DS-PAW 手册

2023 鸿之微科技（上海）股份有限公司

2026-01-29

目录:


交换关联泛函&修正
• LDAs, GGAs, 杂化泛函
• DFT+U修正
• vdW-DF 泛函/DFT-DF修正
电子结构
• 态密度
• 电子局域密度
• 部分电荷（pcharge）
• Bader电荷分析
• 能带
• 能带反折叠
• wannier basis
结构演变
• 共轭梯度法
• 拟牛顿法
• NEB过渡态搜索
• BO第一性分子动力学模拟
磁学性质
• 线性自旋
• 非线性自旋
• 自旋轨道耦合
响应理论
• 声子性质
• 弹性常数
• 介电性质
• Born有效电荷
• 压电张量
光学性质
• 介电函数对频率的响应
更多功能
• 外加电场
• 隐式溶剂化模型
• 恒电势偏压

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 command 命令说明

1.1.1 list 命令列表

- *-lic*
- *-info*
- *-example*
- *-ipp*
- *-mpi*
- *-mpiargs*
- *-pob*

1.1.2 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: *-example*

使用方法: *-example* 用于快速执行一次计算，可检查 DS-PAW 是否正确安装，执行命令：*DS-PAW -example*

命令名称: *-ipp*

使用方法: *-ipp* 用于查看 DS-PAW 赝势头数据信息，包括截断能、价电子数等。执行命令：*DS-PAW -ipp*

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。DS-PAW 在部分功能计中无法开启 pob，此时会给出 warning 并将 pob 关闭

1.2 run 程序运行

1.2.1 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
```

1.2.2 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。

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、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、溶剂化能计算、固定电势计算、wannier 插值能带计算**；DS-PAW 软件的参数大致可分为以下几类：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，基本参数多有默认值。本章筛选部分参数进行介绍，参数列表及详情另见参数说明 部分。

2.1 relax 结构弛豫计算

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原子受力会比较大，且即便是其他 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，因此在计算某个结构的具体性质之前需进行结构弛豫计算。

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

(续下页)

(接上页)

```
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.convergenceType = force
relax.convergence =  0.05
relax.methods = CG

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，由 task 参数控制：

- `task`： 设置计算类型，本次计算为 relax 即结构弛豫；

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 *.h5* 的结构文件格式（早期 json 文件可支持但不建议用户使用，后续 DS-PAW 发布版本将会完全摒弃 json 格式的输出），*.as* 文件可使用 Device Studio 软件直接生成，也可手动构造；
- `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** 、**PBE** 及多种修正泛函；
- `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；

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 表示只弛豫原子位置；另有可选值 volume 表示只弛豫晶格体积；all 表示弛豫弛豫原子位置、晶胞形状和体积；
- `relax.convergenceType`：设置结构弛豫收敛的判据类型，force 表示以原子受力作为判据，另有可选值 energy；
- `relax.convergence`： 设置结构弛豫时，原子受力的收敛精度大小；
- `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

为展示结构弛豫前后结构的变化，此例手动将第一个 Si 原子在 x 方向的坐标从 **-0.125** 改为 **-0.115** 。

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；

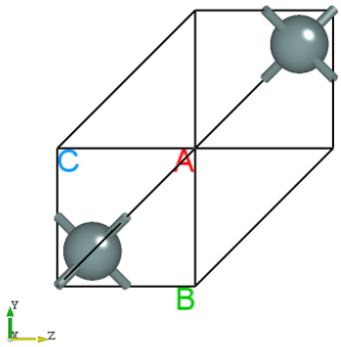
在无图形界面的 linux 环境下运行软件的方式与在 windows 的环境下运行程序会有很大的区别，在 linux 下需要通过命令行的方式来运行程序；一般情况下需要先加载一下环境变量，通常会将需要用的环境变量写入文本或者 **~/.bashrc** ，通过 **source** 的命令来加载环境；环境加载完成之后，运行 *DS-PAW relax.in* ，此时使用单机版的 DS-PAW 进行计算；如需并行计算则需运行 *DS-PAW -mpi mpirun -mpiargs* “*-n 2*” *relax.in* ，**-mpi** 指定 mpirun 的名称，**-mpiargs** 指定 mpirun 后面的参数。命令介绍见第一节软件简介 部分。使用排队系统（例如 PBS、slurm 等）提交任务需先配置相应的 `.pbs` 或 `.slurm` 脚本，使用 *qsub DS-PAW.pbs* 或 *sbatch DS-PAW.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.h5* 、*latestStructure.as* 等输出文件:

- *DS-PAW.log*： DS-PAW 计算之后得到的日志文件；
- *relax.h5*： 弛豫计算对应的 h5 输出文件，结构解析见输出文件格式说明 部分，DS-PAW 可读取该 h5 文件进行续算；
- *latestStructure.as*： 弛豫终点的 as 结构文件，可直接查看数据；

将 *latestStructure.as* 拖入 Device Studio 查看结构如下所示：



查看 *latestStructure.as* 文件，可得弛豫结束后的晶胞信息如下：

```
Total number of atoms
2
Lattice
0.0000000000000000    2.7500000000000000    2.7500000000000000
2.7500000000000000    0.0000000000000000    2.7500000000000000
2.7500000000000000    2.7500000000000000    0.0000000000000000
Direct
Si   0.8801735223171917  0.8748246492235915  0.8748246492235915
Si   0.1298264776828063  0.1251753507764085  0.1251753507764085
```

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，可以设置 relax.covergenceType = energy。**

2.2 scf 自洽计算

自洽计算能得到特定晶体的电荷密度和波函数文件，电荷密度文件用于后续计算该体系的能带、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 = true
sys.functional = PBE
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 输入自洽相关参数介绍：

- `task`： 设置计算类型，此次计算为 **scf** 自洽计算；
- `cal.cutoffFactor`：设置 `cal.cutoff` 的系数，计算采用的截断能大小等于 `cal.cutoff * cal.cutoffFactor`；
- `io.charge`： 控制电荷密度文件输出的开关；
- `io.wave`： 控制波函数文件输出的开关；

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
```

常规自洽计算会选取结构弛豫得到的稳态构型作为结构输入；

备注

1. **在结构弛豫和自洽计算中可保存 elf、potential 的数据，只需要将 io.elf 和 io.potential 设置为 true 即可；**
2. **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

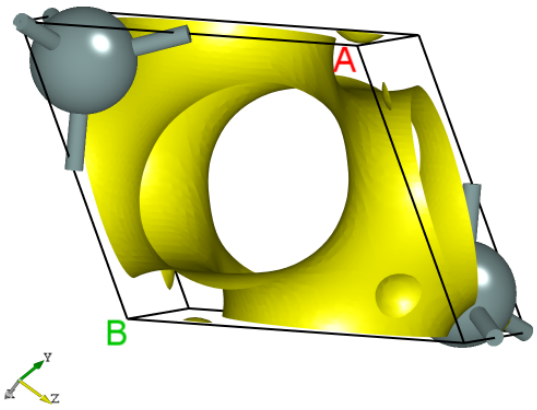
准备好输入文件 *scf.in* 和 *structure.as* 后，将文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 *DS-PAW scf.in* 。

2.2.3 analysis 计算结果分析

根据上述的输入文件，计算完成之后可得到 *DS-PAW.log* 、*scf.h5* 、*rho.bin* 、*wave.bin* 、*rho.h5* 等输出文件。

- *DS-PAW.log*： DS-PAW 计算之后得到的日志文件，记录自洽计算中能量迭代等主要信息；
- *scf.h5*： 自洽计算对应的 h5 输出文件，结构解析见输出文件格式说明 部分；
- *rho.bin*： 电荷密度的二进制文件，用于后续的后处理计算；
- *rho.h5*： 电荷密度的 **h5** 格式文件，可简单转换成 VESTA 能读取的格式（见辅助工具使用教程），从而将电荷密度信息可视化；
- *wave.bin*： 波函数的二进制文件，用于后续计算；

可通过 python 脚本将 *rho.h5* 文件转成 *VESTA* 软件支持的格式，具体操作见辅助工具使用教程 部分。处理可得一维、二维、三维电荷密度图，其中三维图效果应如下所示：



2.3 band 能带计算

普通能带计算有两种方式可完成，task=band 的两步法及 task=scf 的一步法。此节以 Si 体系为例，介绍两种方法下对应的参数设置。

2.3.1 Si 体系能带计算输入文件

2.3.1.1 task = band 两步算

输入文件包含参数文件 *scf.in* 和 *band.in* ，结构文件 *structure.as* ，*scf.in* 设置与上节自洽计算一致，*band.in* 参数如下：

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

cal.iniCharge = ./rho.bin
cal.methods = 2
cal.smearing = 1
cal.cutoffFactor = 1.5
cal.totalBands = 12

#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`： 设置计算类型，本次计算为 band 能带计算；
- `cal.iniCharge`： 设置电荷密度文件的路径，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`：设置能带计算时高对称点标签，一个`band.kpointsCoord`对应一个`band.kpointsLabel`；
- `band.kpointsCoord`： 设置能带计算时高对称点的分数坐标，每三个数为一组；
- `band.kpointsNumber`： 设置每相邻两个高对称点间的 k 点个数，有两种写法：
  - 当设置参数为 band.kpointsNumber= [30, 30, 30, 30, 30] 时，所有高对称点之间撒点数均为 30；
  - 当设置参数为 band.kpointsNumber= [30] 时，高对称点 G 与 X 之间撒点数为 30，以此求得撒点密度；对高对称点 X 与 W、W 与 K、K 与 G、G 与 L 之间进行等密度撒点，实际撒点数可从 DS-PAW.log 的参数打印部分获取；
- `band.EfShift`： 决定是否读取 rho.bin 中的 EFermi 作为 band 计算输出中的 EFermi。默认为 true，表示从 rho.bin 读取 EFermi。

structure.as 文件同自洽计算。（见 2.2 节）

备注

1. **两步算时，scf.in 和 band.in 中参数 cal.cutoffFactor 及 cal.cutoff 必须保持一致，否则会出现格点数据不匹配的问题。**
2. **cal.iniCharge 指定自洽计算生成的电荷密度文件 rho.bin 所在路径。**

2.3.1.2 task = scf 一步算

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

```
# task type
task = scf
#system related
sys.structure = structure.as
sys.symmetry = true
sys.functional = PBE
sys.spin = none
#scf related
cal.methods = 2
cal.totalBands = 12
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]
```

备注

1. 能带一步算对应结果文件为 scf.h5 ， 此时能带数据存储在 scf.h5 文件中， 可直接调用辅助工具使用教程 的 bandplot.py 脚本处理 scf.h5 文件。
2. **io.band=true 只在 task=scf 时生效。**
3. **当 io.band 生效时，不再需要设置 cal.iniCharge = ./rho.bin，对 K 空间高对称点的求解将在 scf 计算时同步完成。**
4. **scf.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。**

2.3.2 run 程序运行

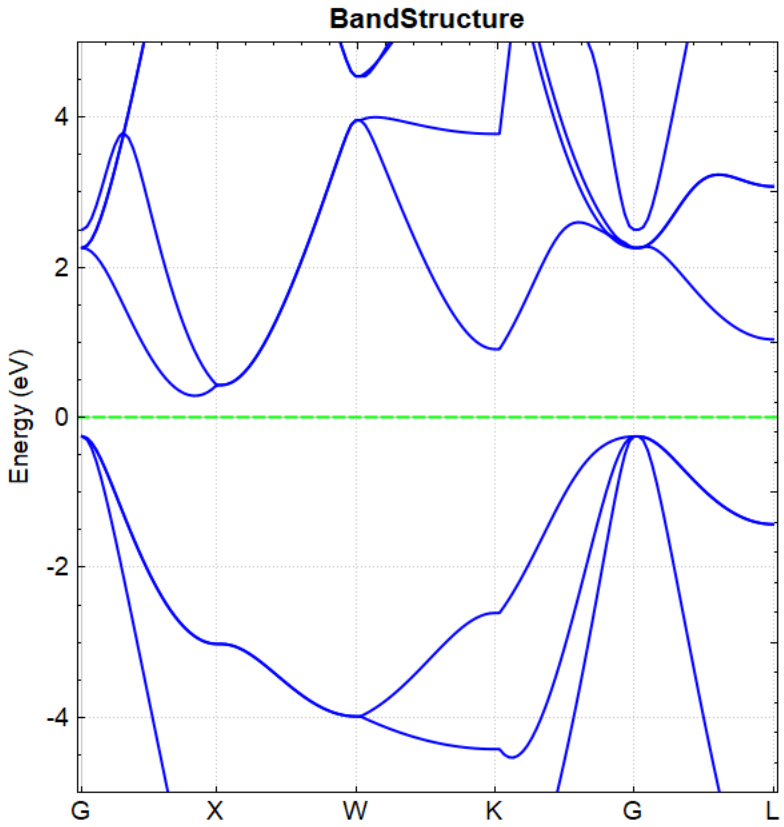
以两步算为例，将参数控制文件 *scf.in* 、*band.in* 和结构文件 *structure.as* 上传到服务器，按照结构弛豫中介绍的方法依次执行 *DS-PAW scf.in* 、*DS-PAW band.in* 。

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* 、*scf.h5* 、*band.h5* 等输出文件。

- *DS-PAW.log*： DS-PAW 能带计算之后得到的日志文件，可直接读取带隙、VBM、CBM 等重要信息；
- *band.h5*： 能带计算对应的 **h5** 输出文件；保存能量本征值等重要数据，具体的数据结构详见输出文件格式说明 部分；

可使用 **python** 对 *band.h5* 进行数据处理，具体操作见辅助工具使用教程 部分。处理得到的能带图效果应如下所示：



备注

1. **能带计算一步法和两步法得到的能带图效果一致。**

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 = true
sys.functional = PBE
sys.spin = none

cal.iniCharge = ./rho.bin
cal.methods = 2
cal.smearing = 1
cal.cutoffFactor = 1.5
cal.totalBands = 12

#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：

投影能带计算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` ： 控制能带计算中投影计算的开关；

2.4.2 run 程序运行

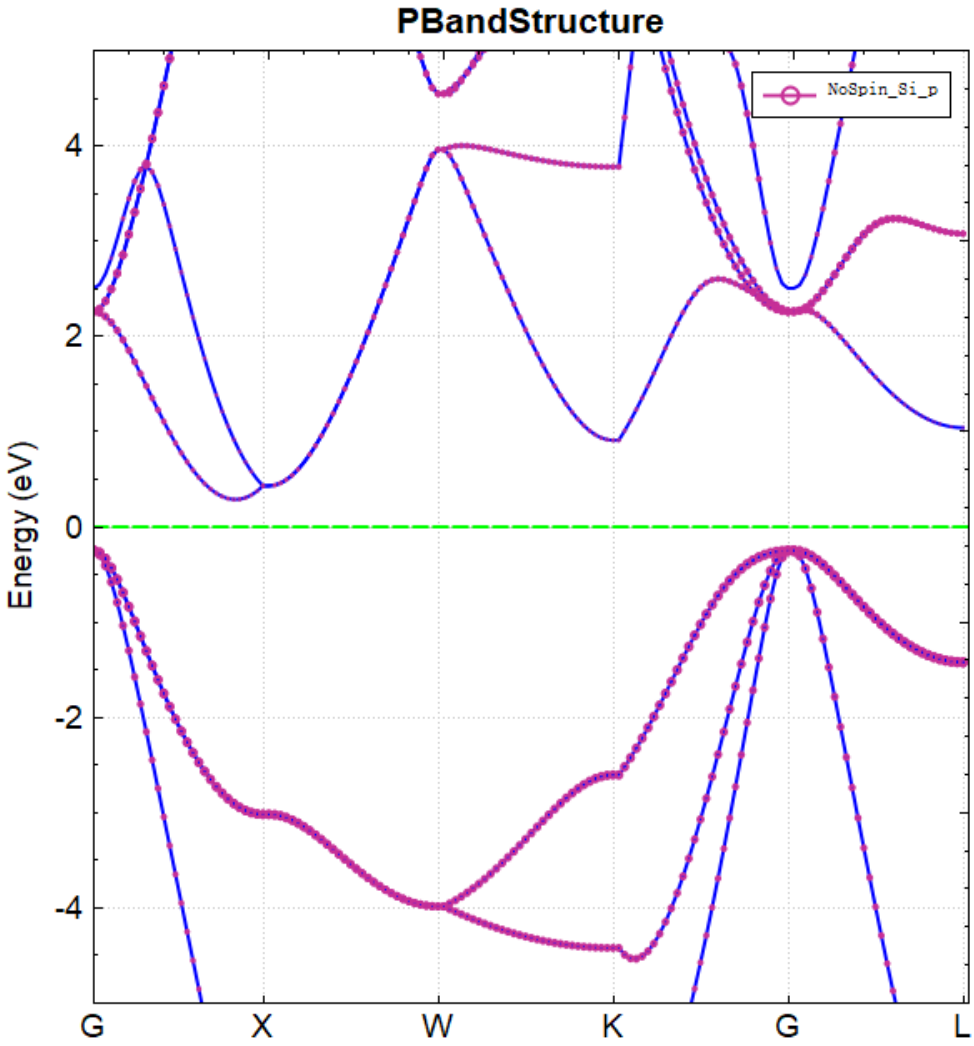
准备好输入文件 *pw_band.in* 和 *structure.as* 以及 *rho.bin* 之后，将文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 *DS-PAW pw_band.in* 。

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.h5* 等输出文件。

- *DS-PAW.log*： DS-PAW 能带计算之后得到的日志文件；
- *band.h5*： 能带计算对应的 **h5** 输出文件，此时投影能带的数据也将会被保存在 *band.h5* 中，具体的数据结构详见输出文件格式说明 部分；

可使用 **python** 对 *band.h5* 进行数据处理，具体操作见辅助工具使用教程 部分。处理得到的能带图效果应如下所示：



2.5 dos 态密度计算

态密度计算有两种方式可完成，task=dos 的两步法及 task=scf 的一步法。此节以 Si 体系为例，介绍两种方法下对应的参数设置。

2.5.1 Si 体系态密度计算输入文件

2.5.1.1 task = dos 两步算

输入文件包含参数文件 *scf.in* 、*dos.in* 及结构文件 *structure.as* ，*scf.in* 设置与自洽计算一致，*dos.in* 参数如下：

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

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

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

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`： 设置计算类型，本次计算为 dos 态密度计算；
- `cal.iniCharge`： 设置电荷密度的读取路径，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`： 设置 k 点网格密度，态密度计算时 k 点建议增大到自洽计算的两倍左右；

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`： 设置态密度计算时能量的区间；
- `dos.resolution`： 设置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；

structure.as 文件同自洽计算。（见 2.2 节）

备注

1. **两步算时，scf.in 和 dos.in 中参数 cal.cutoffFactor 及 cal.cutoff 必须保持一致，否则会出现格点数据不匹配的问题。**

2.5.1.2 task = scf 一步算

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

