后的文件 DS-PAW-pcharge.vesta ，将其重命名为 pcharge.vasp 以便在 **VESTA** 显示；

(4) 将 pcharge.vasp 文件拖入 **VESTA** 中则可得到如下的势能图，展现的效果可以通过设置 vesta 中 *Isosurface level* 数值来调节，具体步骤为 **vesta-properties-Isosurfaces-Isosurface level** 。

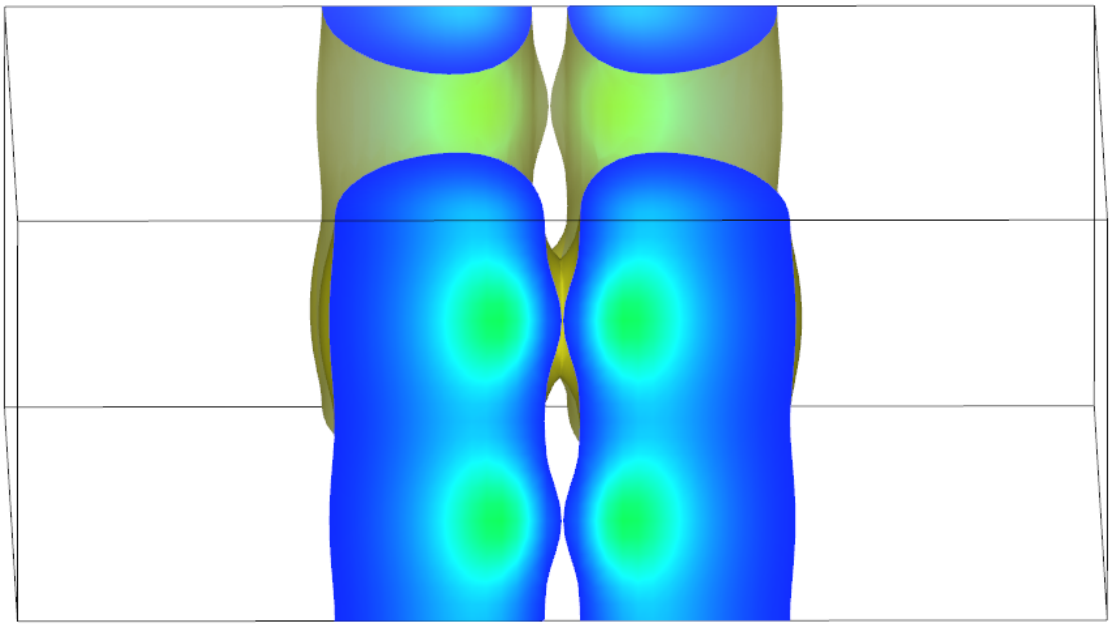
7.9 optical 光学性质数据处理

以快速入门 Si 体系光学性质计算得到的 optical.json 为例：

（1）**反射率数据处理**：

创建 *optical.py* 文件，具体代码如下：

```
import matplotlib.pyplot as plt
from dspawpy.plot import plot_optical

plot_optical("./optical.json","Reflectance",index=0)

plt.xlabel('Photon energy (eV)', fontsize=16)
plt.ylabel('Reflectance', fontsize=16)
plt.tick_params(labelsize=16)

plt.savefig("optical.png",img_format="png")
plt.show()
```