```
# task type
task = scf
#system related
sys.structure = structure.as
sys.symmetry = true
sys.functional = PBE
sys.spin = none
#scf related
cal.methods = 2
cal.smearing = 4
cal.ksamping = G
cal.kpoints = [10, 10, 10]
cal.cutoffFactor = 1.5
#outputs
io.charge = true
io.wave = true
#dos related
io.dos=true
dos.range=[-10, 10]
dos.resolution=0.05
```

备注

1. 态密度一步算对应结果文件为 scf.h5 ， 此时态密度数据存储在 scf.h5 文件中，可直接调用辅助工具使用教程 的 dosplot.py 脚本处理 scf.h5 文件。
2. **io.dos=true 只在 task=scf 时生效**。
3. **当 io.dos 生效时，不再需要设置 cal.iniCharge = ./rho.bin，此时通过自洽计算获得 DOS**。

2.5.2 run 程序运行

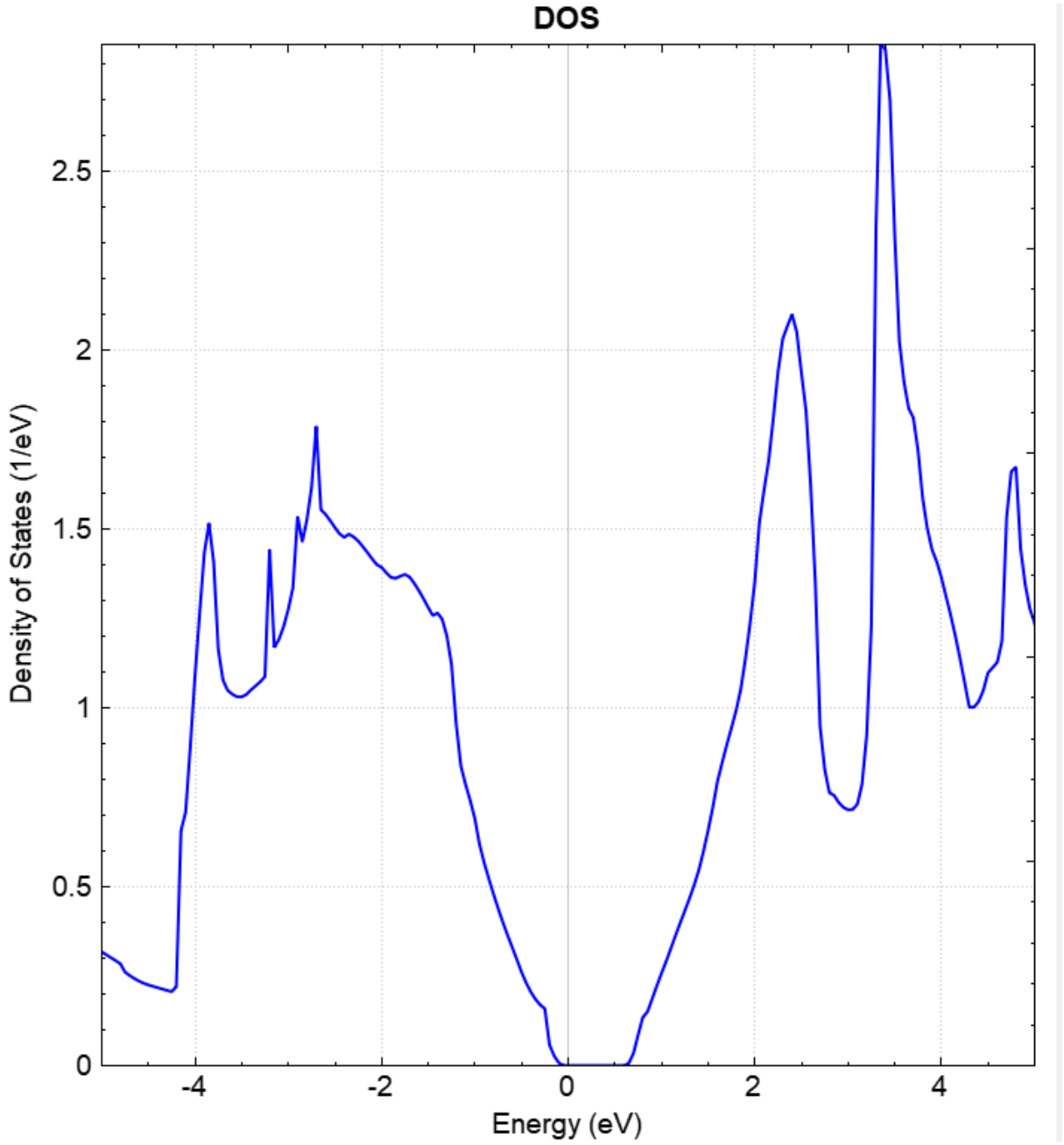
以两步算为例，将参数控制文件 *scf.in* 、*dos.in* 和结构文件 *structure.as* 上传到服务器，按照结构弛豫中介绍的方法依次执行 *DS-PAW scf.in* 、*DS-PAW dos.in* 。

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* 、*scf.h5* 、*dos.h5* 等输出文件。

- *DS-PAW.log* ： DS-PAW 态密度计算之后得到的日志文件；
- *dos.h5* ： 态密度计算对应的 h5 文件，具体结构见输出文件格式说明 部分；

可使用 **python** 对 *dos.h5* 进行数据处理，具体操作见辅助工具使用教程 部分。处理得到的态密度图效果应如下所示：


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

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* 和自洽计算得到的电荷密度文件 *rho.bin* ，*pdos.in* 如下：

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

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

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

pdos.in 输入参数介绍：

投影态密度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` ： 控制态密度计算中投影计算的开关；

2.6.2 run 程序运行

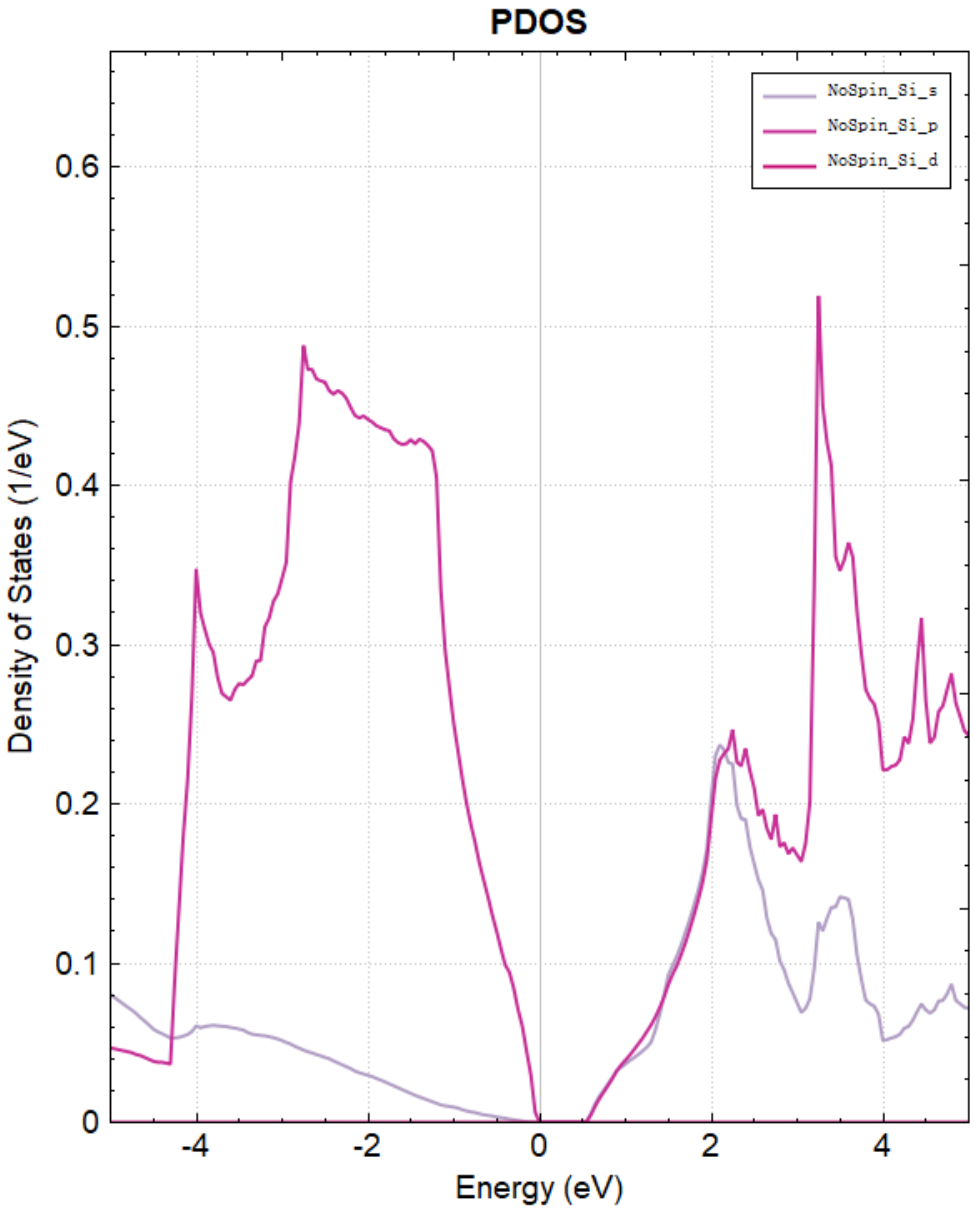
准备好输入文件 *pdos.in* 和 *structure.as* 以及 *rho.bin* 之后，将文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 *DS-PAW pdos.in* 。

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.h5* 等输出文件。

- *DS-PAW.log* ： DS-PAW 态密度计算之后得到的日志文件；
- *dos.h5* ： 态密度计算对应的 **h5** 输出文件；投影态密度的数据保存在 *dos.h5* 文件中，具体的数据结构详见输出文件格式说明 部分；

可使用 **python** 对 *dos.h5* 进行数据处理，具体操作见辅助工具使用教程 部分。处理得到的投影态密度图效果应如下所示：


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)

2.7 potential 势函数计算

势函数计算有两种方式可完成，task=potential 的两步法及 task=scf 的一步法。此节以 Si 体系为例，介绍两种方法下对应的参数设置。

2.7.1 Si 势函数计算输入文件

2.7.1.1 task = potential 两步算

输入文件包含参数文件 *scf.in* 、*potential.in* 和结构文件 *structure.as* ，*scf.in* 设置与自洽计算一致，*potential.in* 设置如下：

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

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`： 设置计算类型，本次计算为 `potential` 势函数计算；
- `cal.iniCharge`： 设置读取电荷密度文件的路径，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；

势函数计算中新增参数：

- `potential.type`： 控制势函数保存的类型，当选择 **all** 的时候，势函数计算完成之后会同时保存静电势（离子势和 hartree 势之和）及局域势（静电势和交换关联势之和）；

structure.as 文件同自洽计算。（见 2.2 节）

备注

1. **两步算时，scf.in 和 potential.in 中参数 cal.cutoffFactor 及 cal.cutoff 必须保持一致，否则会出现格点数据不匹配的问题。**
2. **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。**
3. **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。**

2.7.1.2 task = scf 一步算

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

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

备注

1. 势函数一步算对应结果文件为 scf.h5 ， 此时势函数数据存储在 scf.h5 文件中，可直接调用辅助工具使用教程 的势函数处理脚本分析 scf.h5 文件。
2. **io.potential=true 只在 task=scf 时生效**。

2.7.2 run 程序运行

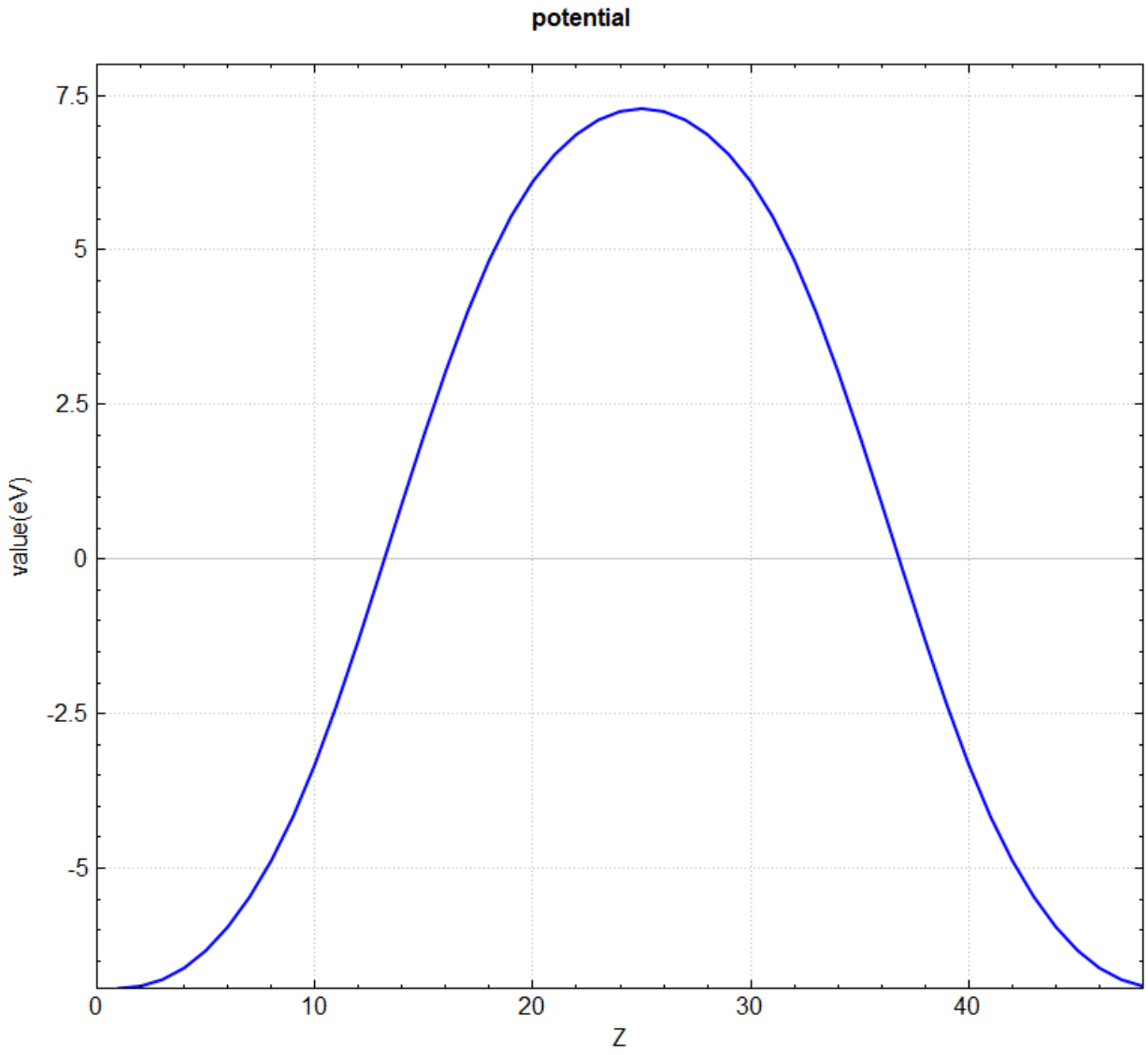
以两步算为例，将参数控制文件 *scf.in* 、*potential.in* 和结构文件 *structure.as* 上传到服务器，按照结构弛豫中介绍的方法依次执行 *DS-PAW scf.in* 、*DS-PAW potential.in* 。

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* 、*scf.h5* 、*potential.h5* 等输出文件。

- *DS-PAW.log* ： DS-PAW 势函数计算之后得到的日志文件；
- *potential.h5* ： 势函数计算对应的 **h5** 输出文件，具体结构见输出文件格式说明 部分；

可使用 **python** 脚本将 *potential.h5* 格式的转化成 **VESTA** 软件支持的格式，也可直接使用脚本对三维势函数进行面内平均，具体操作见辅助工具使用教程 部分。处理得到的真空方向势函数曲线如下所示：



2.8 elf 电子局域密度计算

电子局域密度计算有两种方式可完成，task=elf 的两步法及 task=scf 的一步法。此节以 Si 体系为例，介绍两种方法下对应的参数设置。

2.8.1 Si 电子局域密度计算输入文件

2.8.1.1 task = elf 两步算

输入文件包含参数文件 *scf.in* 和 *ELF.in*，结构文件 *structure.as*，*scf.in* 设置与自洽计算一致，*ELF.in* 设置如下：

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

cal.iniCharge = ./rho.bin
cal.methods = 2

cal.smearing = 1
cal.ksamping = G
cal.kpoints = [10, 10, 10]
cal.cutoffFactor = 1.5
```

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`：设置计算类型，本次计算为 ELF 计算；
- `cal.iniCharge`：设置电荷密度文件的读取路径，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. **两步算时，scf.in 和 ELF.in 中参数 cal.cutoffFactor 及 cal.cutoff 必须保持一致，否则会出现格点数据不匹配的问题。**
2. **ELF 计算不支持 non-collinear 计算。**

2.8.1.2 task = scf 一步算

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

```
# task type
task = scf
#system related
sys.structure = structure.as
sys.symmetry = true
sys.functional = PBE
sys.spin = none
#scf related
cal.methods = 2
cal.smearing = 1
cal.ksamping = G
```

(续下页)

(接上页)

```
cal.kpoints = [10, 10, 10]
cal.cutoffFactor = 1.5
#outputs
io.charge = true
io.wave = true
#elf related
io.elf = true
```

备注

1. 电子局域密度一步算对应结果文件为 scf.h5，此时电子局域密度数据存储在 scf.h5 文件中，可直接调用辅助工具使用教程 的电子局域密度处理脚本分析 scf.h5 文件。
2. **io.elf=true 只在 task=scf 时生效**。
3. **ELF 计算不支持 non-collinear 计算**。

2.8.2 run 程序运行

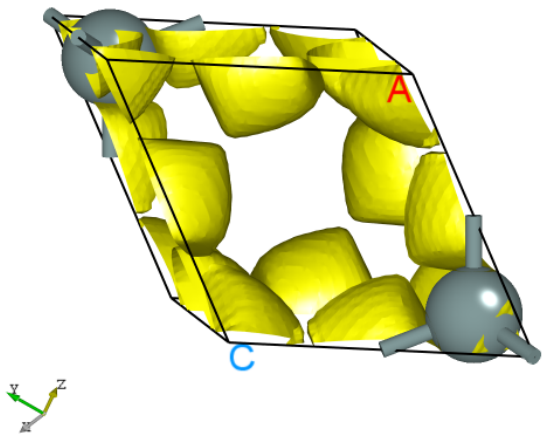
以两步算为例，将参数控制文件 *scf.in* 、*ELF.in* 和结构文件 *structure.as* 上传到服务器，按照结构弛豫中介绍的方法依次执行 *DS-PAW scf.in* 、*DS-PAW ELF.in* 。

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* 、*scf.h5* 、*elf.h5* 等输出文件。

- *DS-PAW.log*： DS-PAW 局域密度计算之后得到的日志文件；
- *elf.h5*： ELF 计算对应的 **h5** 输出文件，具体结构见输出文件格式说明 部分；

可使用 **python** 脚本将 *elf.h5* 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。处理得到的三维电子局域密度图效果应如下所示：



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* ，自洽计算得到的电荷密度文件 *rho.bin* 和波函数文件 *wave.bin* , *pcharge.in* 如下：

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

cal.methods = 2
cal.smearing = 1
cal.ksamping = G
cal.kpoints = [9, 9, 1]
cal.cutoffFactor = 1.5
cal.iniCharge = ./rho.bin
cal.iniWave = ./wave.bin

#pcharge related
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`：设置计算类型，本次计算为部分电荷密度计算；
- `cal.iniCharge`：设置电荷密度文件的读取路径，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`：设置波函数文件的读取路径，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`：设置需进行电荷密度分析的能带序号，这里 [4,5] 表示分析能带 4 和能带 5 的电荷密度；
- `pcharge.kpointsIndex`：设置进行电荷密度分析的 K 点序号，这里 [12] 表示分析两条能带的电荷密度时 k 点序号都为 12；
- `pcharge.sumK`：控制是否将所分析的所有 K 点能带数据相加。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 rho.bin 及波函数文件 wave.bin。**

2.9.2 run 程序运行

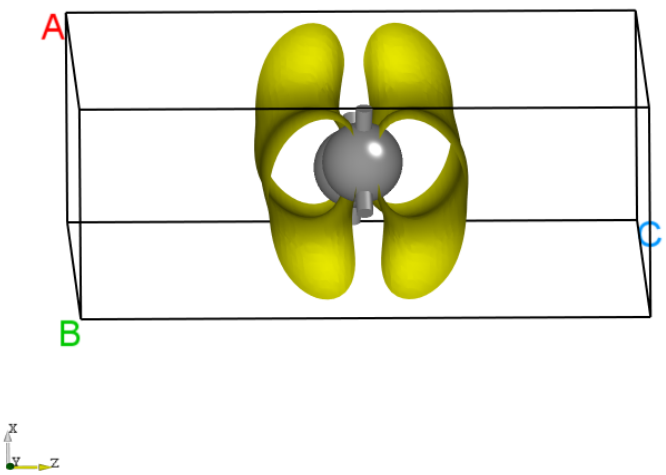
准备好输入文件 *pcharge.in* 、*structure.as* 以及自洽计算得到的 *rho.bin* 、*wave.bin* 文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 *DS-PAW pcharge.in* 。

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.h5* 等输出文件。

- *DS-PAW.log*： DS-PAW 部分电荷密度计算之后得到的日志文件；
- *pcharge.h5* ： 部分电荷密度计算完成之后的 h5 数据文件，此时两条能带的电荷密度数据被保存在 *pcharge.h5* 中，具体的数据结构详见输出文件格式说明 部分；

可使用 **python** 对 *pcharge.h5* 进行数据处理，具体操作见辅助工具使用教程 部分。处理 k 点序号为 **12** 时能带 **4** 的电荷密度图效果应如下所示：



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.spin = none
#scf related
cal.methods = 1
cal.totalBands = 12
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,
```

(续下页)

(接上页)

```
↪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 = true
io.wave = true
```

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`：控制杂化泛函计算的开关，true 表示引入杂化泛函计算；
- `sys.hybridType`：设置杂化泛函的类型，此例为 HSE06；

structure.as 文件同自洽计算。（见 2.2 节）

备注

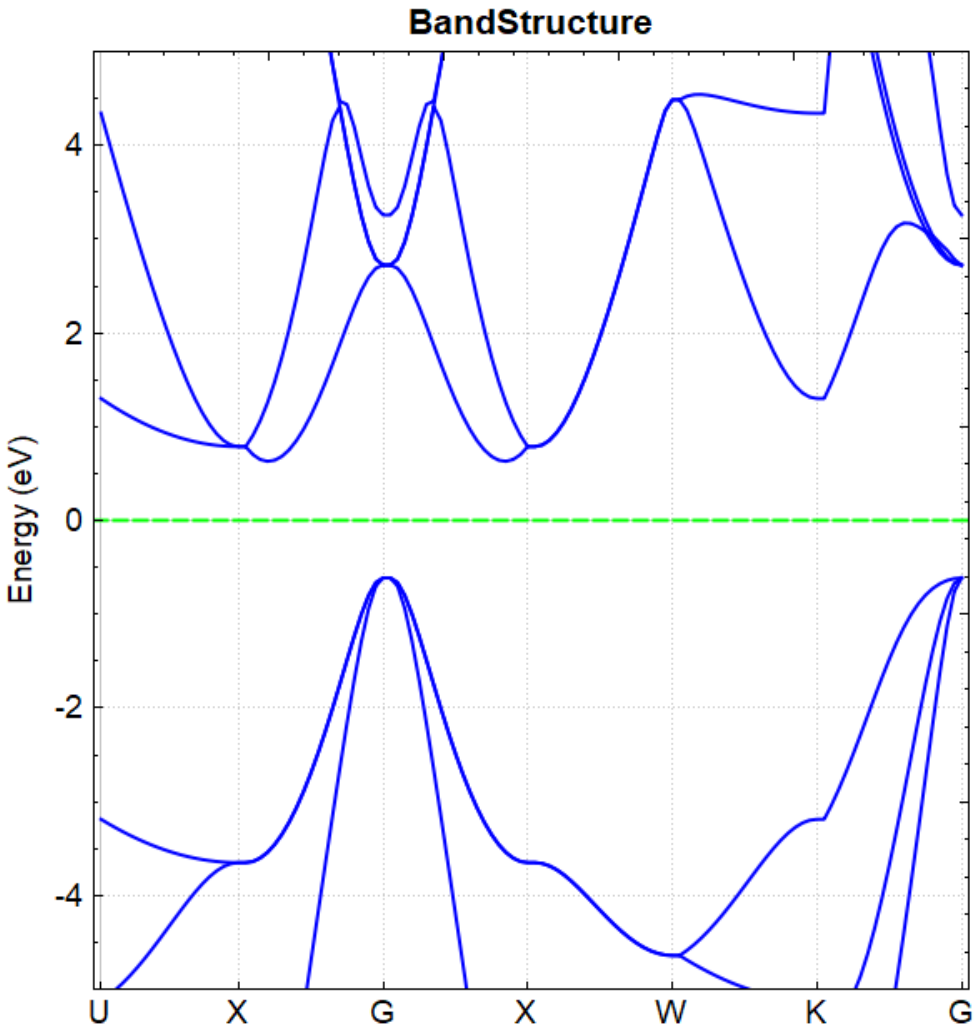
1. **不同于普通计算使用 sys.functional 设置泛函类型，杂化泛函计算通过参数 sys.hybridType 来控制杂化泛函种类。**
2. **杂化泛函计算只支持 task=scf/relax 的计算，因此杂化泛函能带计算只能通过一步算完成。**
3. **杂化泛函计算建议使用 damped MD/conjugated gradient 方法进行电子自洽计算，对应参数设置为 cal.methods = 4/5**
4. **杂化泛函计算也可使用 block Davidson 方法进行电子自洽计算，即此例所设 cal.methods = 1, 此时 scf.mixType 参数会默认为 Kerker**

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* 、*scf.h5* 等输出文件。处理 *scf.h5* 的方法同（见 2.3 节）能带计算的方法，处理得到的能带图效果应如下所示：



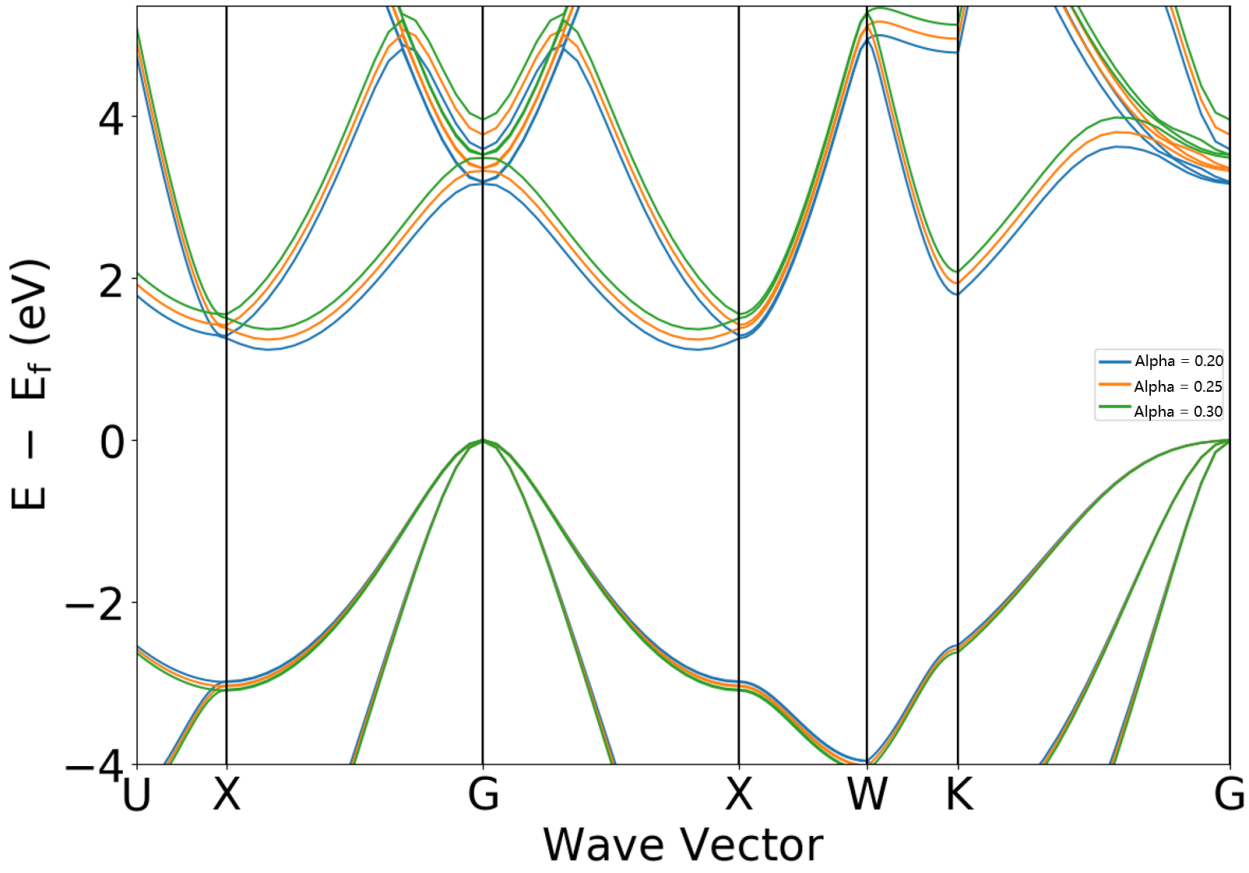
能带图表明打开杂化泛函计算后价带与导带之间的带隙变大，约为 **1.2394** eV，不进行杂化泛函计算得到的能带带隙约为 **0.6433** eV。

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 `sys.hybridAlpha = 0.25`，调整 `sys.hybridAlpha` 参数作以下两次计算：

- *scf.in* 和 *band.in* 中修改参数：`sys.hybridAlpha = 0.20`
- *scf.in* 和 *band.in* 中修改参数：`sys.hybridAlpha = 0.30`

得到如下能带对比图：



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 *DS-PAW.log* 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** eV。

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。

2.11.1 graphite 石墨结构弛豫输入文件

在对石墨进行弛豫时，可选取半经验方法对范德瓦尔斯力进行修正，也可采取泛函修正的方法，下面介绍两种方法下对应的参数设置。

2.11.1.1 半经验修正

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 = true
sys.functional = PBE
sys.spin = none
#scf related
cal.methods = 1
cal.smearing = 1
cal.ksamping = G
cal.kpoints = [21, 21, 7]
cal.cutoff  = 600
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`：控制半经验方法范德瓦尔斯修正的开关，true 表示已打开；
- `corr.VDWType`：设置范德瓦尔斯修正的类型，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. **使用半经验方法对范德瓦尔斯力进行修正时，可选取不同类型的交换关联泛函，sys.functional 的可选值有 PBE/REVPBE/RPBE/PBESOL**
2. **DS-PAW 支持使用半经验方法对范德瓦尔斯力进行修正的同时开启杂化泛函计算。**

2.11.1.2 泛函修正

泛函修正对应的输入文件 *relax.in* 可如下所示：