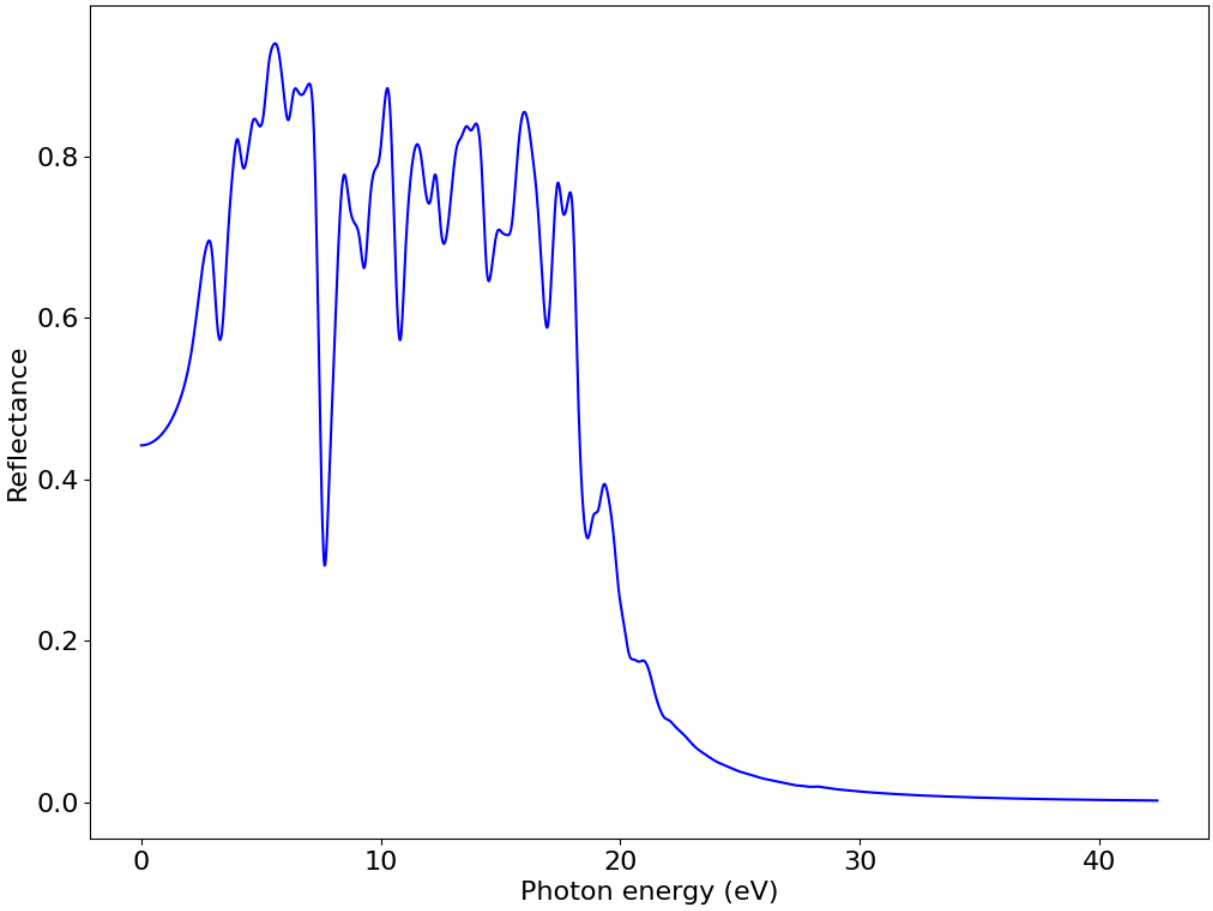


- 第 1，2 行为需要导入的 **python** 库函数；这里用到了 **matplotlib** 库，以及我们自己写的 **dspawpy.plot** 中的 **plot_optical** 模块
- 第 4 行调用 **plot_optical** 处理 `optical.json` 文件中 **Reflectance** 部分的数据，其中 **Reflectance** 为 `optical.json` 文件中的部分数据，这里用户可以根据需要自行修改，若需要处理其他光学性质的数据，可替换该参数为 **AbsorptionCoefficient** 或 **ExtinctionCoefficient** 或 **RefractiveIndex**
- 第 6-8 行设置 x 轴 y 轴的名称以及标签字体大小
- 第 9 行为保存图片的名称和格式，这里将数据保存为 png 格式命名为 **optical.png**
- 第 10 行为显示图片，如果用户运行的环境无法直接显示图片，用户可以将该行注释，之后打开 **optical.png** 文件查看。

知识点：

1. *Reflectance* 为光学性质中的一种，用户可以根据自己的需求将该关键词修改为 “*AbsorptionCoefficient*” 或” *ExtinctionCoefficient*” 或” *RefractiveIndex*”，分别对应吸收系数、消光系数和折射率；

执行代码可以得到以下反射率随能量变化的曲线：


Reflectance
0.0
0.2
0.4
0.6
0.8
0
10
20
30
40
Photon energy (eV)

7.10 neb 过渡态计算数据处理

以快速入门 Pt 体系的过渡态计算得到的 neb.json 为例：

（1）**输入文件之生成中间构型**：

创建 *neb_structure.py* 文件，具体代码如下：

```
from dspawpy.io.structure import from_dspaw_as
from dspawpy.diffusion.neb import NEB,write_neb_structures

#导入初态构型
init_struct = from_dspaw_as("./structure00.as")
#导入末态构型
final_struct = from_dspaw_as("./structure06.as")

#设置插点个数
neb = NEB(init_struct,final_struct,5)
structures = neb.idpp_interpolate()

#修改文件夹存储路径
write_neb_structures(structures,True,"as","./neb")
```

知识点：

1. 用户可以根据需要自行修改插点个数，设置为任意插值的序号即可

执行代码可以得到包含 7 个结构文件的文件夹，其中新生成的中间构型为 5 个。

（2）**过渡态计算数据处理**：

创建 *neb.py* 文件，具体代码如下：

```
import json
from pymatgen.core import Structure
from pymatgen.io.vasp import Poscar
from dspawpy.io.structure import to_file,from_hzw,from_dspaw_as,from_dspaw_atominfo
from dspawpy.diffusion.neb import NEB,write_neb_structures,plot_neb_barrier,plot_neb_
↪converge
import matplotlib.pyplot as plt

plot_neb_barrier("./neb.json")
plt.xlabel('Reaction Coordinate',fontsize=16)
plt.ylabel('Energy (eV)',fontsize=16)
plt.tick_params(labelsize = 16)
plt.savefig("neb_reaction_coordinate.png",img_format="png")
plt.show()

plot_neb_converge("./neb.json","03")
plt.savefig("neb_ionic_step.png",img_format="png")
plt.show()
```

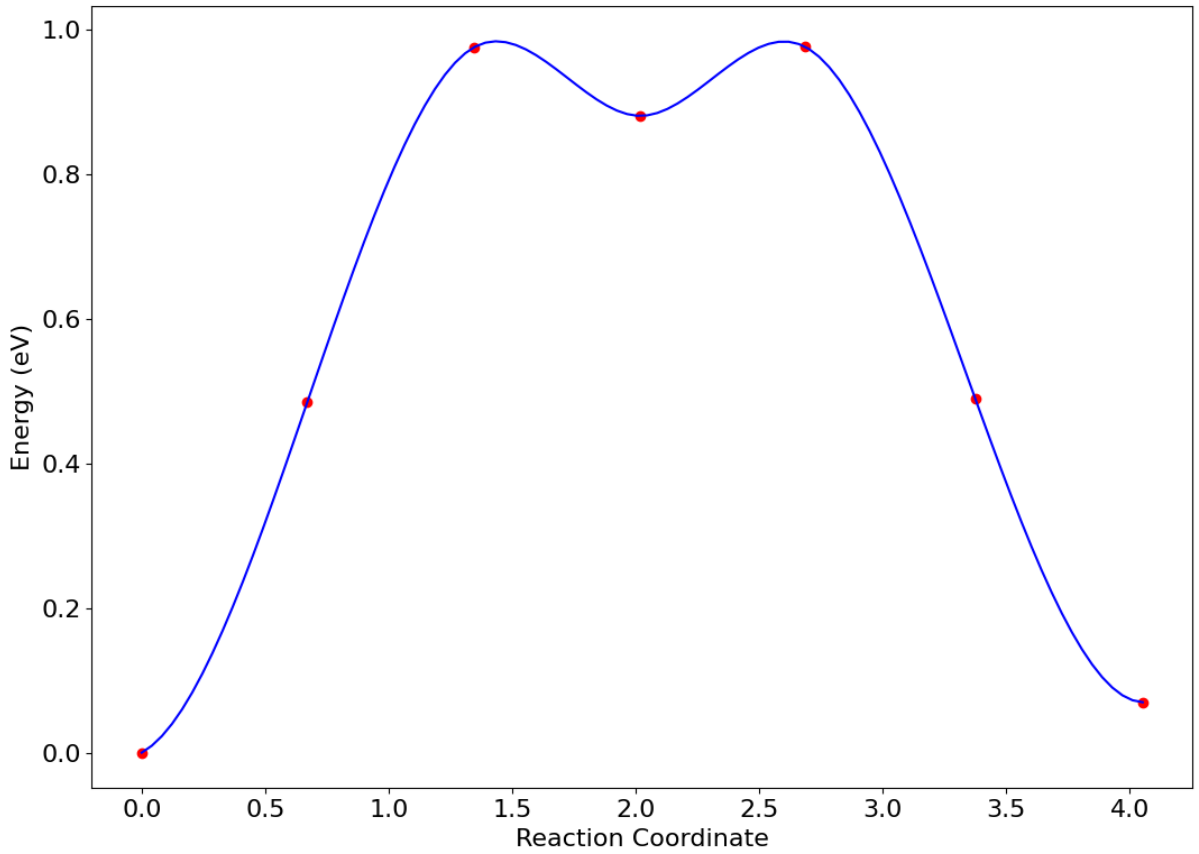
- 第 1-6 行为需要导入的 **python** 库函数
- 第 8 行为调用 **plot_neb_barrier** 函数处理 **json** 文件作反应势垒图，这里 **./neb.json** 指的时读取当前路径下的 **neb.json**