```
# task type
task = relax
#system related
sys.structure = structure.as
sys.symmetry = true
sys.spin = none
#scf related
cal.methods = 1
cal.smearing = 1
cal.ksamping = G
cal.kpoints = [21, 21, 7]
cal.cutoff  = 600
scf.convergence = 1.0e-05
#relax related
relax.max =  60
relax.freedom = all
relax.convergence =  0.01
relax.methods = CG
#vdw related
sys.functional = vdw-optPBE
```

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：

- `sys.functional`：控制泛函的类型，当选择包含范德瓦尔斯修正的泛函时，设置 vdw-系列的泛函参数即可，此例选取 vdw-optPBE 泛函，支持的泛函类型见参数说明 部分。

> **备注**
>
> 1. **从原理出发，范德瓦尔斯有两种不同的修正方式，分别对应参数 corr.VDW = true（半经验修正）和 sys.functional = vdw-…（泛函修正）。**

2.11.2 run 程序运行

以半经验修正为例，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.h5* 、*latestStructure.as* 等输出文件。（为作对比另添加一组不考虑范德瓦尔斯的计算）

将 *latestStructure.as* 拖入 Device Studio 查看结构，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。

[1] Celso R. C. Rêgo, Luiz N. Oliveira, Polina Tereshchuk, and Juarez L. F. Da Silva. Comparative study of van der waals corrections to the bulk properties of graphite. *Journal of Physics: Condensed Matter*, 2015. doi:10.1088/0953-8984/27/41/415502.

Procedure	a (Å)	c (Å)
vdw-D3G this work	2.463	6.954
PBE this work	2.464	7.914
Experiment	2.462	6.707

2.12 optical 光学性质计算

光学计算有两种方式可完成，task=optical 的两步法及 task=scf 的一步法。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

2.12.1.1 task = optical 两步算

输入文件包含参数文件 *scf.in* 、*optical.in* 和结构文件 *structure.as* ，*scf.in* 设置与自洽计算一致，*optical.in* 设置如下：

optical.in 如下：

```
# task type
task = optical
#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 = [12, 12, 12]
cal.cutoffFactor = 1.5
cal.iniCharge = ./rho.bin

#optical related
optical.grid = 2000
optical.sigma = 0.05
optical.smearing = 1
```

在光学计算中可以尽量保留 sys. 和 cal. 的参数到 *optical.in* 中，之后设置光学计算特有的参数即可：

- `task`：设置计算类型，本次计算为 `task = optical`：光学计算；
- `cal.iniCharge`：设置读取电荷密度文件的路径，支持绝对路径及相对路径，这里./表示当前路径下的 *rho.bin* ；
- `optical.grid`：表示 DS-PAW 计算光学性质时在能量区内的网格点数，此例为 2000；
- `optical.sigma`：决定使用 optical.smearing 决定的展开算法时的展宽宽度，此例为 0.05；

- `optical.smearing`：决定在 optical 计算时对能量展宽时的展宽算法，此例为 1。

2.12.1.2 task = scf 一步算

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

```
# 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 = [12, 12, 12]
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`：控制光学性质计算的开关，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

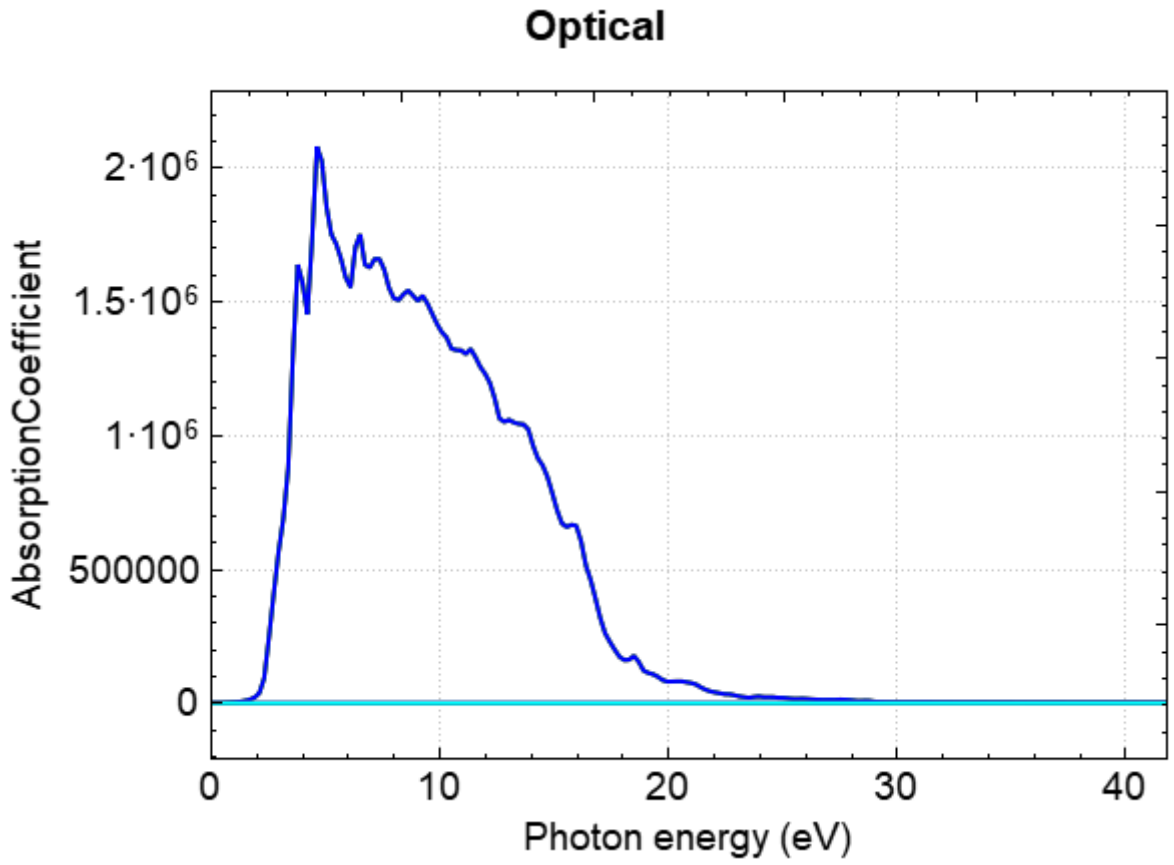
以两步算为例，准备好输入文件之后，将 *scf.in* 、*optical.in* 和 *structure.as* 文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 *DS-PAW scf.in* 、*optical.in* 。

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* 、*scf.h5* 、*optical.h5* 等输出文件。

- *DS-PAW.log*： DS-PAW 光学性质计算之后得到的日志文件；
- *optical.h5*： 光学性质计算完成之后的 **h5** 数据文件，注意 h5 文件的名称与 task 类型严格一致。h5 文件的数据结构详见输出文件格式说明 部分；

可使用 **python** 对 *optical.h5* 或者一步算的结果 *scf.h5* 进行数据处理，具体操作见辅助工具使用教程 部分。处理可得介电常数实部、介电常数虚部、吸光系数、消光系数、光电导率、反射率、折射率、能量损失随能量的变化的曲线，以吸光系数曲线为例，得到的曲线效果图应如下所示：



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：

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

(续下页)

(接上页)

```
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`：设置计算类型，本次计算为 frequency 频率计算；
- `frequency.dispOrder`：设置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`：设置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. **由于固定了 C，O 原子在 x，y 方向的自由度，故两原子只在 z 方向上可动。**

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.h5* 、*frequency.txt* 等输出文件。

- *DS-PAW.log*： DS-PAW 频率计算之后得到的日志文件；
- *frequency.h5*： 频率计算完成之后的 h5 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输出文件格式说明 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.h5* 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	63.844168	401.144726	2129.612084	264.038342
2 f/i	0.051335	0.322546	1.712346	0.212304

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
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`：设置计算类型，本次计算为 elastic 弹性计算；
- `elastic.dispOrder`：设置弹性计算时原子振动的方式, 1 对应中心差分法；
- `elastic.dispRange`：设置弹性计算时原子位移的大小；

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.h5* 、*elastic.txt* 这 3 个文件。

- *DS-PAW.log*： DS-PAW 弹性计算之后得到的日志文件；
- *elastic.h5*：弹性计算完成之后的 **h5** 数据文件，此时弹性模量被保存在 *elastic.h5* 中，具体的数据结构详见输出文件格式说明 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.h5* 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：

刚性弹性矩阵：

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

柔性弹性矩阵：

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

体积模量、剪切模量、杨氏模量和泊松比:

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

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 H 在 Pt(100) 表面扩散为例，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 *structureNo.as* ， *neb.in* 文件如下:

```
task = neb

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

cal.ksamping = G
cal.kpoints = [3,3,1]
cal.cutoffFactor = 1.0
cal.smearing = 1
cal.sigma = 0.05

neb.freedom = atom
neb.springK = 5
neb.images = 3
neb.iniFin = true
neb.method = LBFGS
```

(续下页)

(接上页)

```
neb.convergence = 0.03
neb.stepRange = 0.1
neb.max = 60

io.wave = false
io.charge = 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`： 设置计算类型，本次计算为 neb 过渡态计算；
- `neb.stepRange`：设置过渡态计算中结构弛豫的步长；
- `neb.max`：设置过渡态计算中结构弛豫的最大步数；
- `neb.iniFin`：控制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`：设置过渡态计算中弹簧系数 K；
- `neb.images`：设置过渡态计算中的中间结构的数目；
- `neb.method`：设置过渡态计算中使用的算法；
- `neb.convergence`：设置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
5.60580000 0.00000000 0.00000000
0.00000000 5.60580000 0.00000000
0.00000000 0.00000000 16.81740000
Cartesian Fix_x Fix_y Fix_z
H 2.80881670 4.20393628 6.94088012 F F F
Pt 1.40145000 1.40145000 1.98192999 T T T
Pt 4.20434996 1.40145000 1.98192999 T T T
Pt 1.40145000 4.20434996 1.98192999 T T T
Pt 4.20434996 4.20434996 1.98192999 T T T
Pt 0.00272621 0.00056545 3.91746017 F F F
Pt 0.00271751 2.80233938 3.91708172 F F F
Pt 2.80568712 -0.00141176 3.91894328 F F F
Pt 2.80548220 2.80426217 3.91792247 F F F
Pt 1.39865124 1.40124680 5.84694340 F F F
Pt 4.21951864 1.40156999 5.84719575 F F F
Pt 1.38647954 4.20437926 5.89984296 F F F
Pt 4.23154392 4.20414605 5.89983612 F F F
```

末态结构 *structure04.as* 参考如下

```
Total number of atoms
13
Lattice
5.60580000 0.00000000 0.00000000
0.00000000 5.60580000 0.00000000
0.00000000 0.00000000 16.81740000
Cartesian Fix_x Fix_y Fix_z
H 1.52157824 2.80289997 6.91583941 F F F
```

(续下页)

(接上页)

```
Pt 1.40145000 1.40145000 1.98192999 T T T
Pt 4.20434997 1.40145000 1.98192999 T T T
Pt 1.40145000 4.20434997 1.98192999 T T T
Pt 4.20434997 4.20434997 1.98192999 T T T
Pt 0.02556963 0.00000000 3.90765450 F F F
Pt 0.02708862 2.80290000 3.91082177 F F F
Pt 2.83159105 0.00000000 3.91547525 F F F
Pt 2.82981856 2.80290000 3.90913282 F F F
Pt 1.45998966 1.38039927 5.88134827 F F F
Pt 4.25691060 1.38811299 5.84551487 F F F
Pt 1.45998966 4.22540069 5.88134827 F F F
Pt 4.25691060 4.21768697 5.84551487 F F F
```

备注

1. **neb 计算时初态末态需先进行结构弛豫。**
2. **中间结构的生成可调用 "辅助工具使用教程-过渡态部分" 的 neb_interpolate_structures.py 脚本，完成插值可调用 neb_visualize.py 脚本对插值结构进行预览，可调用 calc_dist.py 脚本查看 image 之间的距离是否合理。**
3. **过渡态计算时的结构文件 structureNo.as 需存放在命名为 No 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 DS-PAW 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。**

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 *structureNo.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* 、*latestStructure00.as* 、*scf.h5* 等输出文件；

» 中间结构 *structureNo.as* 所在文件夹 `No` （参与过渡态计算的中间结构所在文件夹，中间结构的个数由 `neb.images` 参数决定）会生成结构优化所得的 *nebNo.h5* 、*latestStructureNo.as* 等输出文件；

» 最外层目录将会生成 *DS-PAW.log* 、*neb.h5* 这 2 个文件，其中 *neb.h5* 为 `No` 文件夹下各 *nebNo.h5* 文件的信息汇总。

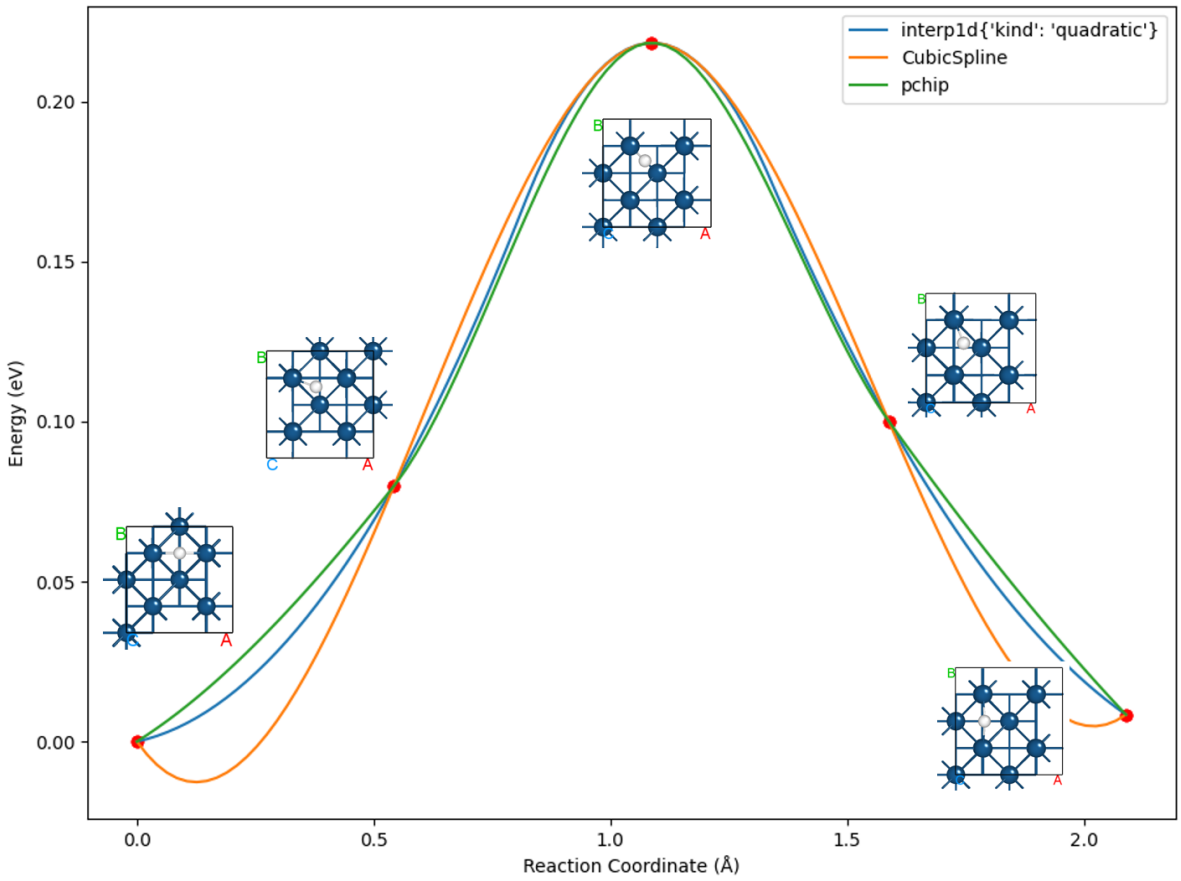
- *DS-PAW.log*： DS-PAW 过渡态计算之后得到的日志文件；
- *neb.h5*： 过渡态计算完成之后的 **h5** 数据文件；此时反应坐标及能量变化等数据被保存在 *neb.h5* 中，具体的数据结构详见输出文件格式说明 部分；

可使用 **python** 脚本 `8neb_check_results.py` 对 neb 计算的结果进行分析，需在完整的 neb 计算目录下执行分析脚本，具体操作见辅助工具使用教程 部分。

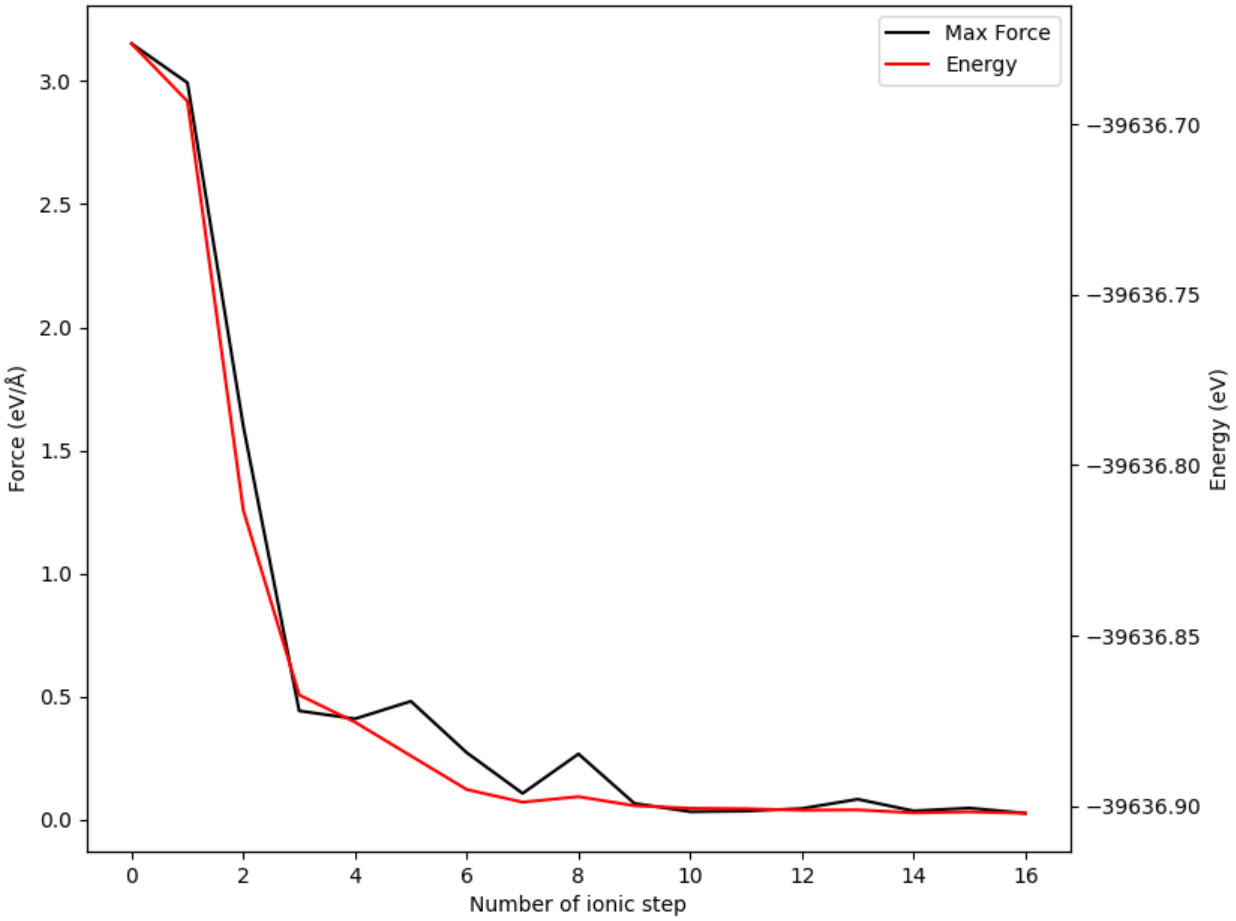
处理可得到 NEB 计算各构型的能量和受力表格：

Image	Force (eV/Å)	Reaction coordinate (Å)	Energy (eV)	Delta energy (eV)
00	0.1803	0.0000	-39637.0984	0.0000
01	0.0263	0.5428	-39637.0186	0.0798
02	0.0248	1.0868	-39636.8801	0.2183
03	0.2344	1.5884	-39636.9984	0.1000
04	0.0141	2.0892	-39637.0900	0.0084

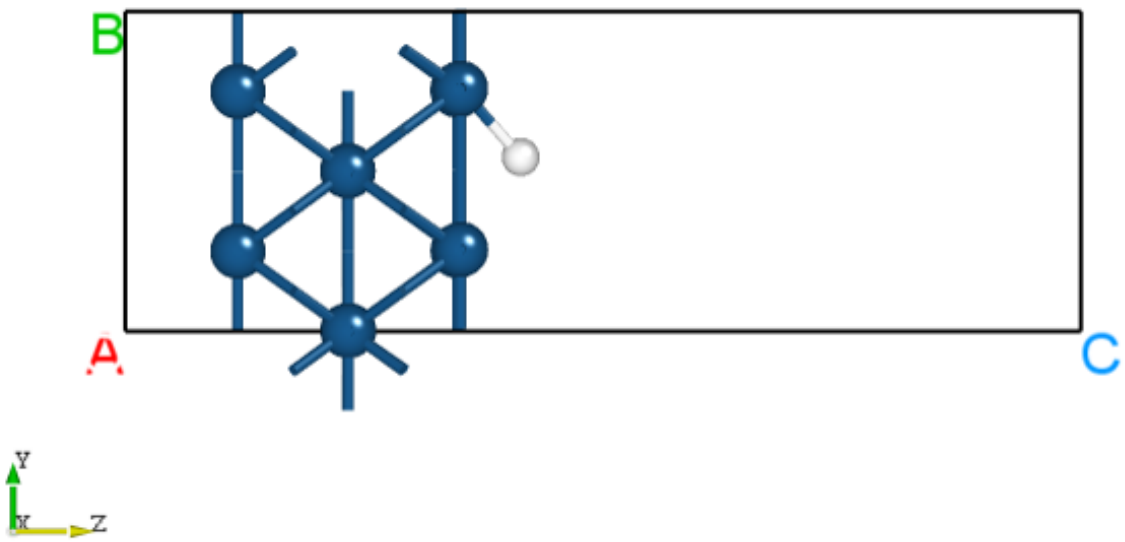
处理得到的势垒曲线效果应如下所示：



处理得到的 02 image 在弛豫过程中的能量与受力应如下所示：



另可使用 **python** 脚本 *neb_movie.py* 分析过渡态搜寻中的轨迹变化，生成的 *neb_movie.json* 文件可用 Device Studio 打开，截取一帧如下所示：



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，介绍如何用两种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

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
```

(续下页)