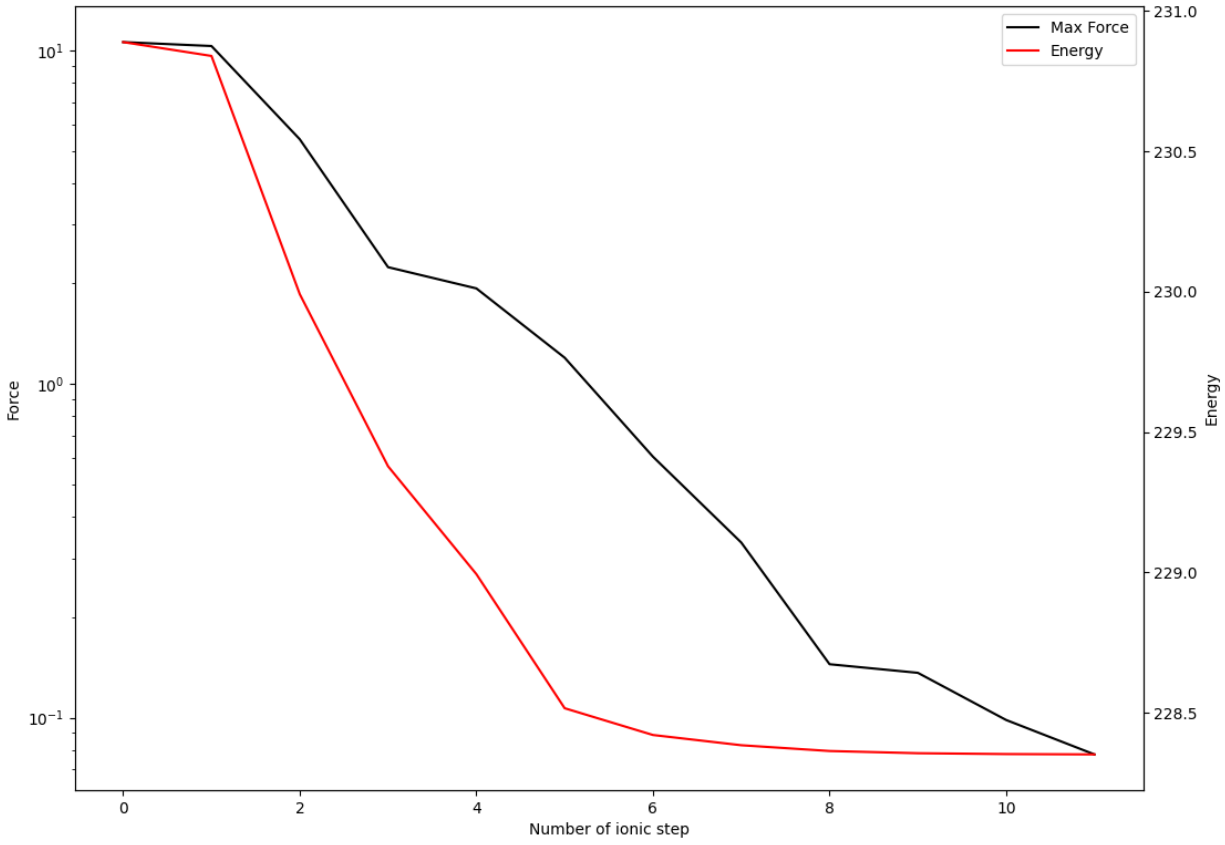
- 第 9-11 行设置 x 轴 y 轴的名称以及标签字体大小
- 第 12 行为保存图片的名称和格式，这里将数据保存为 png 格式命名为 **neb_reaction_coordinate.png**
- 第 13 行为显示图片，如果用户运行的环境无法直接显示图片，用户可以将该行注释，之后打开 **neb_reaction_coordinate.png** 文件查看
- 第 15 行为调用 **plot_neb_converge** 函数处理 **json** 文件作 **03** 构型的弛豫受力图
- 第 16 行为保存图片的名称和格式
- 第 17 行为显示图片

知识点：

1. 作能量和力在离子弛豫过程中的变化曲线时，*03* 为处理插入的第 *3* 个构型的数据，这里用户可以根据需要自行修改，设置为任意插值的序号即可；

执行代码可以得到以下能量随反应路径变化曲线：




Max Force
Energy
Force
Energy
Number of ionic step
10^1
10^0
10^{-1}
231.0
230.5
230.0
229.5
229.0
228.5
0
2
4
6
8
10

7.11 phonon 声子计算数据处理

以 Au 体系的声子能带态密度计算得到的 `phonon.json` 为例：

（1）**声子能带数据处理**：

创建 *phonon_bandplot.py* 文件，具体代码如下：

```
1  from pymatgen.phonon.plotter import PhononBSPlotter
2  from dspawpy.io import get_phonon_band_data
3
4  band_data = get_phonon_band_data("./phonon.json")
5
6  bsp =  PhononBSPlotter(band_data)
7  plt = bsp.get_plot()
8
9  plt.savefig("phonon_bandplot.png",img_format="png")
10 plt.show()
```

- 第 1、2 行为需要导入的 **python** 库函数，这里用到了 pymatgen 中的 **PhononBSPlotter** 库，以及 *dspawpy.io* 中的 `get_phonon_band_data` 函数；
- 第 4 行为读取 `phonon.json` 文件构造 **band_data** 这个结构体；
- 第 6 行为调用 **PhononBSPlotter** 处理 **band_data** 数据；
- 第 7 行为使用 `bsp` 下的 **get_plot** 进行画图；
- 第 9 行为保存图片的名称和格式，将数据保存为 png 格式命名为 **phonon_bandplot.png** ；
- 第 10 行为显示图片，如果用户运行的环境无法直接显示图片，用户可以将该行注释，之后打开 **phonon_bandplot.png** 文件查看。