(接上页)

```
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`：设置计算类型，本次计算为 phonon 声子计算；
- `phonon.type`：设置声子计算的类型，bandDos 对应计算声子能带和态密度；
- `phonon.structureSize`：设置声子计算时超胞的大小；
- `phonon.primitiveUVW`：设置声子能带计算时原胞 UVW 的系数；
- `phonon.method`：设置声子计算的方法，dfpt 为密度泛函微扰理论方法；
- `phonon.qpoints`：设置声子计算 q 空间网格取样为 41*41*41；
- `phonon.dosRange`：设置声子态密度计算的能量区间为 [0,20]；
- `phonon.qpointsLabel`：设置声子能带计算时高对称点标签；
- `phonon.qpointsCoord`：设置声子能带计算时高对称点坐标；
- `phonon.qpointsNumber`：设置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
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. **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。**

3. **phonon.iniPhonon 可指定路径读取声子计算（phonon.type = phonon）得到的 phonon.h5 文件，从而直接进行能带与态密度的计算。**
4. **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

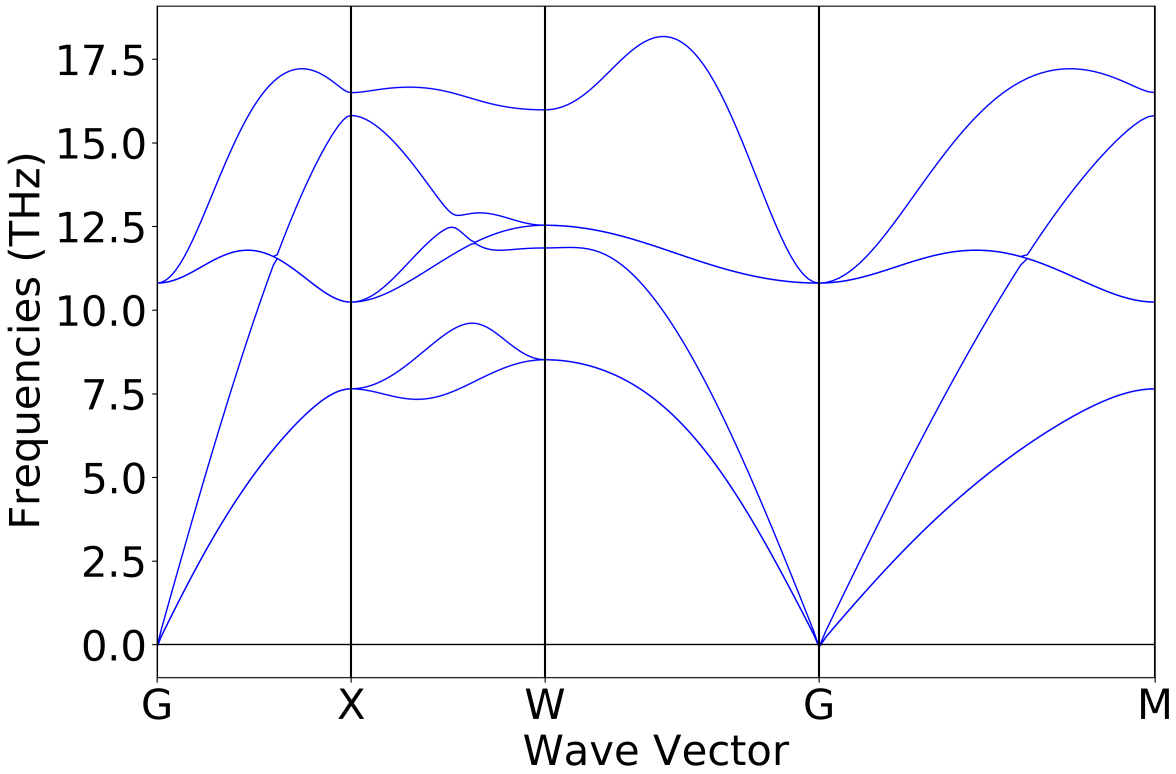
准备好输入文件之后，将 *phonon.in* 和 *structure.as* 文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 *DS-PAW phonon.in* 。

2.16.3 analysis 计算结果分析

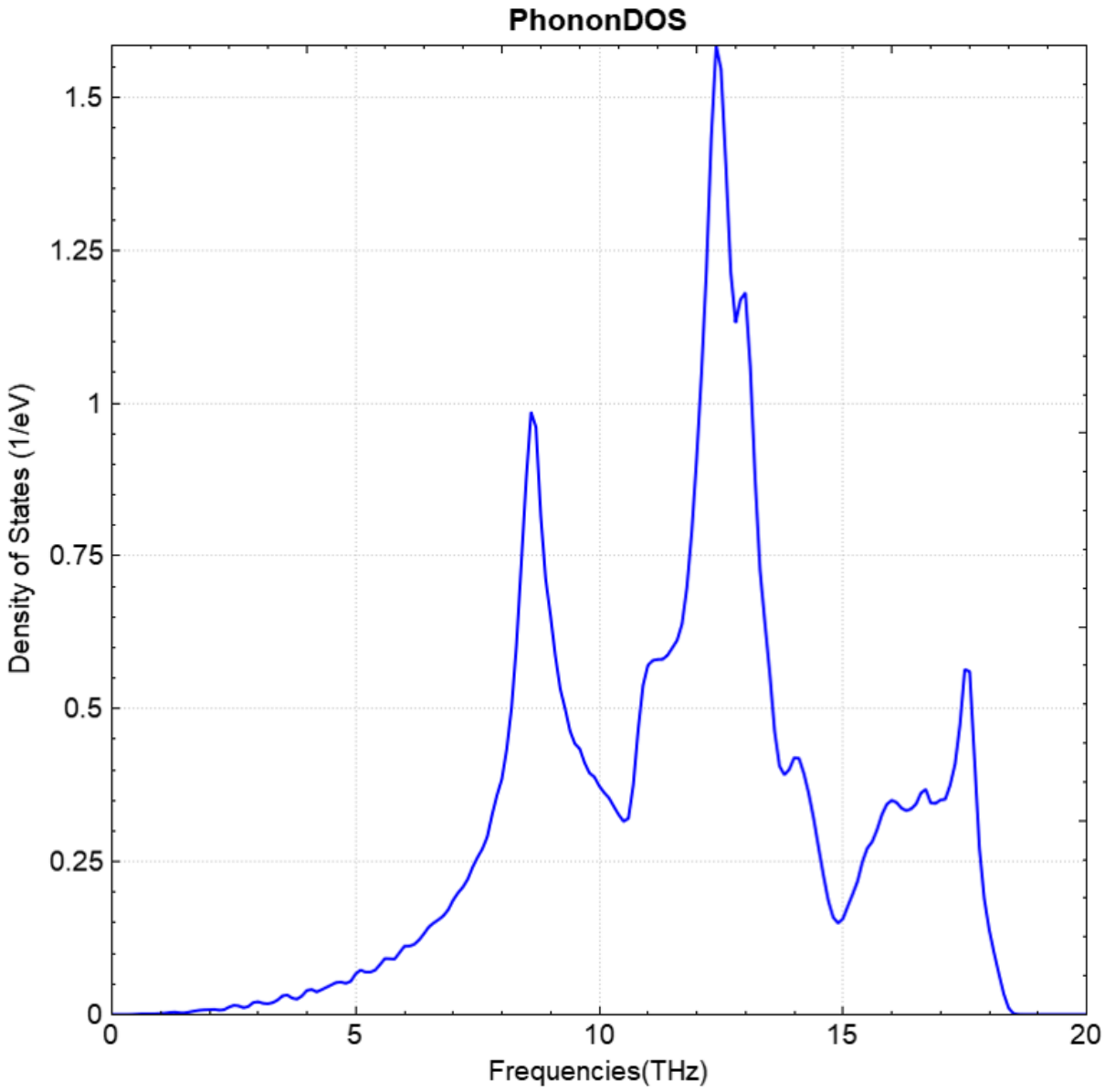
根据上述的输入文件，计算完成之后将会得到 *DS-PAW.log* 、*phonon.h5* 、*dfpt.json* 和 *dfpt.as* 等输出文件。

- *DS-PAW.log*： DS-PAW 声子计算之后得到的日志文件；
- *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.h5*： 声子计算完成之后的 **h5** 数据文件；此时声子能带数据被保存在 *phonon.h5* 中，具体的数据结构详见输出文件格式说明 部分；

可使用 **python** 脚本对 *phonon.h5* 进行数据处理，处理得到的声子能带和态密度效果图应如下（a）、（b）所示：



(a)


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

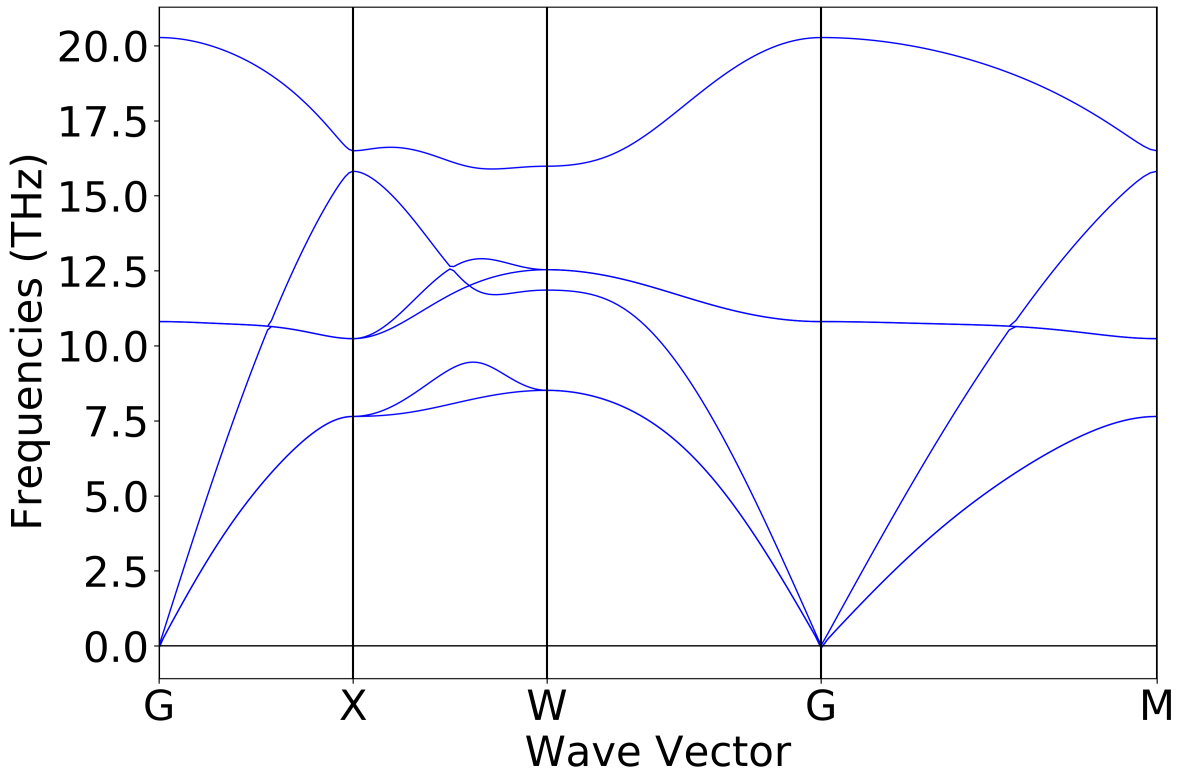
(b)

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
```

得到的声子能带效果图应如下（c）所示：



(c)

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 方法计算得到输出文件与 dfpt 方法不同。

```
task = phonon

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

cal.methods = 1
```

(续下页)

(接上页)

```
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.qpoints = [41,41,41]
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.h5* 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 **python** 脚本处理有限位移方法计算得到的 *phonon.h5* 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 = false
sys.functional = PBE
#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：

- `sys.spin`：设置体系自旋类型，non-collinear 表示非线性自旋；
- `sys.soi`：控制是否考虑自旋轨道耦合效应；该参数在 sys.spin=non-collinear 时生效；

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
```

能带计算的输入文件 *soiband.in* ， 内容如下

```
# task type
task = band
#system related
sys.structure = structure.as
sys.symmetry = true
sys.functional = PBE
#scf related
cal.methods = 2
cal.smearing = 1
cal.ksamping = G
```

(续下页)

(接上页)

```
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**。

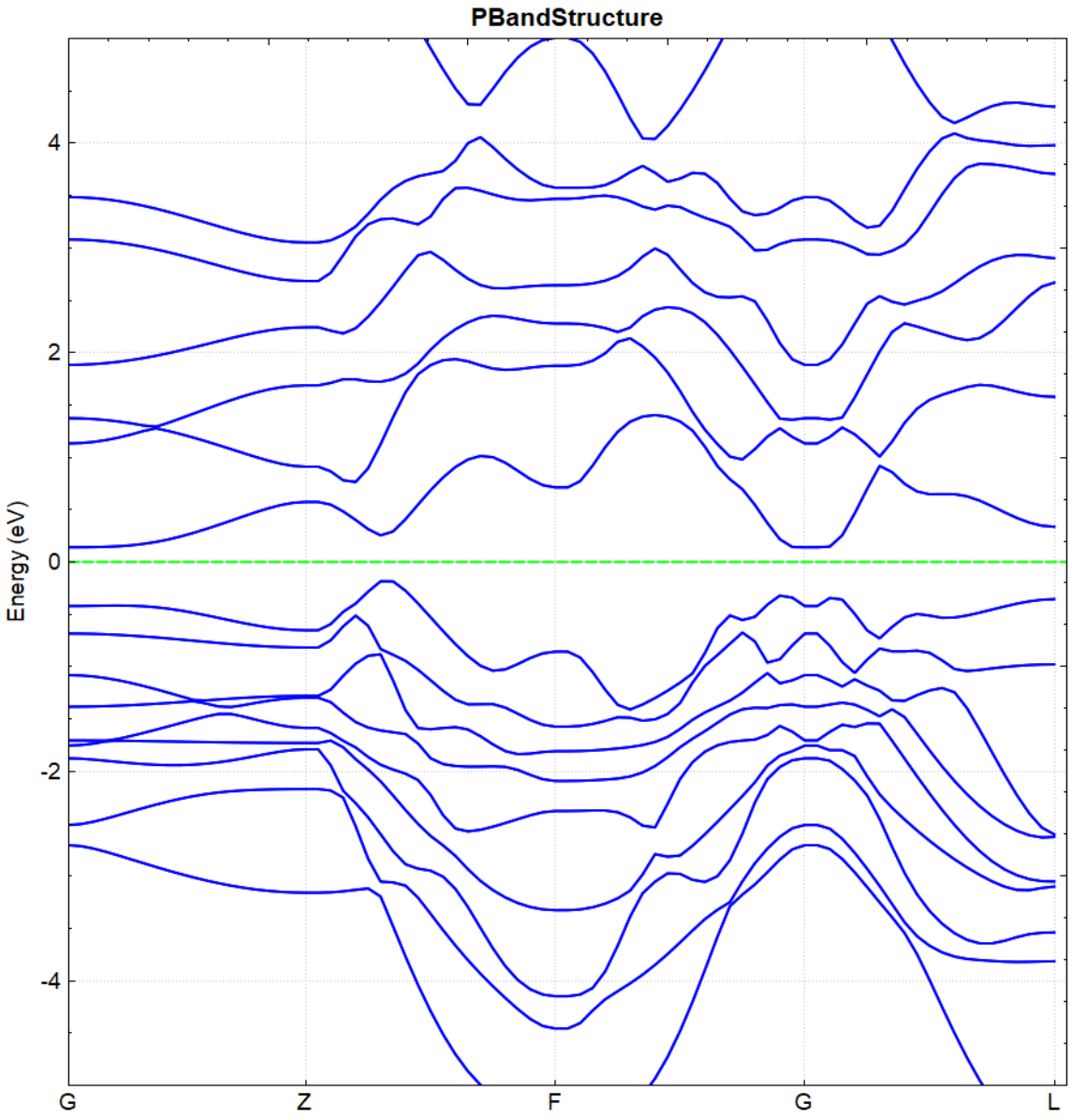
2.17.2 run 程序运行

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* 。

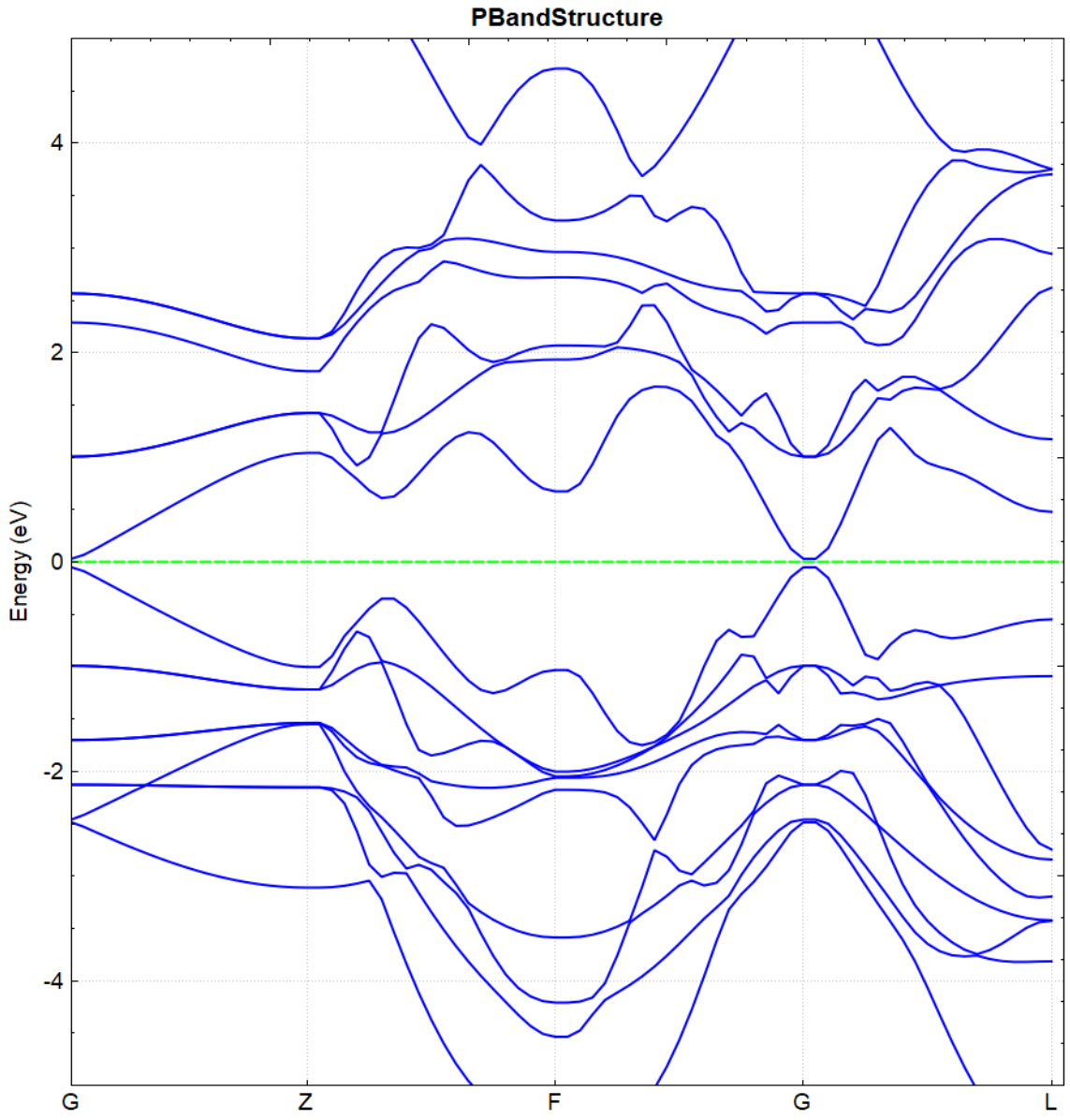
2.17.3 analysis 计算结果分析

根据上述的输入文件，计算完成之后将会得到 *DS-PAW.log* 、*scf.h5* 、*band.h5* 等输出文件。

处理 *band.h5* 的方法同（见 2.3 节）能带计算的方法，得到的能带效果图应如下图（a）所示，另作不考虑自旋轨道耦合的计算，得到的能带效果图应如下（b）所示：



(a)



(b)

从 *DS-PAW.log* 可读取 **BandGap** 值，图（a）和图（b）的带隙值分别为 **0.3251** eV 和 **0.0814** eV，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 = PBE
sys.spin = none

#scf related
cal.methods = 1
cal.smearing = 1
cal.ksamping = G
cal.kpoints = [1, 1, 1]
cal.sigma = 0.1

#aimd related
aimd.ensemble = NPT
aimd.thermostat = langevin
aimd.atomFCoeffElements = [H_1]
aimd.atomFCoeffs = [1]
aimd.latticeFCoeff = 1
aimd.pressure = 100
aimd.timeStep = 1
aimd.totalSteps = 2000
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`：设置计算类型，本次计算为 aimd 分子动力学模拟计算；
- `aimd.ensemble`：设置分子动力学模拟时选用的系综，此例系综设置为 NPT ；
- `aimd.thermostat`： 设置分子动力学模拟时选用的恒温器或恒压器，此例选用 langevin 控温控压；
- `aimd.atomFCoeffElements`：设置考虑为 langevin 原子的元素名称，此例将其中一个氢原子设置为 langevin 原子，将该氢原子重命名为 H_1 ；
- `aimd.atomFCoeffs`：设置考虑为 langevin 原子对应的摩擦系数，单位 ps-1；
- `aimd.latticeFCoeff`：设置 langevin 恒温器中晶胞摩擦系数的大小，单位 ps-1；
- `aimd.pressure`：设置 NPT 模拟时体系的目标压强值，单位 kbar；

- `aimd.timeStep`：设置分子动力学模拟时的时间步长，单位 fs；
- `aimd.totalSteps`：设置分子动力学模拟的总步数；
- `aimd.iniTemp`：设置分子动力学模拟时的初始温度，单位 K；

structure.as 文件参考如下：

```
Total number of atoms
3
Lattice
4.00000000 0.00000000 0.00000000
0.00000000 4.00000000 0.00000000
0.00000000 0.00000000 4.00000000
Cartesian
H 2.63934013 1.89542007 1.58223984
H_1 1.36065987 2.11498988 2.45934006
O 1.65002999 1.88501012 1.54065994
```

备注

1. **元素重命名规则为“原元素名 + 下划线 + 自定义字段”。**
2. **此例中设置第二个氢原子为 langevin 原子并将其重命名为 H_1，在 structure.as 文件中需手动对该原子的元素名称进行修改。**
3. **计算体系中存在自定义元素名称 H_1，程序会自动查找 H_1 对应的 H 元素的赝势，用户无需额外准备新的赝势。**

2.18.2 run 程序运行

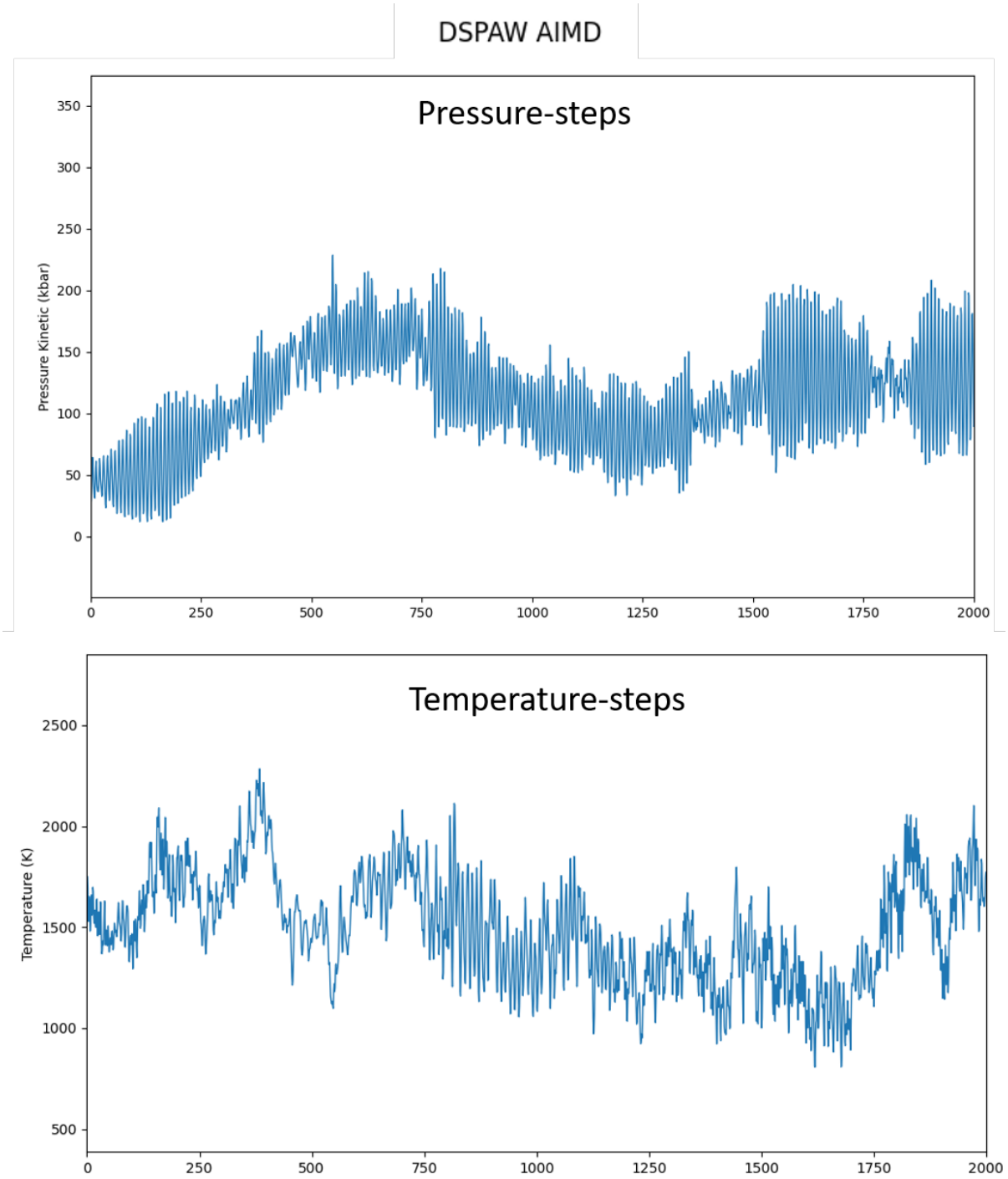
准备好输入文件之后，将 *aimd.in* 和 *structure.as* 文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 *DS-PAW aimd.in* 。

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.h5* 、*latestStructure.as* 等输出文件。

- *DS-PAW.log*： DS-PAW 分子动力学模拟计算得到的日志文件；
- *aimd.h5*：分子动力学计算对应的 **h5** 输出文件；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.h5* 中，具体的数据结构详见输出文件格式说明 部分；
- *latestStructure.as*：分子动力学模拟计算所得的末态 as 结构文件，保存末态构型和速度信息；

可使用 **python** 脚本对 *aimd.h5* 文件进行数据处理，具体操作见辅助工具使用教程 部分。在 NPT 系综下模拟 2000 步得到的 **压力随时间**、**温度随时间**变化曲线效果图应如下所示：



备注

1. 不同系综对应可选热浴范围：NVE 系综可选 andersen 热浴、NVT 系综可选 andersen、noseHoover

和 langevin 三种热浴，NPT 系综可选 langevin 热浴、NPH 系综可选 langevin 热浴；

2. **如需模拟高温退火过程，设置 aimd.ensemble 为 SA，同时通过 aimd.iniTemp 和 aimd.finTemp 设置初始和末态温度即可；**
3. **参数 aimd.finTemp 只在模拟退火时生效，恒温系综如 NPT 和 NVT，末态温度等于初态温度。**
4. **当模拟体系中含 langevin 原子时，建议将 langevin 原子对应的赝势文件存放在计算目录，以避免程序找不到赝势文件报错 E3058。**

2.19 efield 外加电场计算

本节将以硅烯模型的能带计算为例，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 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
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，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

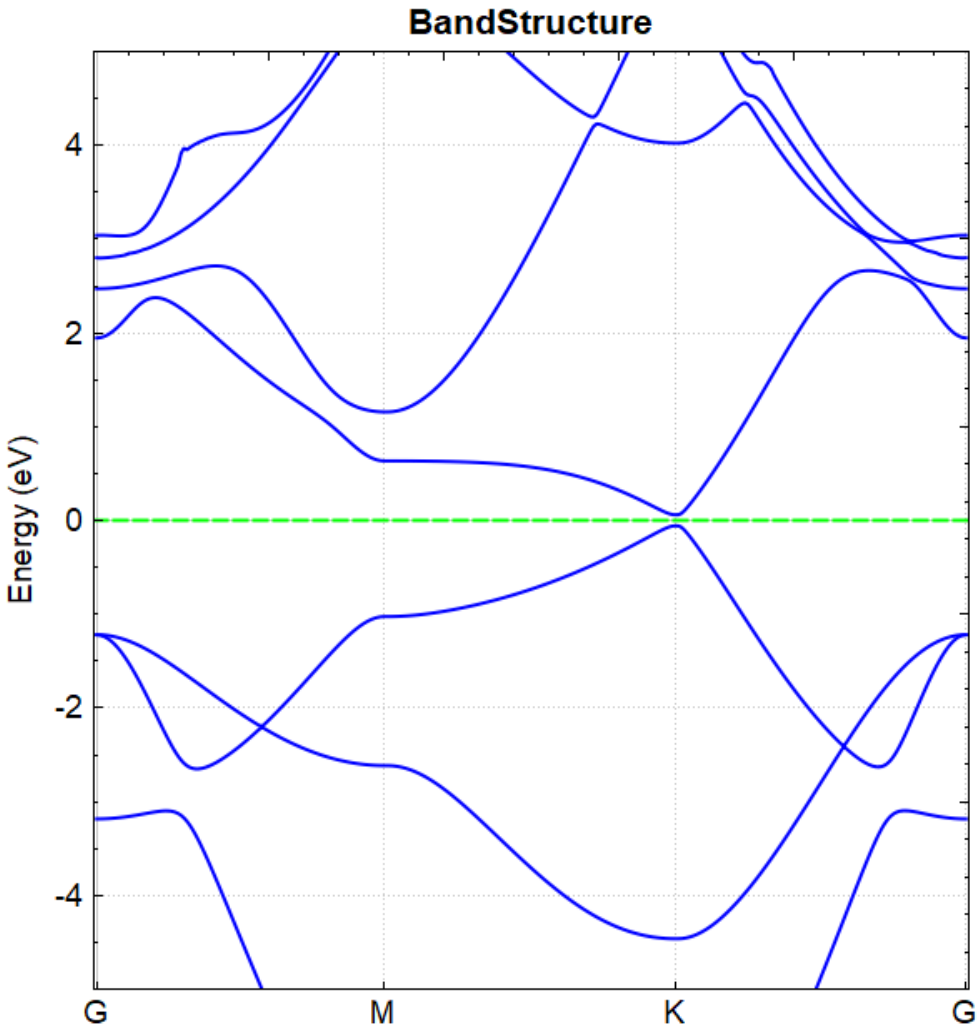
准备好输入文件之后，将 *Efield.in* 和 *structure.as* 文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 *DS-PAW Efield.in* 。

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* 、*scf.h5* 等输出文件。

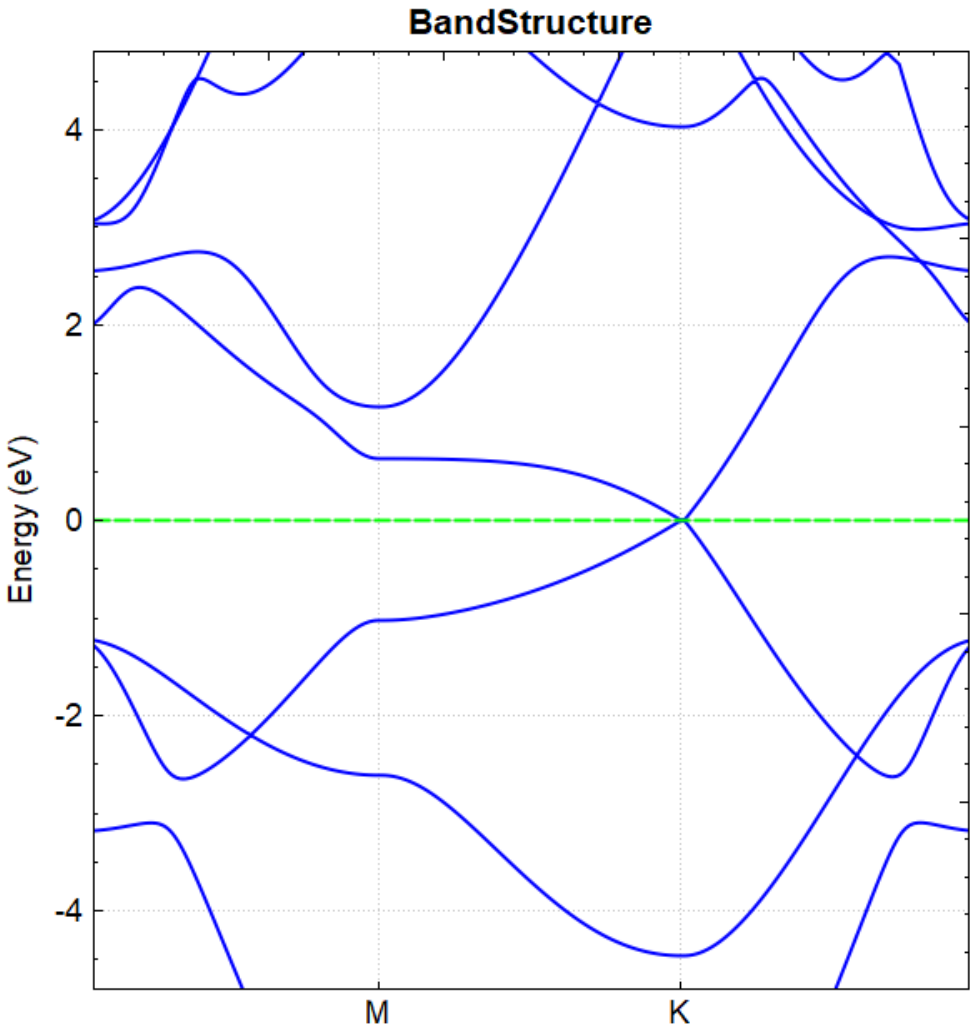
scf.h5：自洽计算对应的 **h5** 输出文件，当 io.band = true 时，*scf.h5* 文件会写入能带数据；

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)

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 *DS-PAW.log* 文件可读出加电场与不加电场 **BandGap** 的值分别为 **0.1176** eV 和 **0.0010** eV 。

备注

1. **外加电场的单位 eV/Å 为原子受力的单位**

2.20 polarization 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* 和一系列不同相结构的结构文件 *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.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`：控制自洽计算中铁电计算的开关；

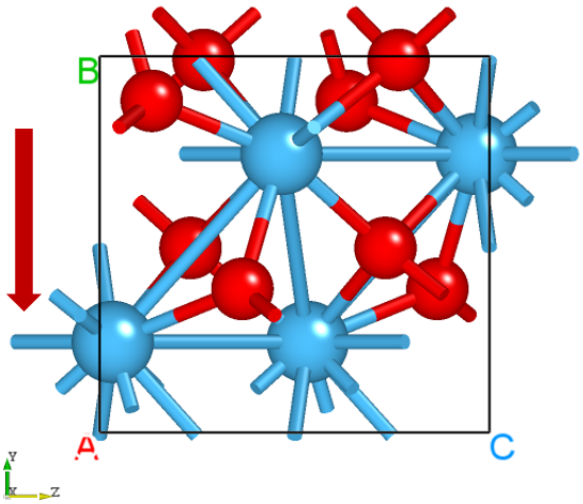
HfO_2 极化方向向下的铁电相结构 *structure.as* 文件参考如下：

```
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 1.22145222 0.17639072
Hf 1.34815269 3.75802848 2.45245381
Hf 3.69806665 1.22145222 2.80523525
Hf 3.69806665 3.75802848 5.08129834
O 0.35195212 1.93667284 1.92589951
O 0.35195212 4.47324910 0.70294502
O 2.32678304 2.48829365 3.85528783
O 2.32678304 5.02486989 4.03124575
O 2.71943629 5.02486989 1.40240122
O 2.71943629 2.48829365 1.22644331
```

(续下页)

(接上页)

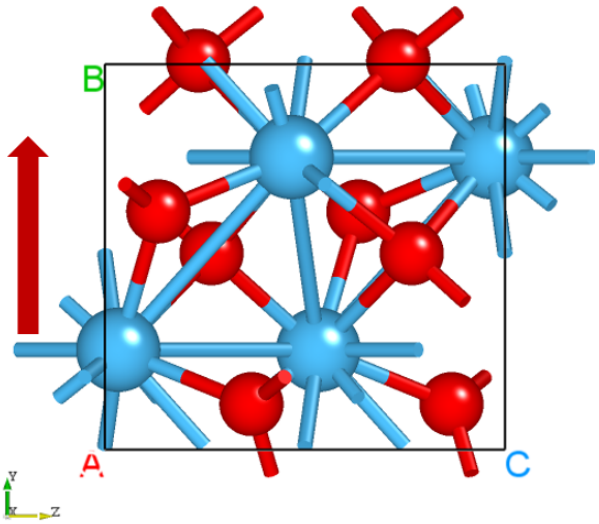
```
O 4.69426723 1.93667284 4.55474404
O 4.69426723 4.47324910 3.33178954
```



铁电相极化方向向下

HfO_2 极化方向向上的铁电相结构 *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 1.31512402 0.17639072
Hf 1.34815269 3.85170026 2.45245381
Hf 3.69806665 1.31512402 2.80523525
Hf 3.69806665 3.85170026 5.08129834
O 0.35195212 0.59990340 1.92589951
O 0.35195212 3.13647965 0.70294502
O 2.32678304 2.58485884 4.03124575
O 2.32678304 5.12143510 3.85528783
O 2.71943630 5.12143510 1.22644331
O 2.71943630 2.58485884 1.40240122
O 4.69426723 0.59990340 4.55474404
O 4.69426723 3.13647965 3.33178954
```



铁电相极化方向向上

在极化向下和极化向上的结构中插入系列中间过渡结构，使用 **线性插值**（neb.linear_interpolate）的方法，具体参见辅助工具 *neb_structure.py* 脚本，本例插入中间结构 **11** 个，包括初末态极化相共 **13** 个构型，对所有构型依次进行极化计算。

2.20.2 run 程序运行

准备好输入文件之后，将 *polarization.in* 和各 *structure.as* 文件上传到服务器，将 13 个结构放入 13 个目录，依次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 13 组 *DS-PAW.log* 、*scf.h5* 和 *polarization.txt* 等输出文件。

- *DS-PAW.log*： DS-PAW 铁电计算之后得到的日志文件；
- *scf.h5*： 自洽计算对应的 **h5** 输出文件，注意 h5 文件的名称与 task 类型严格一致。h5 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，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，便于用户快速获取信息。

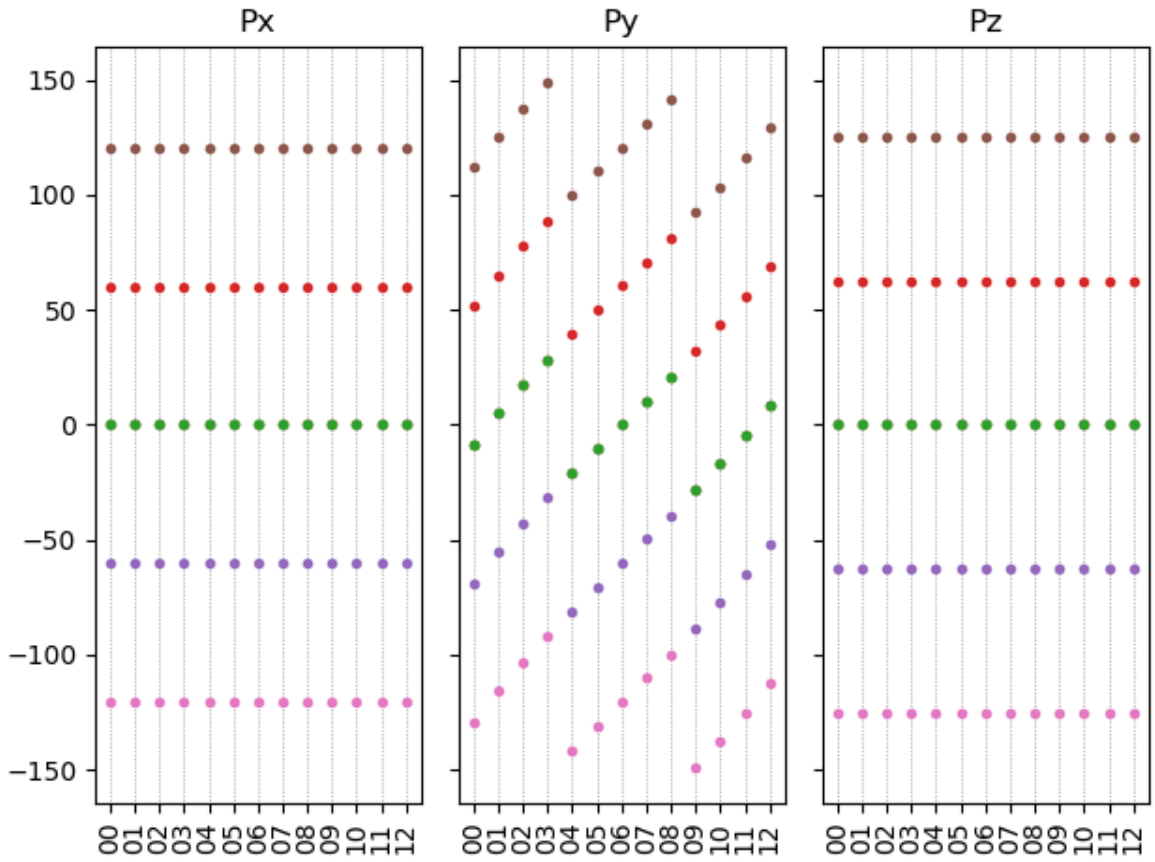
以极化向下（00）铁电相体系为例，从 *polarization.txt* 文件可得 HfO_2 的铁电极化数据如下所示：

Total(x y z) ((μC/cm^2))		
-0.000043	-8.715604	-0.000002
Quantum(x y z) (μC/cm^2)		
60.067225	60.387821	62.584436

以极化向上（12）铁电相体系为例，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))		
-0.000049	8.715446	0.000001
Quantum(x y z) (μC/cm^2)		
60.067225	60.387821	62.584436

可使用 *PolaTotal.py* 脚本对写入极化数据的 *scf.h5* 文件进行数据处理，具体操作见辅助工具使用教程 部分。对 13 组铁电计算的数据进行处理，得到结果图如下：



13 组结构对应极化数值

上图为经过极化量子周期性换算得到的 x, y ,z 三个方向的极化强度 Px, Py, Pz。因 HfO_2 极化方向为 y 方向，Px, Pz 数值不随原子位移而改变。

取 Py 方向极化数最接近 0 的一组数据分析，HfO_2 的极化强度值为铁电相（极化向下，序号 00 或极化向上，序号 12）与中心对称相（过渡态，序号 06）的极化数之差，结合 *polarization.txt* 文件及如上极化数据图，求得：

00 与 06 构型的极化差值为 **-69.103 μC/cm^2**

12 与 06 构型的极化差值为 **69.103 μC/cm^2**

遂，HfO_2 的极化强度值为 **69.103 μC/cm^2**

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 = true
sys.functional = PBE
sys.spin = none

#scf related
cal.methods = 1
cal.smearing = 1
cal.ksamping = G
cal.kpoints = [10, 10, 10]
cal.cutoff = 650
#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`：控制自洽计算中 bader 电荷计算的开关；

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* 、*scf.h5* 、*bader.txt* 等输出文件。

- *DS-PAW.log*： DS-PAW bader 电荷计算之后得到的日志文件；
- *scf.h5*： 自洽计算对应的 **h5** 输出文件，注意 h5 文件的名称与 task 类型严格一致。h5 文件解析见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，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。

Total number of valence electronics: 64

Element	X	Y	Z	Charge	AtomicVolume	MinDistance
Cl	0.25	0.25	0.25	7.85852	35.893	1.65799
Cl	0.75	0.75	0.25	7.85704	35.83	1.65799
Cl	0.75	0.25	0.75	7.84024	35.0495	1.65799
Cl	0.25	0.75	0.75	7.87537	36.6765	1.65799
Na	0.75	0.25	0.25	8.14221	10.0598	1.10532
Na	0.25	0.75	0.25	8.14223	10.0607	1.10532
Na	0.25	0.25	0.75	8.14221	10.0598	1.10532
Na	0.75	0.75	0.75	8.14221	10.0598	1.10532

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 = 650

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`：控制能带计算中能带反折叠计算的开关；

- `band.primitiveUVW`：设置 UVW 系数，超胞的晶格矢量乘上 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* 。

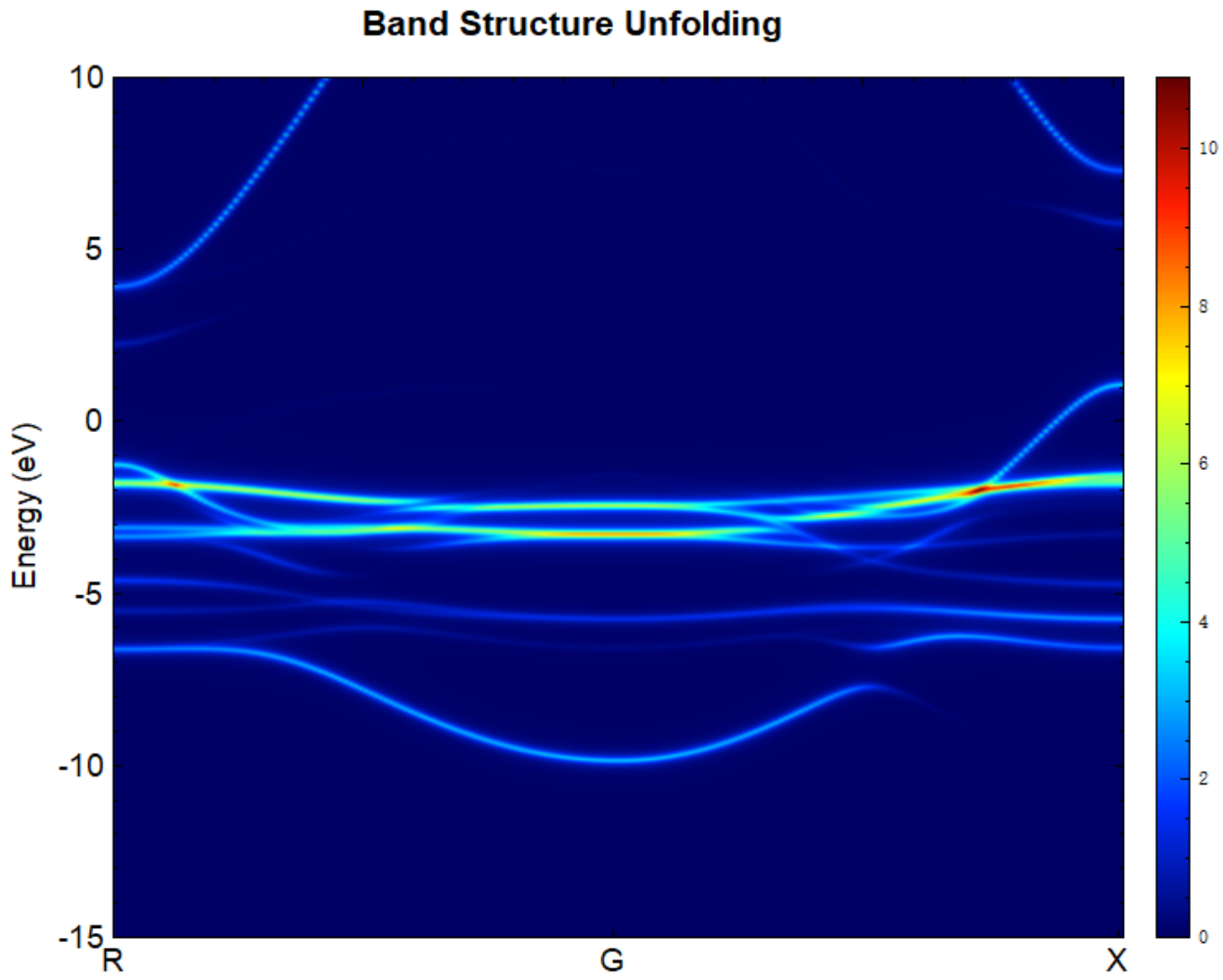
2.22.3 analysis 计算结果分析

根据上述的输入文件，计算完成之后将会得到 *DS-PAW.log* 、*scf.h5* 、*band.h5* 等输出文件。

band.h5： 能带计算对应的 **h5** 输出文件，相比能带计算该文件新增 **UnfoldingBandInfo** 部分，具体结构解析见输出文件格式说明 部分。

可使用脚本 *bandunfolding.py* 对 *band.h5* 进行数据处理，具体操作见辅助工具使用教程 部分。该例得到的能带效果图应如下所示，与文献报道结果[2] 一致。

[2] Mingxing Chen and M. Weinert. Layer k-projection and unfolding electronic bands at interfaces. *Phys. Rev. B*, 98:245421, Dec 2018. doi:10.1103/PhysRevB.98.245421.