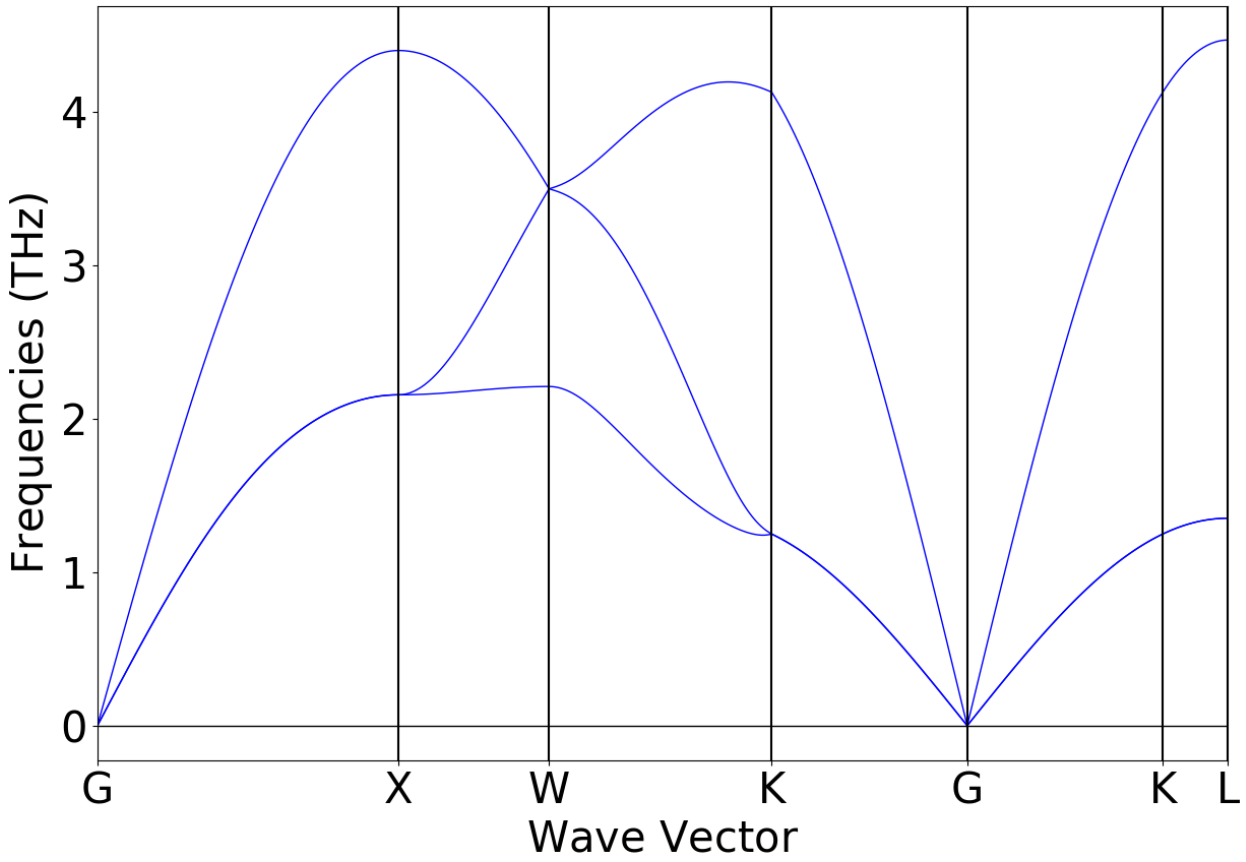
执行代码可以得到以下声子能带曲线：

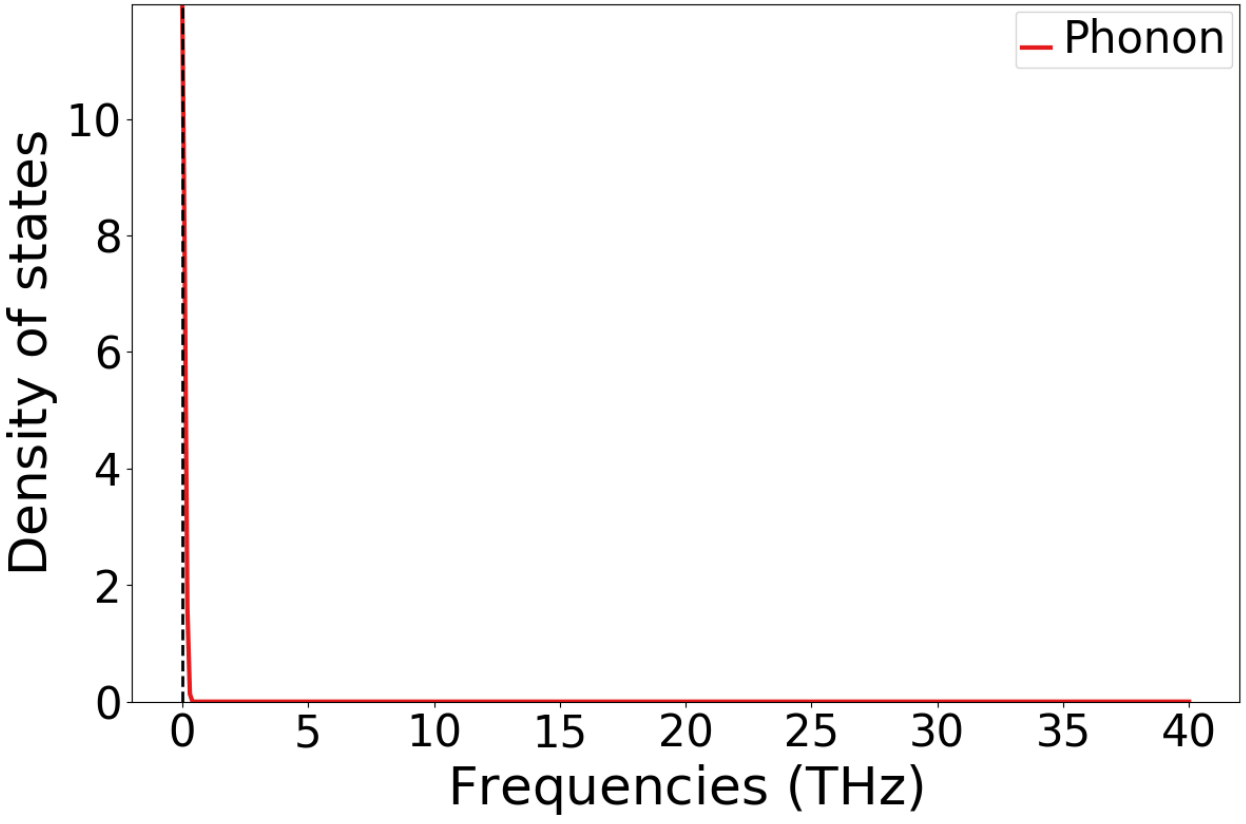
（2）**声子态密度数据处理**：

创建 *phonon_dosplot.py* 文件，具体代码如下：

```
1  from pymatgen.phonon.plotter import PhononDosPlotter
2  from dspawpy.io import get_phonon_dos_data
3
4  dos = get_phonon_dos_data("phonon.json")
5
6  dp = PhononDosPlotter()
7  dp.add_dos("Phonon",dos)
8  plt = dp.get_plot()
9
10 plt.savefig("phonon_dosplot.png",img_format="png")
11 plt.show()
```

执行代码可以得到以下声子态密度曲线：


Frequencies (THz)
0
1
2
3
4
G
X
W
K
G
K
L
Wave Vector


Phonon
Density of states
0
2
4
6
8
10
0
5
10
15
20
25
30
35
40
Frequencies (THz)

7.12 aimd 分子动力学模拟数据处理

以快速入门 H_2O 分子体系分子动力学模拟得到的 `aimd.json` 为例：

（1）**分子动力学模拟数据处理**：

创建 *aimd2pdb.py* 文件，具体代码如下：

```
1 import json
2 from dspawpy.io.structure import from_dspaw_atominfos,to_pdb
3
4 with open("./aimd.json","r") as file:
5     aimd = json.load(file)
6
7 structures = from_dspaw_atominfos(aimd["Structures"])
8 to_pdb(structures,"aimd.pdb")
```

- 第 1，2 行为需要导入的 **python** 库函数
- 第 4，5 行为需要读取的 **json** 文件，这里 **./aimd.json** 指的是读取当前路径下的 **aimd.json** ，如果需要读取其他的 **json** 文件可以自行修改