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

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。

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 task 的可选值如下：

- `task`：设置计算类型，新增 **epsilon** 参数，此处对应介电常数的计算；

> **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.h5* 、*epsilon.txt* 等输出文件。

- *DS-PAW.log*： DS-PAW 介电常数计算之后得到的日志文件；
- *epsilon.h5*： 介电常数计算对应的 **h5** 输出文件，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，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[3] **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
cal.methods = 1
cal.smearing = 1
cal.ksamping = G
cal.kpoints = [10, 10, 10]
```

(续下页)

[3] M. Gajdo, K. Hummer, G. Kresse, J. Furthmüller, and F. Bechstedt. Linear optical properties in the projector-augmented wave methodology. *Phys. Rev. B*, 73:045112, Jan 2006. doi:10.1103/PhysRevB.73.045112.

(接上页)

```
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`：设置计算类型，新增 **epsilon** 参数，此处对应压电张量计算；
- `scf.convergence`：设置介电张量计算电子收敛的精度，建议提高精度，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.h5* 、*epsilon.txt* 等输出文件。

- *DS-PAW.log*： DS-PAW 压电张量计算之后得到的日志文件；
- *epsilon.h5*： 介电常量计算对应的 **h5** 输出文件，具体的数据结构详见输出文件格式说明 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，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[4] **-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 = 650
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：

[4] Fabio Bernardini, Vincenzo Fiorentini, and David Vanderbilt. Spontaneous polarization and piezoelectric constants of iii-v nitrides. *Phys. Rev. B*, 56:R10024–R10027, Oct 1997. doi:10.1103/PhysRevB.56.R10024.

```
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.h5* 、*latestStructure.as* 等输出文件。

- *relax.h5*： 弛豫计算对应的 **h5** 输出文件；
- *latestStructure.as*： 弛豫终点的 as 结构文件，可直接查看数据；

将 *latestStructure.as* 拖入 Device Studio 查看结构，或直接打开该文件，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`：控制声子计算中热力学计算的开关，仅在 phonon.method = dfpt 时生效；
- `phonon.thermalRange`：设置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

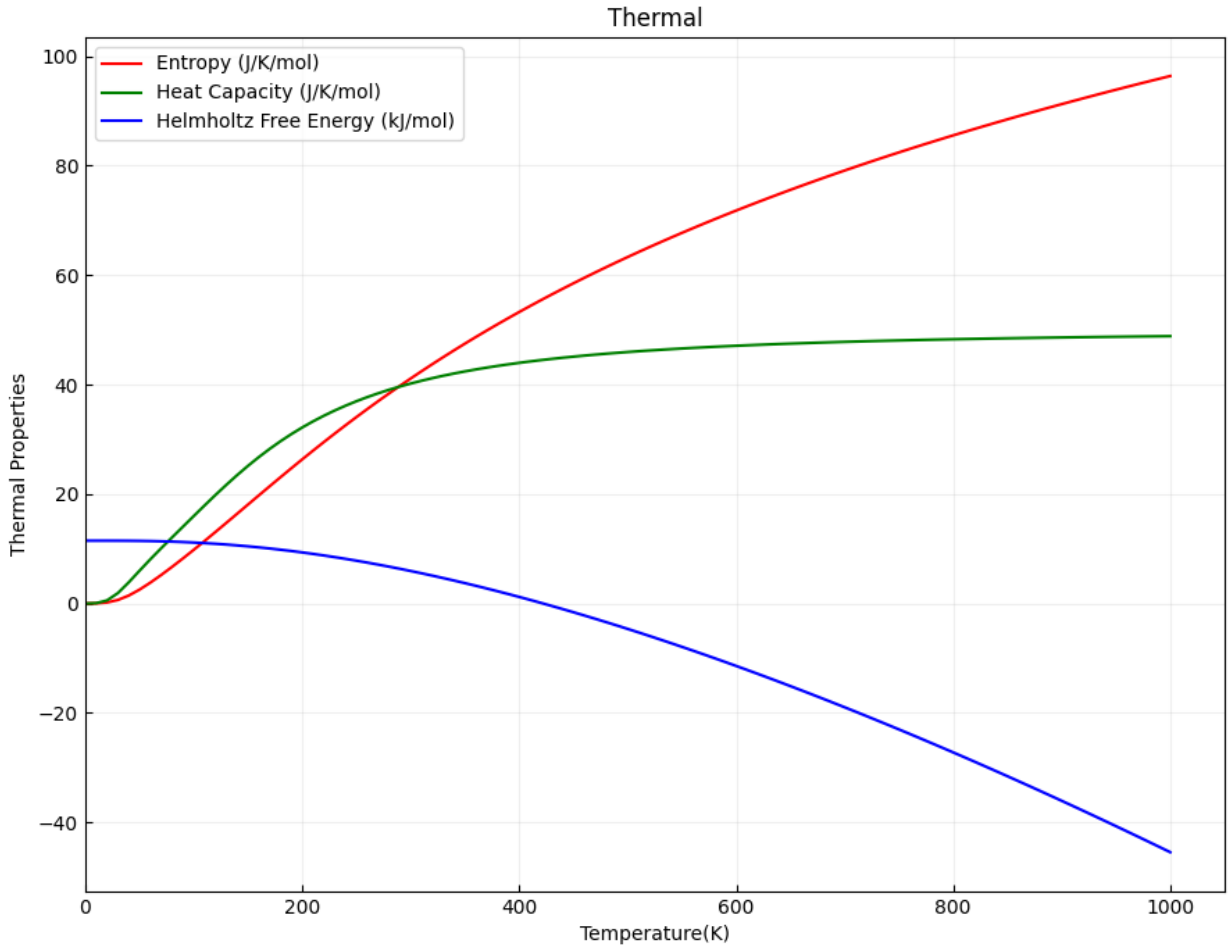
准备好输入文件之后，将 *phonon-thermal.in* 和 *structure.as* 文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 *DS-PAW phonon-thermal.in* 。

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.h5* 等输出文件。

- *DS-PAW.log*： DS-PAW 声子计算得到的日志文件；
- *phonon.h5*： DS-PAW 声子计算对应的 **h5** 输出文件，打开热力学计算的开关，生成的 *phonon.h5* 文件中会写入 **ThermalInfo** 数据，具体解析见输出文件格式说明 部分。

可使用 *phonon_thermal.py* 脚本对声子热力学数据进行处理，具体操作见辅助工具使用教程 部分。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：



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 = 650
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`：设置过渡态弛豫的维度，设置为 all 对应弛豫晶胞大小；
- `neb.method`：设置过渡态搜寻的方法，当 neb.freedom = all 时该参数可选值为 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
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
```

(续下页)

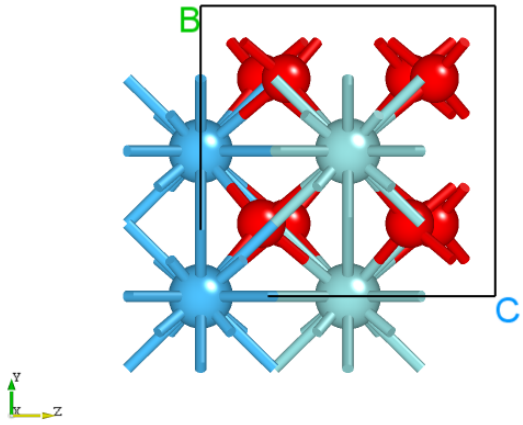
(接上页)

```
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
```

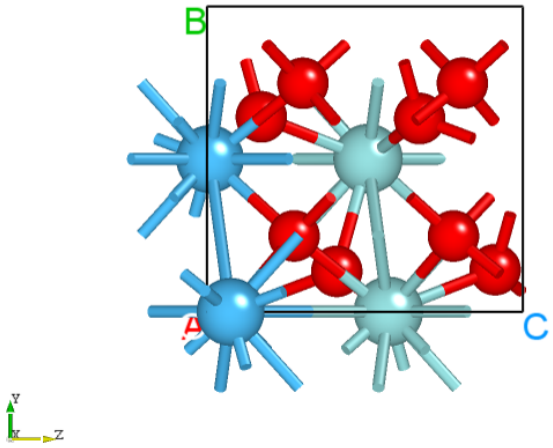
末态结构 *structure07.as* 参考如下

```
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

1. 在 `neb.freedom = all` 时，`neb.method` 可选值为 **QM2** 或 **FIRE**
2. **中间结构的生成可调用 “辅助工具使用教程-过渡态部分” 的 neb_interpolate_structures.py 脚本，完成插值可调用 neb_visualize.py 脚本对插值结构进行预览，可调用 calc_dist.py 脚本查看 image 之间的距离是否合理。**

2.27.2 run 程序运行

准备好输入文件之后，将 *ssneb.in* 和包含 *structureNo.as* 文件的多个文件夹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：

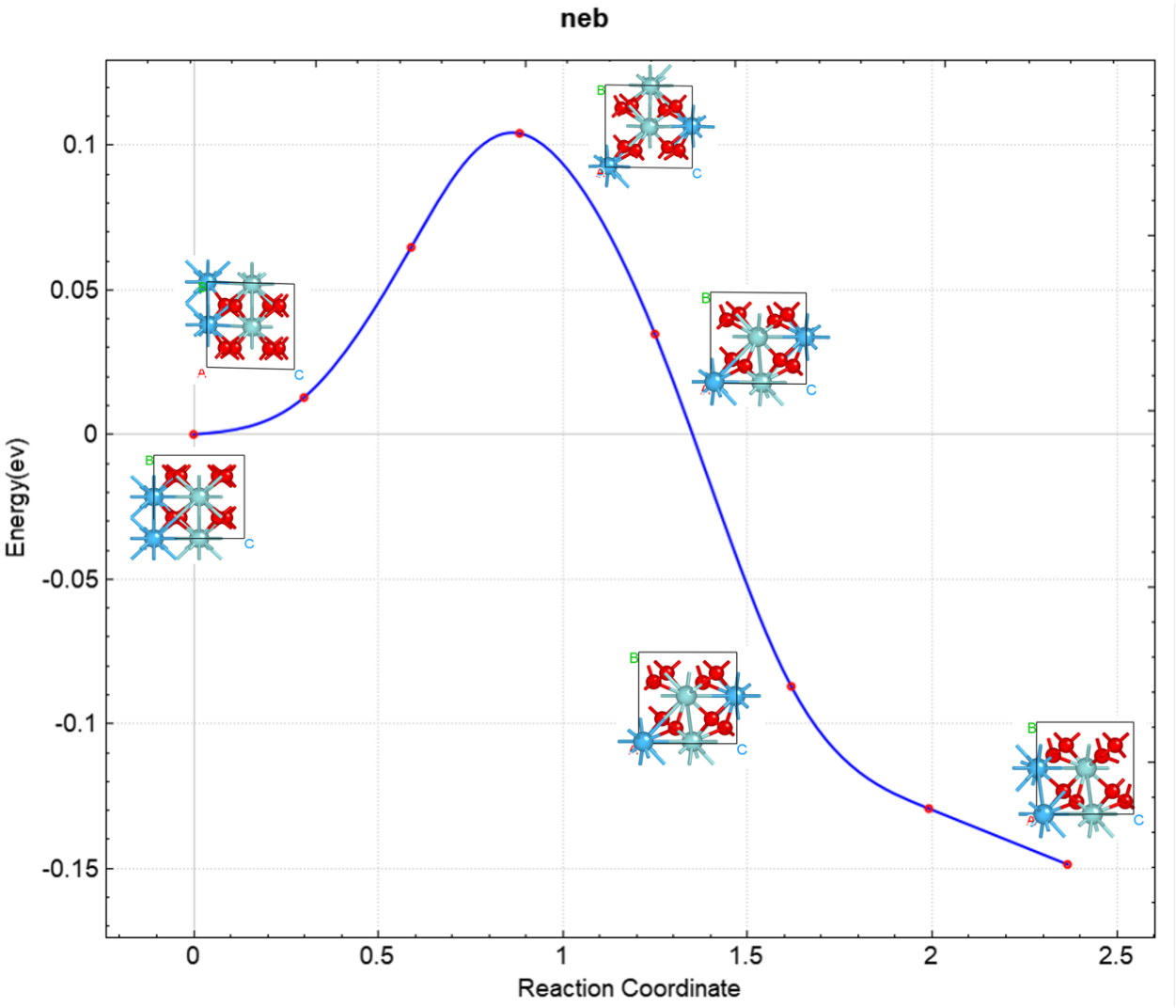
» 初态和末态结构所在文件夹会生成自洽计算所得的 *DS-PAW.log* 、*latestStructure00.as* 、*scf.h5* 等输出文件；

» 中间结构 *structureNo.as* 所在文件夹 `No` （参与过渡态计算的中间结构所在文件夹，中间结构的个数由 `neb.images` 参数决定）会生成结构优化所得的 *nebNo.h5* 、*latestStructureNo.as* 等输出文件；

» 最外层目录将会生成 *DS-PAW.log* 、*neb.h5* 这 2 个文件，其中 *neb.h5* 为 `No` 文件夹下各 *nebNo.h5* 文件的信息汇总。

- *DS-PAW.log*： DS-PAW 过渡态计算之后得到的日志文件；
- *neb.h5*： 过渡态计算完成之后的 **h5** 数据文件；此时反应坐标及能量变化等数据被保存在 *neb.h5* 中，具体的数据结构详见输出文件格式说明 部分；

可使用 **python** 脚本 *neb.py* 对 neb 计算的结果进行分析，需在完整的 neb 计算目录下执行分析脚本，具体操作见辅助工具使用教程 部分。处理得到的反应势垒曲线效果应如下所示：



2.28 solvation 溶剂化能计算

本节将以 H_2O 体系为例，介绍在 DS-PAW 中如何计算隐式溶剂模型下的溶剂化能。

2.28.1 H_2O 溶剂化能计算输入文件

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 = 3
cal.sigma = 0.2
cal.ksamping = G
cal.kpoints = [1, 1, 1]
cal.supGrid = true
cal.cutoff = 800
scf.convergence = 1.0e-6

#implicit solvation model
sys.sol = true
sys.solEpsilon = 80
sys.solTAU = 5.25E-4

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

scf.in 输入参数介绍：

- `sys.sol`：控制引入隐式溶剂化模型的开关，true 则考虑溶剂化效应；
- `sys.solEpsilon`：设置溶剂介电常数大小，此例设置为 80；
- `sys.solTAU`：指定单位面积的有效界面张力的大小，单位 eV/Å^2，默认值为 5.25E-4，该参数建议设置为小于 1e-3 的数值；
- `io.boundCharge`：控制溶剂束缚电荷密度文件输出的开关。

结构文件 *structure.as* 参考如下

```
Total number of atoms
3
Lattice
10.00000000 0.00000000 0.00000000
0.00000000 10.00000000 0.00000000
0.00000000 0.00000000 10.00000000
Cartesian
H 5.63934499 4.89541998 4.58224001
H 4.36065501 5.11499002 5.45934003
O 4.65002501 4.88500998 4.54065997
```

2.28.2 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* 。

2.28.3 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* 、*scf.h5* 、*rhoBound.h5* 等输出文件。

- *DS-PAW.log*： DS-PAW 隐式溶剂化模型计算之后得到的日志文件；
- *scf.h5*： 自洽计算对应的 **h5** 输出文件；
- *rhoBound.h5*： 使用隐性溶剂模型计算得到的溶剂束缚电荷密度文件；

此例计算可得到考虑溶剂化能的总能 E_{sol} ，溶剂化能计算公式如下：

Solvation Energy=E(sys.sol=true)-E(sys.sol=false)

根据该公式可知，需另进行 sys.sol = false 的计算以获得不考虑溶剂化能的总能 E_{nosol}，将 E_{nosol} 代入以上公式可得水的溶剂化能为 **-0.313** eV，与文献[5] 报道的结果一致。

使用隐式溶剂化模型进行计算时，可以同时得到溶质周围的溶剂束缚电荷密度分布文件 *rhoBound.h5* ，可使用 **python** 脚本 *trans_rho.py* 对该文件进行后处理，具体操作见辅助工具使用教程 部分。转换成的可视化文件在 VESTA 中打开，可以得到如下等密度面分布图：

从图中可以看到，溶剂化的正负屏蔽电荷密度分布在水分子的外围，形成一个溶解壳层，符合模型的预期，并与其它软件算得的溶剂束缚电荷密度分布类似。

[5] Kiran Mathew, Ravishankar Sundararaman, Kendra Letchworth-Weaver, TA Arias, and Richard G Hennig. Implicit solvation model for density-functional study of nanocrystal surfaces and reaction pathways. *The Journal of chemical physics*, 140(8):084106, 2014. doi:10.1063/1.4865107.

2.29 fixedpotential 固定电势计算

本节将以 $Cu-slab$ 体系为例，介绍如何在 DS-PAW 中进行固定电势计算。

2.29.1 $Cu-slab$ 固定电势计算输入文件

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

```
# task type
task = scf

sys.functional = PBE
sys.structure = structure.as

cal.ksamping = G
cal.cutoff = 650
cal.sigma = 0.2
cal.smearing = 3
cal.kpoints = [7,7,1]

scf.convergence = 1.0e-6
scf.max = 200

sys.sol = true
sys.solEpsilon = 78.4
sys.solLambdaD = 3.04
sys.solTAU = 0

# Potential fixed
sys.fixedP = true
sys.fixedPPotential = 2.155

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

fixedP.in 部分输入参数介绍：

- `task`： 设置计算类型，本例在 task=scf 时作固定电势计算；
- `sys.sol`： 打开溶剂化模型，固定电势计算需在隐式溶剂模型的基础上完成；
- `sys.solEpsilon`：设置溶剂介电常数大小，此例设置为 78.4；
- `sys.solLambdaD`：使用泊松玻尔兹曼方程且设置德拜长度（Debye length）的值，若不设置，则使用的是泊松方程，没有描述界面离子对静电势的贡献；
- `sys.solTAU`：指定单位面积的有效界面张力的大小，单位 eV/Å^2，默认值为 5.25E-4，该参数建议设置为小于 1e-3 的数值；
- `sys.fixedP`： 打开固定电势计算的开关；
- `sys.fixedPPotential`： 设置固定电势的电势值，默认以标准氢电极电势（SHE）作为参考电极电势，若以零电荷电位（Potential of Zero Charge, PZC）作为参考电极可设置参数 `sys.fixedPType = PZC`；

备注

1. 关于德拜长度 `sys.solLambdaD`, 其表达式为 $\lambda_D = \sqrt{\frac{\varepsilon\varepsilon_0 k_B T}{2c^0 z^2 q^2}}$

   1M 带 +/-1 电荷阴阳离子的水溶液的德拜长度为：3.04 Å

structure.as 文件参考如下：

```
Total number of atoms
8
Lattice
3.63404989 0.00000000 0.00000000
0.00000000 3.63404989 0.00000000
0.00000000 0.00000000 23.62132454
Cartesian
Cu 0.00000000 0.00000000 1.81702310
Cu 1.81702495 0.00000000 3.63404620
Cu 1.81702495 1.81702495 1.81702310
Cu 0.00000000 1.81702495 3.63404620
Cu 0.00000000 0.00000000 5.46390548
Cu 1.81702495 0.00000000 7.22885308
Cu 1.81702495 1.81702495 5.46390548
Cu 0.00000000 1.81702495 7.22885308
```

2.29.2 run 程序运行

准备好输入文件之后，将 *fixedP.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 fixedP.in* 。

2.29.3 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* 、*scf.h5* 等输出文件。

- *DS-PAW.log*： DS-PAW 完成固定电势计算之后得到的日志文件；
- *scf.h5*： task 等于 scf 时 DS-PAW 对应的 **h5** 输出文件；具体的数据结构详见输出文件格式说明 部分；

DS-PAW 采用最速下降法进行固定电势计算，计算过程中迭代体系的电荷数进行多次自洽计算。*DS-PAW.log* 文件会写入多次自洽计算的收敛过程，此例在 LOOP 5 达到收敛精度，如下展示 LOOP 5 结束体系对应的电势数值：

```
## FINISHED FIXEDPOTENTIAL LOOP 5 ##
Electron                      : 149.993000
ElectrodePotential_SHE        : 2.157747 V
ElectrodePotential_PZC        : 2.484286 V
ElectrodePotential_SHE(PZC)   : -0.326539 V
Chemical Potential(electron)  : -6.757747 eV
Grand Total Energy(sigma->0)  : -43088.518081 eV
```

其中

Electron 为迭代终点体系的电荷数;
ElectrodePotential_SHE 为迭代终点体系相对于标准氢电极电势的电势值;
ElectrodePotential_PZC 为迭代终点体系相对于 PZC 的电势值;

ElectrodePotential_SHE(PZC) 给出体系在中性条件 (即 PZC) 下（相对于 SHE）的电极电势值；
Chemical Potential(electron) 给出迭代终点体系电子化学势值（以隐式溶剂模型模拟的溶液深处电势为零点）；
Grand Total Energy(sigma->0) 给出迭代终点电子巨正则系综下的体系总能，与体系总能、电子数变化值、电子化学势值相关。

计算结束所得体系的电势值为 **2.157** V，与目标电势值 **2.155** V 接近。

备注

1. **进行固定电势计算时必须在隐式溶剂模型下进行，即 sys.fixedP = true 时，必须设置 sys.sol = true 。**
2. **目前，仅在 task = scf 时支持固定电势计算。**
3. **ElectrodePotential_SHE=ElectrodePotential_PZC+ElectrodePotential_SHE(PZC)**

2.30 wannier 插值能带计算

本节将以 Si 体系为例，介绍如何在 DS-PAW 中使用 wannier 函数进行插值能带计算。

2.30.1 Si 插值能带计算输入文件

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

```
# task type
task = wannier
sys.structure = structure.as
sys.symmetry = false
sys.functional = PBE
sys.spin = none
cal.methods = 1
cal.smearing = 1
cal.ksamping = G
cal.kpoints = [16,16,16]
cal.totalBands = 12

#wannier related
wannier.functions = 12
wannier.wannMaxIter = 20000
wannier.outStep = 50

#interpolated band related
wannier.interpolatedBand = true
wannier.kpointsLabel= [G,X,W,K,G,L]
wannier.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]
wannier.kpointsNumber = [100]

```

(续下页)

(接上页)

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

wannier.in 输入参数介绍：

- `task = wannier`：设置计算类型，新增可选值 wannier，用于进行 wannier 计算；
- `wannier.functions`：设置 wannier 函数的个数；
- `wannier.wannMaxIter`：设置求解最大局域化 wannier 函数过程中的总迭代次数；
- `wannier.outStep`：设置 task=wannier 时输出文件中输出迭代结果的步长；
- `wannier.interpolatedBand`：控制插值能带计算的开关；
- `wannier.kpointsLabel`：设置 wannier 函数拟合插值能带时高对称点符号；
- `wannier.kpointsCoord`：设置 wannier 函数拟合插值能带时高对称点坐标；
- `wannier.kpointsNumber`：设置插值能带高对称 K 点之间的撒点数；此例设置参数为 wannier.kpointsNumber= [100]，高对称点 G 与 X 之间撒点数为 100，以此求得撒点密度；对高对称点 X 与 W、W 与 K、K 与 G、G 与 L 之间进行等密度撒点，实际撒点数可从 DS-PAW.log 的参数打印部分获取；

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
```

备注

1. **初始投影的设置在 structure.as 文件中完成，首先在第 7 行添加 WannProj 标签，然后在原子坐标后写入初始投影轨道名称**，DS-PAW 可识别的投影轨道名称见参数说明 的 wannier 部分。
2. **此例未自定义初始投影，程序执行随机选择初始投影。若需定义初始投影，可参考下述写法。**

自定义初始投影轨道的 *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 WannProj
Si -0.125000000 -0.125000000 -0.125000000  [s,p,sp3-1,sp3-2]
Si 0.125000000 0.125000000 0.125000000     [s,p,sp3-3,sp3-4]
```

备注

1. **自定义初始投影轨道时，structure.as 文件中投影轨道总数需等于 wannier 函数的个数（wannier.functions），否则程序报错。**
2. **此例总投影轨道为 2*(1+3+1+1) = 12，与 wannier.functions = 12 参数设置一致。**
3. **对无需设置初始投影轨道的原子，在对应坐标后写入 [] 即可。**
4. **此例 cal.totalBands 设置为 12，因此提交计算时需调用 12 核。**

2.30.2 run 程序运行

准备好输入文件之后，将 *wannier.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 wannier.in* 。

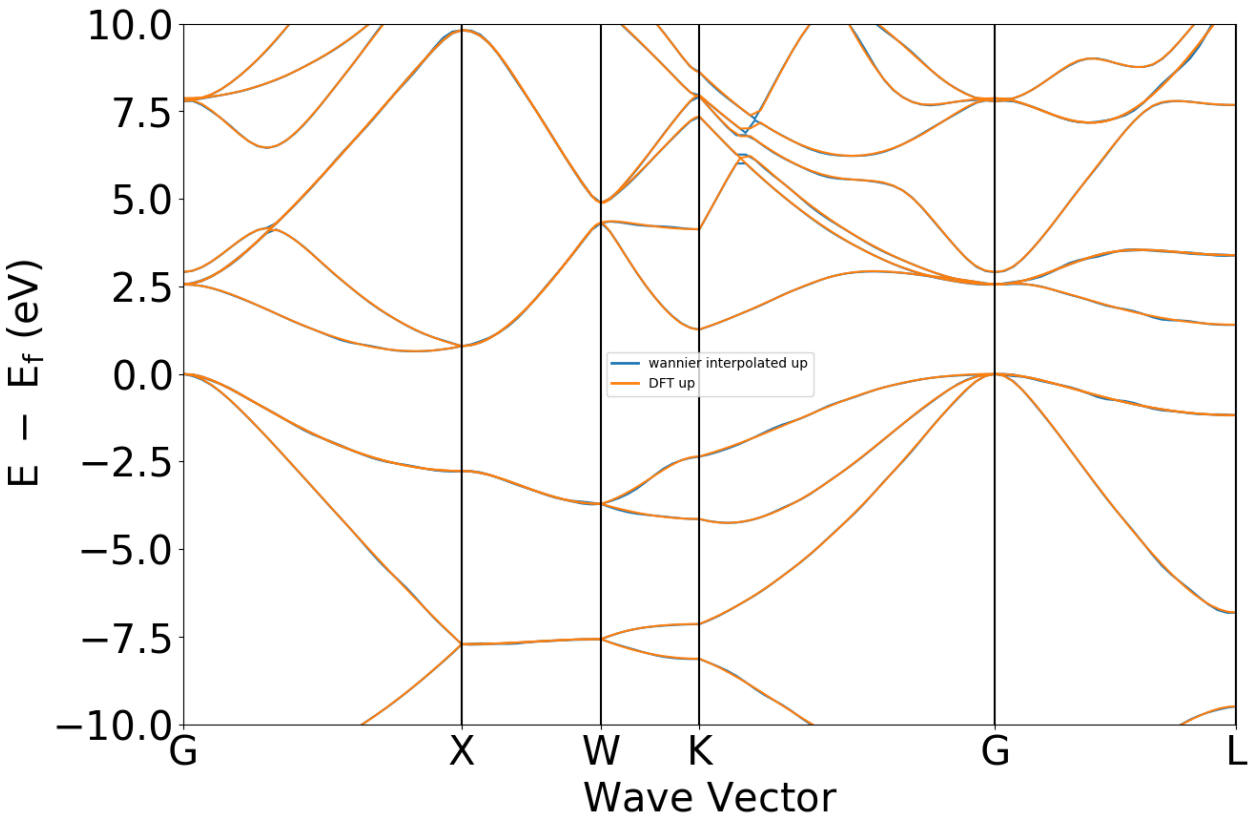
2.30.3 analysis 计算结果分析

根据上述的输入文件，计算完成之后将会得到 *DS-PAW.log* 、*wannier.h5* 等输出文件。

- *DS-PAW.log*： DS-PAW 完成 wannier 插值能带计算之后得到的日志文件；
- *wannier.h5*：wannier 函数插值能带计算对应的 **h5** 输出文件；具体的数据结构详见输出文件格式说明 部分；

可使用辅助工具使用教程 –> band 能带数据处理–> *bandplot.py* 脚本直接对 wannier 插值能带作图，读取 *wannier.h5* 即可。

也可使用 *bandcompare.py* 对 wannier 插值能带与 DFT 能带图对比，具体操作见辅助工具使用教程 部分。能带对比效果图应如下所示：



备注

1. **wannier 计算不支持打开 pob，且 dft 能带计算数（cal.totalBands）会随程序调用核数（cores）而改变，因此建议 wannier 计算调用 cores 与 cal.totalBands 参数保持一致。**
2. **当 wannier 函数的个数（wannier.functions）小于 cal.totalBands 时，wannier 函数最大局域化的过程需解纠缠，此时若用户未定义能量冻结窗口（wannier.disFrozWin），程序执行提取默认 Frozen Window 进行计算。**
3. **若自定义 Frozen Window，需确保 wannier.FrozWin 内所含能带的数量不能大于 wannier.functions 的数量，否则程序会报错 E4024。同时需确保窗口的合理性，否则无法得到好的拟合结果。**
4. **2023A 版本 DS-PAW 暂不支持 spin 类型为 non-collinear 的 wannier 计算。**

2.31 ref 参考文献

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，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，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重点关注：

- `sys.symmetry`： DS-PAW 可以通过对称性来减少程序的计算量，但同时可能会带来能量简并等不合理的结果，本次计算关闭对称性；
- `sys.spin`： 指定体系的磁性为 **collinear** 即 **共线自旋**；
- `cal.kpoints`： 对于没有周期性的维度，K 点可设置为 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
```

结构文件使用笛卡尔坐标，因此第七行中坐标类型为 `Cartesian`；为了使结构也尽量地减少对称性，将晶格修改为 **[7.5，8，8.9]** 格子。

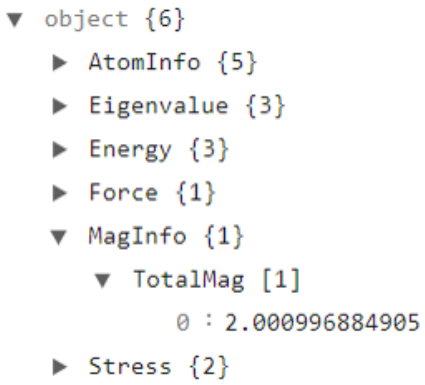
3.1.2 run 程序运行

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* 、*scf.h5* 等输出文件。

使用 HDFView 软件打开 *scf.h5* 文件，Eigenvalue 部分数据如下：



在 *scf.h5* 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，从 **MagInfo** 部分可得总磁矩为 **2μB** ，同时在 *DS-PAW.log* 中也可读取体系的总磁矩为 **2μB** 。

3.2 NiO 体系的反铁磁计算

本节将以 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

本次案例省略了结构弛豫过程，用户重现此案例时需先进行结构弛豫计算。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 = 650
```

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** 为反铁磁材料因此设置磁性为 `collinear` ；
- `cal.cutoff`： 设置平面波的截断为 **650 eV** ；

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
```

设置磁矩需在结构文件的第七行的 `Cartesian` 后加上 **Mag** 标签，在原子坐标所在行设置每个原子的磁矩，由于需要体现反铁磁（整个体系不显示磁矩，单个原子有磁矩），此例选取了 4 个原子的晶胞，设置 4 个 Ni 原子的磁矩分别为 **2**、**-2**、**0**、**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
```

(续下页)

(接上页)

```
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* 、*scf.h5* 等输出文件。从 *DS-PAW.log* 中可读取自洽完成之后总磁矩为 **1e-8μB** ，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* 、自洽计算所得的电荷密度文件 *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 = 650
dos.range = [-20, 20]
dos.resolution = 0.05
dos.project = true
```

pdos.in 输入参数介绍：

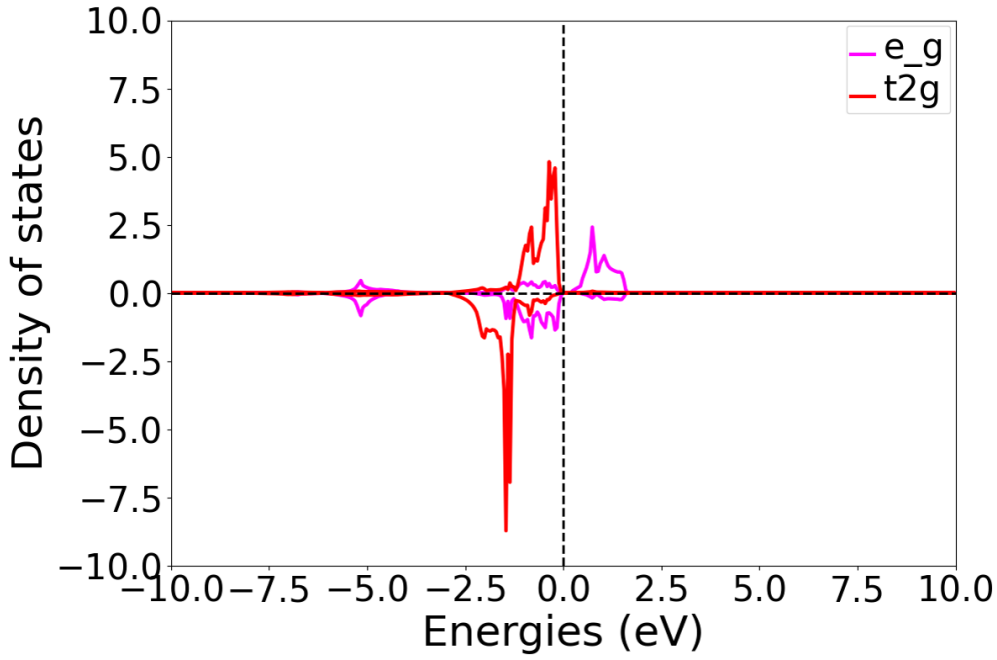
- `dos.range`： 表示能量计算区间为 -20 到 20 eV；
- `dos.resolution`： 表示在能量计算区间内打点的间隔精度；
- `dos.project`： 控制态密度的投影计算，本次计算打开了态密度的投影。

3.2.5 run 程序运行

将新建的 *pdos.in* 文件上传至服务器，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.h5* 等输出文件。使用辅助工具使用教程中相应脚本对 *dos.h5* 文件进行数据处理，分析第 2 个 Ni 原子的 t2g 和 eg 轨道，可得如下所示态密度分布图，此为不加 U 值的情况：



NiO 态密度

3.2.7 *NiO* 体系 DFT+U 的态密度计算

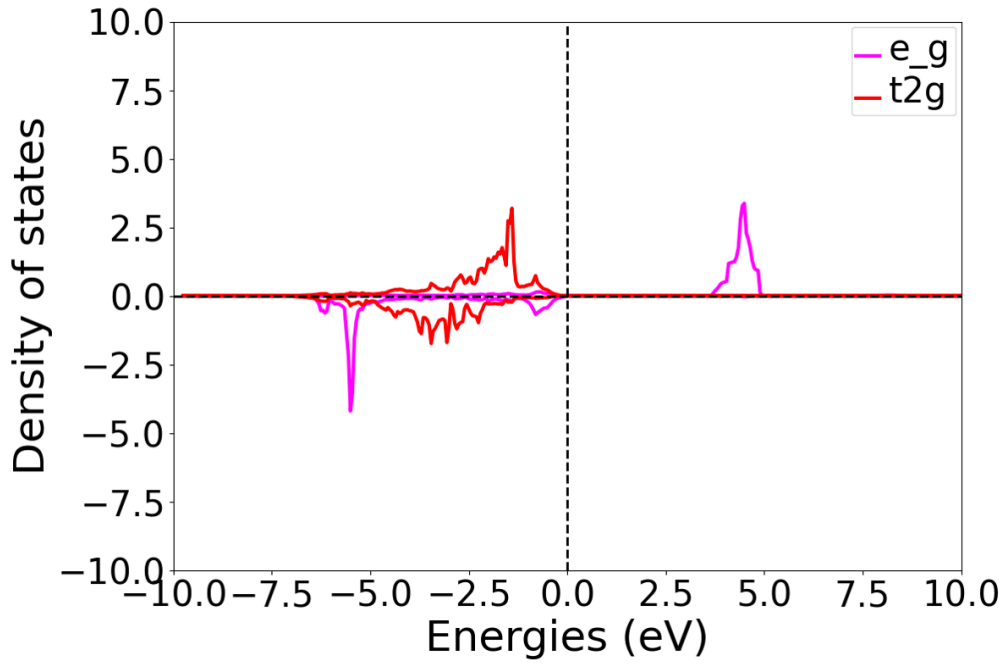
NiO 体系 DFT+U 的态密度计算的计算流程与上节 NiO 体系的态密度计算流程一致，不同之处在于需在自洽和态密度的计算中都加入 DFT+U 相关参数，需增加的输入参数如下所示：

```
#correction related
corr.dftu=true
corr.dftuForm = 1
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.dftuForm` 设置 DFT+U 方法，1 对应 DFT+U+J 方法 (Liechtenstein’ s formulation);
- `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，分析第 2 个 Ni 原子在进行 DFT+U 计算之后 t2g 和 eg 轨道态密度分布，得到的分布图如下所示：



NiO 态密度 (**DFT+U**)

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]。**
2. **DFT+U 默认使用方法为 DFT+U(Dudarev’s formulation)，对应参数 corr.dftuForm = 2，使用该方法时 J 值强制为 0，因此在该情况下设置 J 值无效。**

3.3 *AuAl* slab 模型功函数计算

本节将以 *AuAl* slab 模型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

本次案例省略了结构弛豫过程，用户重现此案例时需先进行结构弛豫计算。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 = 530

io.potential=true
potential.type = hartree

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

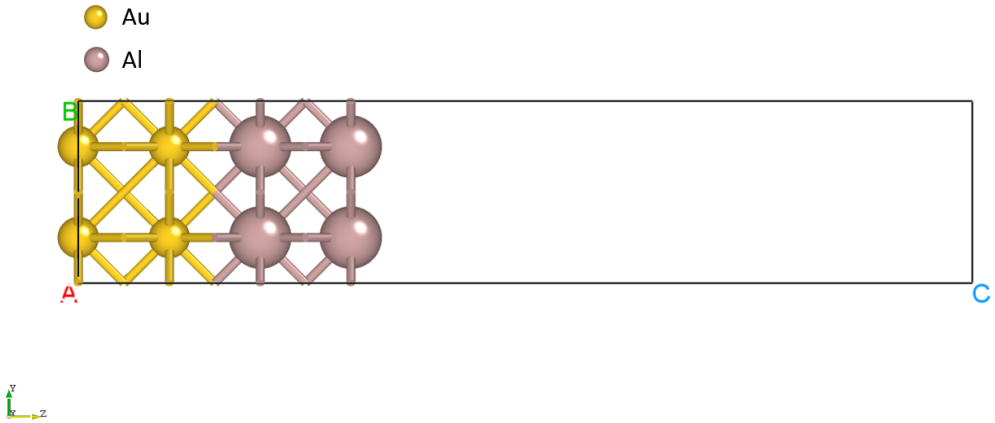
本次计算的输入文件中以下几个参数需要重点关注：

- `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
```

结构参考如下图：



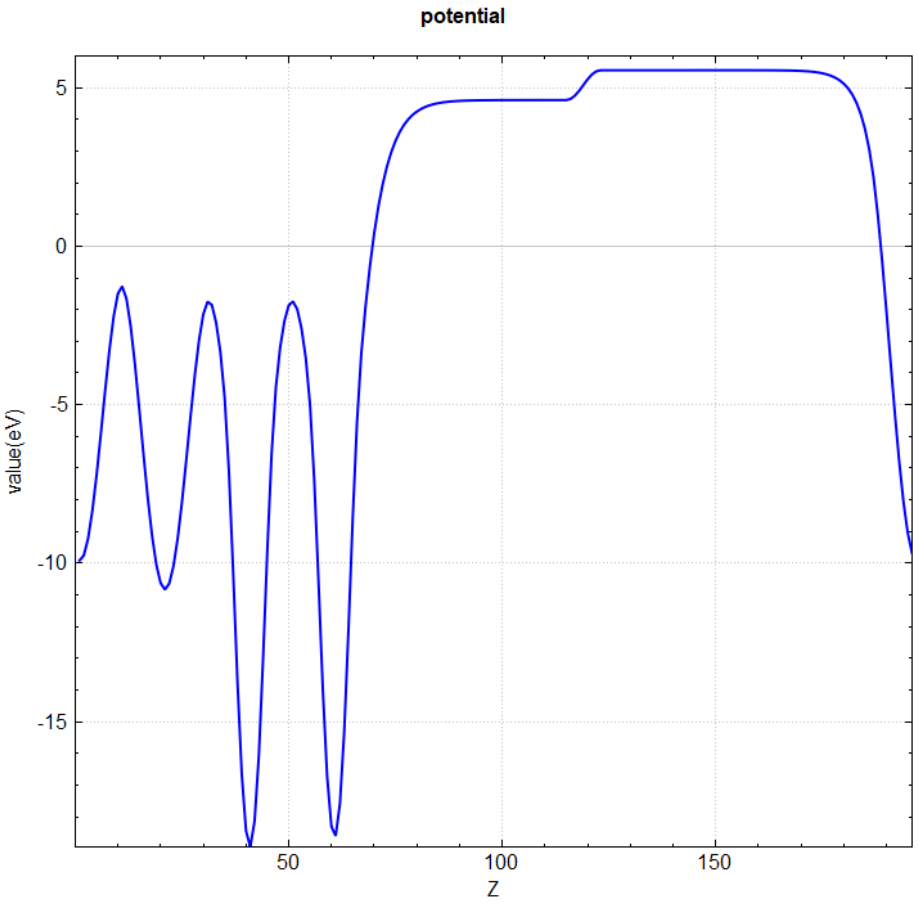
3.3.2 run 程序运行

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* 、*scf.h5* 等输出文件。对 *scf.h5* 文件进行数据处理可得功函数。

可使用 **python** 脚本分析 *scf.h5* 文件，将三维势函数进行面内平均，具体操作见辅助工具使用教程 部分。处理得到的真空方向势函数曲线如下所示：



AuAl 一维势函数图

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

从 *scf.h5* 中可读取费米能级为 0.113 eV

根据公式 $w = -e\phi - E_F$ 可得 *AuAl* slab 模型中，Au 的功函数为 **5.387** eV，Al 的功函数为 **4.487** eV。文献参考值[1]：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。

3.4 $Ru-N4$ 电催化氮还原反应计算

本节将展示如何使用 DS-PAW 模拟一个电催化还原氮气反应（Electrocatalytic nitrogen reduction reaction, eNRR）。该反应以碳基负载过渡金属 Ru 单原子为催化剂，使用 DS-PAW 对电催化氮气分子的吸附及还原过程进行模拟。

在电化学界面反应过程中，由于电化学反应界面通常与恒定电极电势的外电极相连，为确保电子的化学势与外电极的电势达到平衡，即电子的巨正则系综（grand canonical ensemble），实际体系中会存在电子的流入与流出过程。传统的第一性原理计算通常是在正则系综下，即在电荷守恒的条件下展开的，因此它并不能

[1] William M Haynes, David R Lide, and Thomas J Bruno. *CRC handbook of chemistry and physics*. CRC press, 2016. doi:10.1201/9781315380476.

很好的描述电化学界面反应，我们将在电荷守恒的条件下展开的计算模型称为恒电荷模型（constant charge model，CCM）。

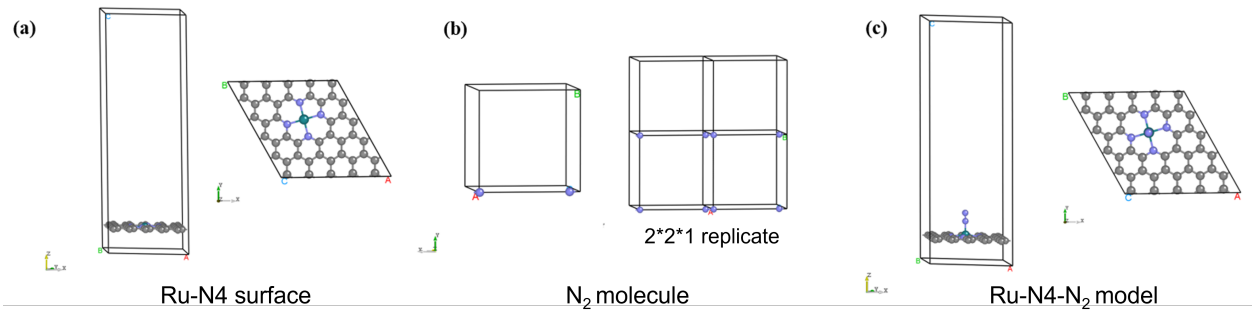
因恒电荷模型并不适于处理电化学界面问题，我们可采用在电子巨正则系综下展开第一性原理计算，这种计算方法又被称为固定电势方法（fixed potential method/constant potential method）。在此例中我们将利用固定电势计算的模型称为恒电势模型（constant potential model，CPM）。

3.4.1 Flow 计算流程及输入文件

此例以碳基负载过渡金属 Ru 单原子为催化剂，使用 DS-PAW 对电催化氮气分子的吸附及还原过程进行模拟。模拟的反应为氮气分子在碳基负载 Ru 单原子上的吸附过程，反应的表达式可简写作：$(Ru-N_4)+N_2=(Ru-N_4-N_2)$，在计算过程中使用了 CCM_vacuum，CCM_water，CPM_water 三种不同的模型，整个计算流程可大致分为四步，步骤展开如下：

3.4.1.1 Build 搭建模型

模型包括：(a) 碳基负载 Ru 原子模型 $(Ru-N_4)$，(b) N_2 单分子模型, (c) 吸附了 N_2 分子的碳基负载 Ru 原子模型 $(Ru-N_4-N_2)$，模型图如下所示：



3.4.1.2 Relaxation 结构弛豫

对搭建的结构进行结构弛豫，获得稳定结构，在 DS-PAW 中进行结构弛豫需要的核心参数：

```
relax.max =  200                    #指定结构弛豫时最大的离子步步数
relax.freedom = atom                #指定结构弛豫的自由度
relax.convergence =  0.02           #指定结构弛豫时原子受力的收敛判据
relax.methods = CG                  #指定结构弛豫的方法为共轭梯度法
```

3.4.1.3 Energy 能量计算

在不同模型条件下进行能量计算，获得稳定构型对应能量，下面按不同模型分别进行介绍：

CCM_vacuum

常规的真空层下的 scf 计算，即可获得 CCM_vacuum 模型下的能量，下面列出了在 DS-PAW 中进行 **单点能计算**需要设置的核心参数：

```
task = scf

#自洽计算相关
cal.methods = 1                    #指定自洽电子部分优化的方法为BD(block Davidson)方法
cal.smearing = 1                   #指定Gaussian smearing来设置每个波函数的部分占有数
cal.ksamping = G                   #指定生成布里渊区k点网格的方法
cal.kpoints = [2, 2, 1]            #指定布里渊区k点网格取样大小
cal.cutoffFactor = 1.0             #指定平面波基矢的截断能参数cal.cutoff的系数

corr.VDW = true                    #指定引入范德瓦尔斯修正计算
corr.VDWType = D2G                 #指定DFT-D2 of Grimme方法进行范德瓦尔斯修正
```

CCM_water

在 CCM 模型下，也可以利用隐式溶剂化模型来考虑溶剂效应，这里我们以水溶液为例，列出利用 DS-PAW 在 scf 计算中引入 **溶剂化模型**所需要设置的核心参数：

```
task = scf

#溶剂化模型相关
sys.sol = true                          #指定是否应用隐式溶剂化模型,true表示打开
sys.solEpsilon = 78.4                   #指定溶剂介电常数，水的介电常数78.4
sys.solLambdaD = 3.04                   #指定泊松玻尔兹曼方程中德拜长度，单位为Å
sys.solTAU = 0                          #指定单位面积的有效界面张力的大小，单位eV/Å^2
io.boundCharge = false                  #指定是否输出溶剂束缚电荷密度文件,false表示关闭
```