- 第 7 行为调用 *from_dspaw_atominfos* 处理 **structure** 的数据
- 第 8 行将处理的数据转为 **pdb** 格式

执行代码可以得到包含体系原子径向分布信息的文件 *aimd.pdb* ，得到通用轨迹 pdb 文件后，就可以利用 vmd 等分子动力学后处理软件进行分析。例如 VMD 中处理 RDFs 操作步骤：File - New Molecule - browse - open - Load - Extensions - Analysis - Radial Pair Distribution Function g(r) - Selection 1: element H , selction 2: element O - compute g(r)

CHAPTER 8

常见问题

8.1 License 常见错误信息

- 错误信息：`Error code: -10, Get License File Error`
- 错误详情：没有找到 *license* 文件或者没有权限打开

- 错误信息：`Error code: -20, Get License Product Error`
- 错误详情：获取产品信息出错

- 错误信息：`Error code: -30, Check Local Environment Error`
- 错误详情：本地硬件信息验证出错

- 错误信息：`Error code: -40, Check Install Path Error`
- 错误详情：本地安装路径验证出错

- 错误信息：`Error code: -50, Check Validate White User Error`
- 错误详情：白名单验证出错，当前用户不在白名单内

- 错误信息：`Error code: -60, Check Device Studio license Error`
- 错误详情：错误的 *DS* 产品信息

- 错误信息：`Error code: -70, Check Device Studio license Error`

- 错误详情：*DS* 可使用产品目录中没有 *DS-PAW* 软件

- 错误信息：`Error code: -80, Check Device Studio license Error`
- 错误详情：*DS license* 中 *DS-PAW* 当前版本高于注册版本

- 错误信息：`Error code: -90, Check Device Studio license Error`
- 错误详情：*DS license* 中 *DS-PAW* 超出有效期. 注册有效期

8.2 inputcheck 检查输入文件常见错误信息

- 错误信息：`Parameters task error`
- 错误详情：*task* 参数名或参数设置错误

- 错误信息：`Parameters Check error`
- 错误详情：参数名错误

- 错误信息：`Parameters type error`
- 错误详情：参数类型设置错误

- 错误信息：`Parameters data error`
- 错误详情：参数可选值设置问题

- 错误信息：`Parameters size error`
- 错误详情：参数的维度大小设置问题

- 错误信息：`Parameters range error`
- 错误详情：参数范围设置问题

- 错误信息：`Structure key error`
- 错误详情：结构文件中关键字缺失

- 错误信息：`Structure type error`
- 错误详情：结构文件中关键词设置错误

- 错误信息：`Structure size error`
- 错误详情：结构文件中数据的大小错误

8.3 error 计算过程中常见错误信息

- 错误信息：E1015/E1011/E1012/E1014/E1005
- 错误详情：*K* 点读取发生错误
- 解决方案：在各个方向增加 *K* 点密度（可以尝试增加 *20%* 左右，真空方向对应的 *K* 点不用增加）或修改 *cal.smearing* 和 *cal.sigma*，如设置 *cal.smearing = 1, cal.sigma = 0.05*

- 错误信息：E1188
- 错误详情：使用四面体方法时 *K* 点必须大于 *4* 个
- 解决方案：在各个方向增加 *K* 点密度（可以尝试增加 *20%* 左右，真空方向对应的 *K* 点不用增加）或修改 *cal.smearing* 和 *cal.sigma*，如设置 *cal.smearing = 1, cal.sigma = 0.05*

- 错误信息：E1005
- 错误详情：*k* 点 *shift* 读取错误
- 解决方案：尝试使用 *cal.ksampling= G*

- 错误信息：E1013
- 错误详情：*K* 点路径读取错误
- 解决方案：尝试使用 *cal.ksampling= G*

- 错误信息：E1022
- 错误详情：读取 *wave.bin* 中的本征值发生错误
- 解决方案：调整两次计算的输入参数，获取正确的 *wave.bin*

- 错误信息：E1024
- 错误详情：当前计算生成的网格大小与 *rho.bin* 中读到的不一致
- 解决方案：调整两次计算的输入参数，获取正确的 *rho.bin*

- 错误信息：E1042/E1041
- 错误详情：*ZBRENT* 算法在搜寻根函数时发生错误
- 解决方案：从日志文件中读取报错前一步的结构，生成新的结构文件，之后提高收敛精度 *scf.convergence* 继续计算；或修改弛豫算法为 *relax.methods = QN* 重新计算

- 错误信息：E1063
- 错误详情：在使用 *davidson block* 方法时，执行 *LAPACKE_zhegv_work* 函数发生错误
- 解决方案：调整 *cal.methods*

- 错误信息：E1064
- 错误详情：在对角化时，执行 *LAPACKE_zhegv_work* 函数发生错误
- 解决方案：调整 *cal.methods*

- 错误信息：E1073
- 错误详情：并行加速运算发生错误
- 解决方案：在提交脚本中关闭*-pob* 命令重新提交任务

- 错误信息：E1115
- 错误详情：晶格体积为 *0*

- 错误信息：E1186
- 错误详情：在求旋转矩阵的逆时发生错误
- 解决方案：关闭对称性 *sys.symmetry = false*

- 错误信息：E1187
- 错误详情：在求旋转矩阵的逆时发生错误
- 解决方案：尝试使用 *cal.ksampling= MP*

- 错误信息：E1226
- 错误详情：扩胞过程中发生错误
- 解决方案：检查并修改结构文件

- 错误信息：E1248
- 错误详情：在正交化波函数时，*LAPACKE_zpotrf_work* 函数发生错误
- 解决方案：将 *sys.symmetry = false* 同时减小 *relax.stepRange*

- 错误信息：E1249
- 错误详情：在正交化波函数时，*LAPACKE _ztrtri _work* 函数发生错误

- 错误信息：E2024/E2025
- 错误详情：在求旋转矩阵的逆时发生错误
- 解决方案：提高对称性判断的精度，如设置 *sys.symmetryAccuracy = 1.0e-6*

- 错误信息：`E3058`
- 错误详情：赝势读取错误
- 解决方案：目前 *DS-PAW* 提供 *72* 种元素的赝势，暂不支持赝势库以外的元素进行计算

- 错误信息：`Failed to converge the scf calculation`
- 错误信息：电子步在设置步数内未收敛
- 解决方案：可尝试修改算法为 *cal.methods = 1*, 也可尝试加大 *cal.totalBands*

CHAPTER 9

发布说明

9.1 version 版本简介

目前 DS-PAW2022A 软件已上线鸿之微云平台，版本支持结构弛豫（包括固定基矢）、电子结构、力学性质、磁性计算、过渡态（包括 SSNEB）、杂化泛函、弹性和频率、分子动力学、强关联计算、范德瓦尔斯修正（包括 vdW-DF）、声子计算（包括 nac）、加电场、铁电极化 (现代极化理论)、能带去折叠、热力学性质、介电张量、压电张量、波恩有效电荷、Bader 电荷计算等 41 种功能。本期主要更新功能序号为 **29-41** 。

9.2 function 功能列表

(1) `Plane wave basis`

- 设计功能：使用平面波的基组展开波函数

(2) `Projector Augmented Wave pseudopotential(PAW)`

- 设计功能：使用投影缀加平面波赝势

(3) `Relaxation`

- 设计功能：结构弛豫计算，支持弛豫原子位置、晶格弛豫、晶格和原子位置弛豫；同时支持固定原子弛豫

(4) `Self-consistent`

- 设计功能：自洽场计算

---

(5) Spin

- 设计功能：支持无自旋、共线自旋、非共线自旋及自旋轨道耦合体系

---

(6) Functional

- 设计功能：支持 LDA、GGA(PBE)

---

(7) Total energy

- 设计功能：系统总能量计算

---

(8) Force

- 设计功能：原子受力计算

---

(9) Stress

- 设计功能：应力计算

---

(10) Band structure

- 设计功能：能带计算

---

(11) Projected band structure

- 设计功能：投影能带计算

---

(12) Density of states

- 设计功能：电子态密度计算

---

(13) Projected density of states

- 设计功能：电子投影态密度计算

---

(14) Electron local function

- 设计功能：电子局域函数计算

---

(15) Potential

- 设计功能：势函数计算，支持静电势函数、局域势函数计算

---

(16) `VDW correction`

- 设计功能：支持 DFT-D2、DFT-D3 范德瓦尔斯修正

---

(17) `Hybrid DFT`

- 设计功能：支持 PBE0、HSE03、HSE06 杂化泛函修正

---

(18) `DFT+U`

- 设计功能：引入 Hubbard U 处理强关联体系

---

(19) `Optical properties`

- 设计功能：计算介电常数函数、折射率、反射率、吸收系数、消光系数等

---

(20) `Background charge`

- 设计功能：引入背景电荷来计算带电体系

---

(21) `Dipol correction`

- 设计功能：引入偶极修正来处理真空静电势不平整的问题

---

(22) `Partial charge`

- 设计功能：计算某个指定 K 点和指定能带的电荷密度

---

(23) `NEB`

- 设计功能：使用 CI-NEB 的方法搜索过渡态结构

---

(24) `Frequency`

- 设计功能：进行频率计算

---

(25) `Phonon(FD)`

- 设计功能：使用有限位移的方法计算声子能带、声子态密度

---

(26) `Phonon(DFPT)`

- 设计功能：使用线性响应的方法计算声子能带、声子态密度

---

(27) `Elastic`

- 设计功能：进行弹性常数、弹性模量、剪切模量、杨氏模量、泊松比等计算

(28) AIMD

- 设计功能：第一性原理分子动力学计算

(29) revPBE/PBEsol/RPBE functionals

- 设计功能：支持 revPBE/PBEsol/RPBE 交换关联泛函

(30) Nonlocal vdW-DF functionals

- 设计功能：支持 vdw-optPBE/vdw-optB88/vdw-optB86b/vdw-DF/vdw-revPBE/vdw-DF2/vdw-revDF2 范德瓦尔斯泛函

(31) Electric field

- 设计功能：通过引入 dipole 的方式实现对非周期性方向加电场

(32) Solid state NEB

- 设计功能：支持体系晶格可变的 NEB 计算

(33) Ferroelectric polarization

- 设计功能：使用现代极化理论计算铁电极化值

(34) Band unfolding

- 设计功能：将超胞的波函数折叠入原胞的布里渊区中，实现能带去折叠功能

(35) Thermal properties

- 设计功能：通过力常数矩阵计算亥姆霍兹自由能/固定体积热容/熵

(36) Band structure with non-analytical correction

- 设计功能：声子能带计算中考虑长程库仑相互作用

(37) Dielectric tensor

- 设计功能：使用线性响应的方法计算介电张量

(38) Piezoelectric tensor

- 设计功能：使用线性响应的方法计算压电张量

(39) `Born effective charge`

- 设计功能：使用线性响应的方法计算波恩有效电荷

---

(40) `Bader charge`

- 设计功能：使用 Bader 电荷划分方法计算体系中每个原子的价电子数

---

(41) `Constraint relaxation`

- 设计功能：使用约束某个维度的晶格来实现晶格的约束弛豫，常用于二维材料/正交晶系等