CPM_water

在 DS-PAW 中用固定电势方法计算即可获得 CPM 模型下的能量。在新发布的 2023A 版本中进行固定电势计算必须引入溶剂化模型，这里列出了利用 DS-PAW 在隐式水溶液环境下进行 **固定电势计算**的核心参数：

```
task = scf

#溶剂化模型相关
sys.sol = true                          #指定是否应用隐式溶剂化模型,true表示打开
sys.solEpsilon = 78.4                   #指定溶剂介电常数，默认值为水的介电常数78.4
sys.solLambdaD = 3.04                   #指定泊松玻尔兹曼方程中德拜长度，单位为Å
sys.solTAU = 0                          #指定单位面积的有效界面张力的大小，单位eV/Å^2
io.boundCharge = false                  #指定是否输出溶剂束缚电荷密度文件,false表示关闭

#固定电势设置相关
sys.fixedP = true                       #指定是否开启固定电势计算,true表示打开
sys.fixedPConvergence = 0.01
↪#指定固定电势计算的收敛精度，前后两次自洽计算的delta_electron小于收敛精度，计算结束
sys.fixedPMaxIter = 60                  #指定固定电势计算时最大迭代步数
sys.fixedPPotential = 0
↪#指定固定电势计算的目标电极电势值，默认以SHE为参考电极电势
sys.fixedPType = SHE                    #指定sys.
↪fixedPPotential给出的电势值的电势类型，支持SHE（标准氢电极）和PZC（零电荷电位）
```

3.4.1.4 ReactionEnergy 反应能计算

此例选取了 3 个不同的计算模型，首先介绍各模型下的吸附反应式：

CCM_vacuum

该模型下，吸附反应式可写作:

$(Ru-N4)+N2(idealgas)=(Ru-N4-N2)$

我们定义 ΔE 为反应能，反应能的计算表达式为：

$\Delta E=E0(Ru-N4-N2)-E0(Ru-N4)-E0(N2)$

其中，E0 对应真空模型下体系的总能（(sigma→0)，该数值可从自洽计算所得的 *scf.h5* 或 *system.json* 文件中获取，查找关键字“TotalEnergy0”即可。

CCM_water

该模型下，吸附反应式可写作:

$(Ru-N4)(inwater)+N2(idealgas)=(Ru-N4-N2)(inwater)$

$\Delta E=E0(Ru-N4-N2)-E0(Ru-N4)-E0(N2)$

其中，E0 对应水溶液浸润的模型下体系的总能（(sigma→0)，该数值可从自洽计算所得的 *scf.h5* 或 *system.json* 文件中获取，查找关键字“TotalEnergy0”即可。

CPM_water

该模型下模拟的反应过程为气相中的 N_2 在由水溶液浸润并与 0V vs. SHE(标准氢电极) 电极接触的催化剂表面的吸附过程，**此时吸附反应式有两种写法**，为便于描述，我们 **定义了以下物理量符号**：

- $ne0$: 中性体系下的核电荷数
- ne : 当体系电压为设定值 (sys.fixedPPotential 所设数值, 此例对应 0 V) 时体系的总电子数
- dne : 当体系电压为设定值时，体系的带电量：$dne=ne-ne0$
- μ_e : 体系电子化学势，电势零点为溶液深处（即 DFT 计算得到的电荷密度最低点的电势值）
- Δe : 吸附态体系价电子数 (eAB) 与吸附基底和吸附分子总价电子数 (eA+eB) 的差值
- $\Omega 0$: grand total energy(sigma→0): 电子巨正则系综下的体系总能，**其表达式为**：$\Omega 0 = E0-dne*\mu_e$

此时，CPM_water 模型下吸附反应式可参考如下写法：

方法一，在反应式中考虑 Δe ，吸附反应式可写作:

$Ru-N4(0Vvs.SHE)+N2(idealgas)=Ru-N4-N2(0Vvs.SHE)-\Delta e$

$\Delta E=E0(Ru-N4-N2)-\Delta e*\mu_e-E0(Ru-N4)-E0(N2)$

其中，E0 的数值可从自洽计算所得的 *scf.h5* 或 *system.json* 文件中获取，查找关键字“TotalEnergy0”即可。

ne 和 μ_e 的数值可从自洽计算所得的 *DS-PAW.log*（或 *scf.h5* 或 *system.json*）文件中获取，最后一个 LOOP 下查找关键字“Electron”和“Chemical Potential(electron) ”即可。

方法二，考虑电子巨正则系综下的体系总能 $\Omega 0$

由于固定电势计算是模拟的电子的巨正则系综，此时反应能计算式中的体系总能 $E0$ 应当用 $\Omega 0$ 来代替。吸附反应式可写作:

$(Ru-N4)(0Vvs.SHE)+N2(idealgas)=(Ru-N4-N2)(0Vvs.SHE)$

$\Delta E=\Omega 0(Ru-N4-N2)-\Omega 0(Ru-N4)-\Omega 0(N2)$

其中，$\Omega 0$ 的数值可从自洽计算所得的 *DS-PAW.log*（或 *scf.h5* 或 *system.json*）文件中获取，最后一个 LOOP 下查找关键字“Grand Total Energy”即可。

由于 (Ru-N4) 与 (Ru-N4-N2) 的电势为 0V vs. SHE，故对 (Ru-N4) 与 (Ru-N4-N2) 进行 0V 下的固定电势计算，从 DS-PAW 的相应输出文件提取数据，得到如下表格，能量单位为 eV：

system	$E0$	$nE0$	ne	dne	μ_e	$\Omega 0$
N2	-545.9393	10	/	/	/	-545.9393
Ru-N4	-10572.2452	212	211.224	-0.776	-4.60223	-10575.81654
Ru-N4-N2	-11119.6117	222	221.229	-0.771	-4.60054	-11123.15868

表 1. CPM_water 模型下固定电势计算数据

接下来将 **表 1** 的数据代入对应的表达式中进行计算：

方法一在反应式中考虑 Δe ，反应能计算过程如下:

$Ru-N4(0Vvs.SHE)+N2(idealgas)=Ru-N4-N2(0Vvs.SHE)-\Delta e$

$\Delta E=E0(Ru-N4-N2)-\Delta e*\mu_e-E0(Ru-N4)-E0(N2)$

$=-11119.6117-(221.229-211.224-10)*(-4.600)-(-10572.2452)-(-545.9393)$

$=-1.4042eV$

方法二考虑电子巨正则系综下的体系总能 $\Omega 0$ ，反应能计算过程如下:

$(Ru-N4)(0Vvs.SHE)+N2(idealgas)=(Ru-N4-N2)(0Vvs.SHE)$

$\Delta E=\Omega 0(Ru-N4-N2)-\Omega 0(Ru-N4)-\Omega 0(N2)$

$=-11123.1586-(-10575.8165)-(-545.9393)$

$=-1.4027eV$

通过两种方法计算所得的吸附能一致。可见用 DS-PAW 中定义的 $\Omega 0$ 即可很方便的计算固定电势下的反应能。

3.4.2 Run 程序运行

准备好输入文件之后，将结构弛豫计算、能量计算、固定电势计算的 in 文件和 *structure.as* 结构文件上传到安装了 DS-PAW 的环境上，按照流程分多步运行 *DS-PAW input.in* 命令或提交作业脚本完成多次计算。

3.4.3 ReactionEnergy 反应能数据分析

将表 1 的数据分别代入 **CCM_vacuum** 、**CCM_water** 、**CPM_water** 模型的吸附反应式，计算三个模型下，**eNRR 前三步反应的反应能**，结果如下 **表 2** 所示：

reaction/ Δe (eV)	CCM_vacuum	CCM_water	CPM_water
$(Ru-N4)+N2=(Ru-N4-N2)$	-1.3180	-1.3668	-1.4027
$(Ru-N4-N2)+0.5H2=(Ru-N4-N2-H)$	1.1355	1.0833	1.6511
$(Ru-N4-N2-H)+0.5H2=(Ru-N4-N2-H2)$	-0.6833	-0.8030	-1.0305

表 2. 反应能数据表

最后将上述结果绘制成反应坐标曲线，效果如下 **图 3** 所示：

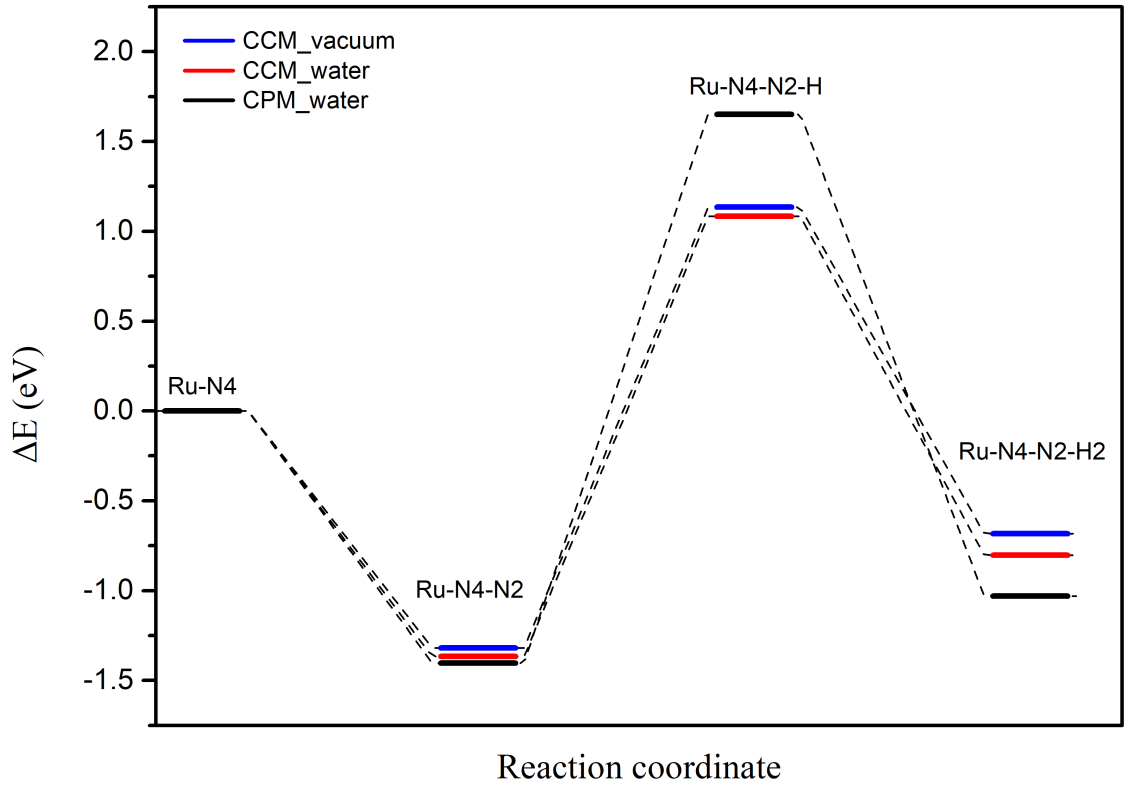


图 3. 反应坐标-反应能曲线

3.5 ref 参考文献

4

赝势说明

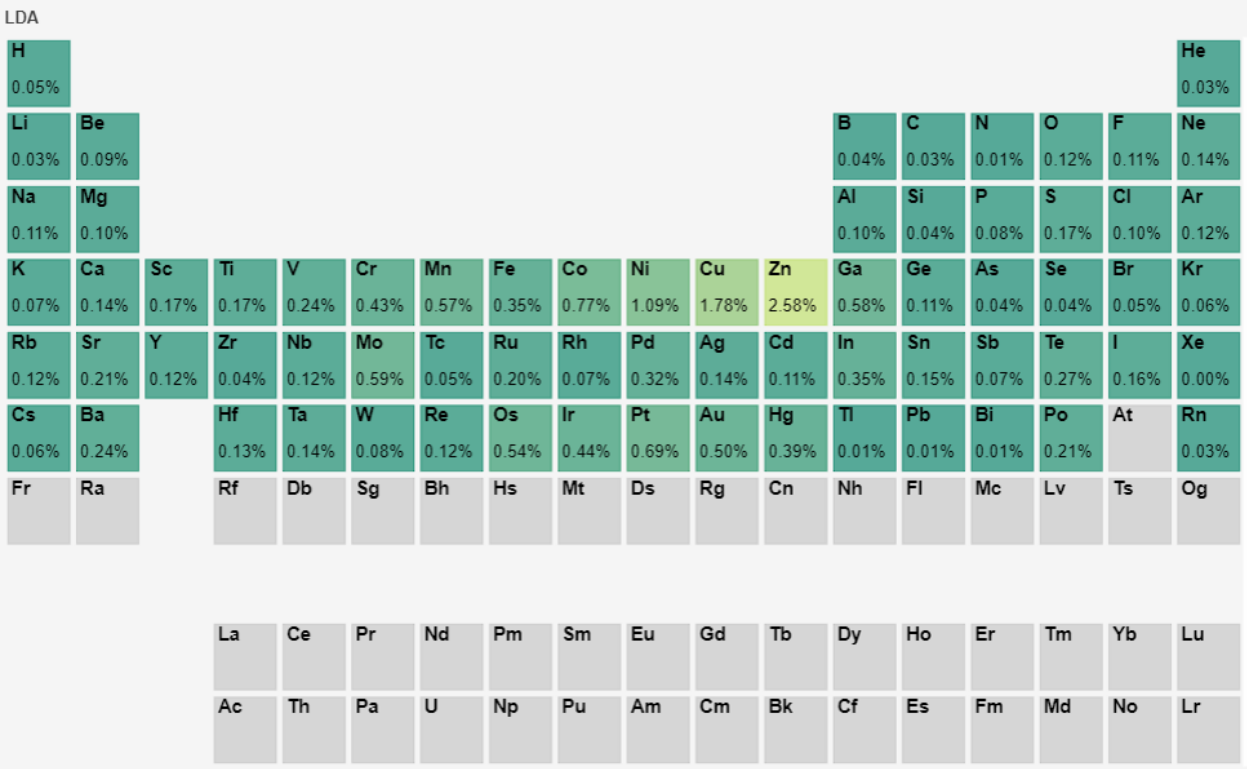
DS-PAW 目前可支持 .paw 、.potcar 、.pawpsp 3 种格式的赝势使用，用户可通过参数 sys.pseudoType 指定使用赝势的类型。

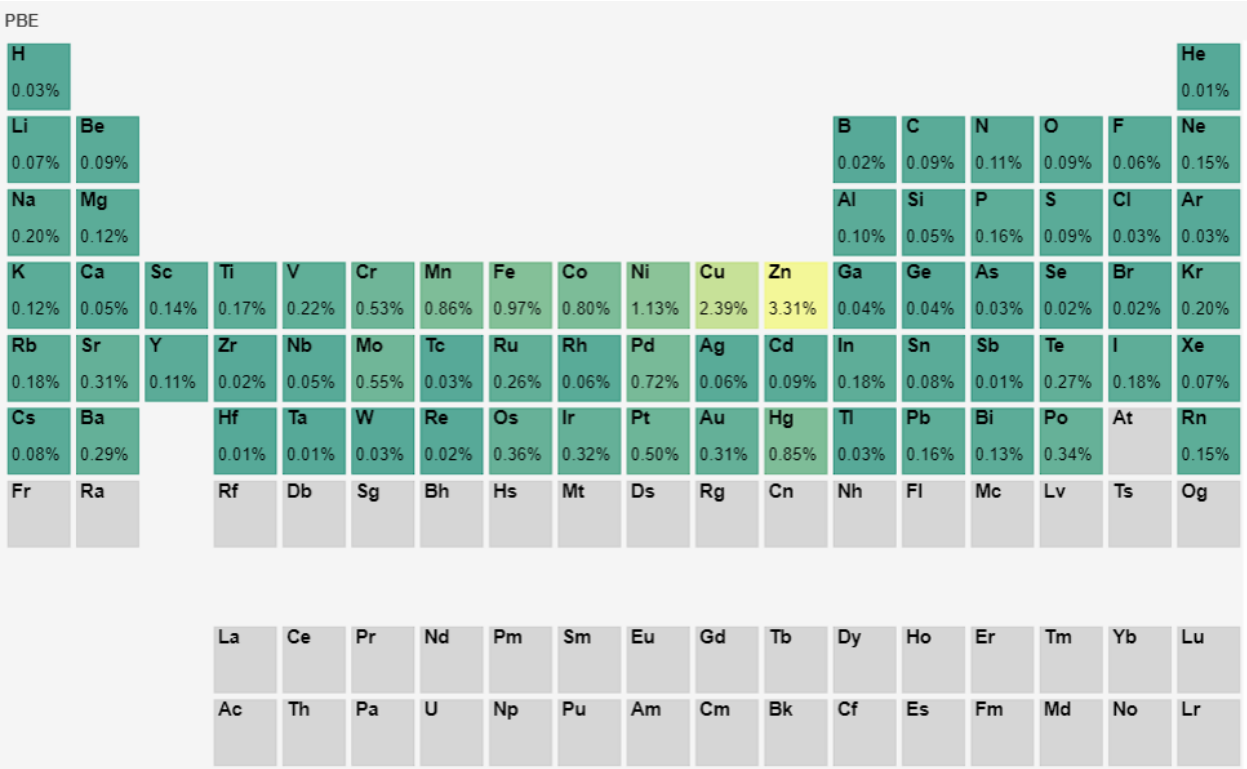
4.1 hzw 内部 paw 赝势

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，基于 1.0 版本赝势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，1.0 版本赝势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。

2024/12/31 发布 1.1 版本赝势，稍后将公布新版赝势性能表现，敬请期待…

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）。

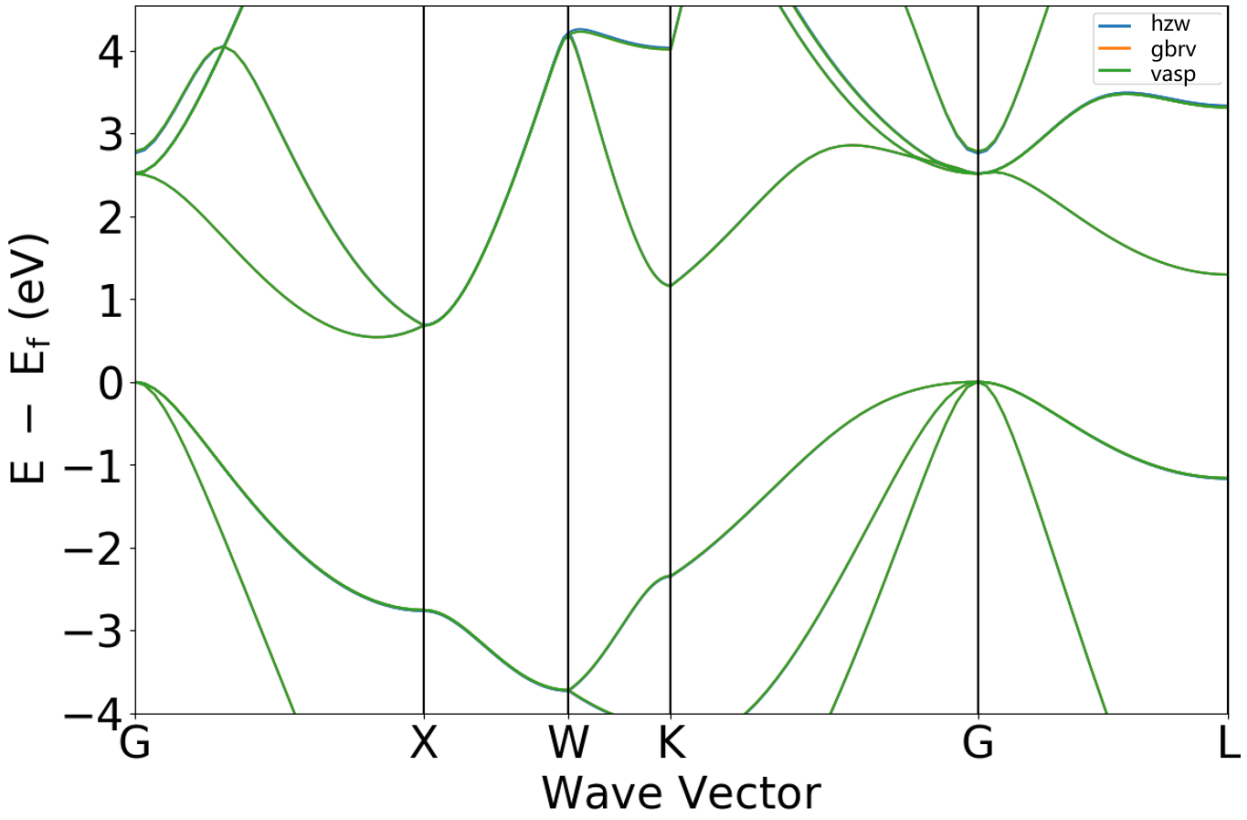
4.3 gbrv 赝势

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

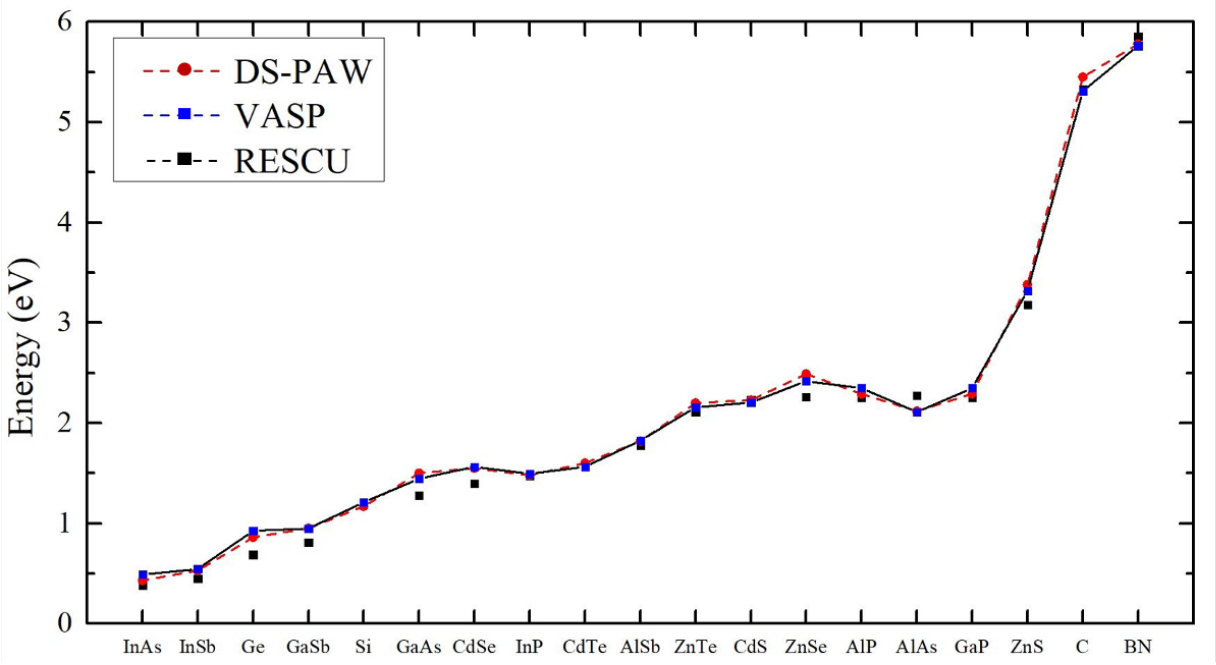
为对比三种赝势的计算结果，本节以 Si 体系为例，分别使用以上三种赝势进行了能带计算，下图为能带对比图，该对比图说明三种赝势在描述 Si 的能带时，体系出较高的一致性，验证了 hzw 赝势的准确性。



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 1.0 版本赝势对这些体系进行 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

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.spinDiff*
- *sys.soi*
- *sys.electron*
- *sys.hybrid*
- *sys.hybridType*
- *sys.hybridAlpha*
- *sys.hybridOmega*
- *sys.sol*
- *sys.solEpsilon*
- *sys.solTAU*
- *sys.solLambdaD*

- *sys.fixedP*
- *sys.fixedPConvergence*
- *sys.fixedPPotential*
- *sys.fixedPType*
- *sys.fixedPMaxIter*

- *cal.iniCharge*
- *cal.iniWave*
- *cal.cutoffFactor*
- *cal.cutoff*
- *cal.methods*
- *cal.smearing*
- *cal.sigma*
- *cal.kpoints*
- *cal.ksamping*
- *cal.totalBands*
- *cal.opticalGrid*
- *cal.iniFixedP*
- *cal.FFTGrid*
- *cal.supGrid*

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

- *scf.max*
- *scf.min*

- *scf.mixBeta*
- *scf.mixType*
- *scf.convergence*
- *scf.timeStep*

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

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

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

- *optical.grid*
- *optical.KKEta*
- *optical.smearing*
- *optical.sigma*
- *optical.Emax*

- *potential.type*

- *corr.chargedSystem*
- *corr.dipol*

- *corr.dipolDirection*
- *corr.dftu*
- *corr.dftuForm*
- *corr.dftuElements*
- *corr.dftuOrbital*
- *corr.dftuU*
- *corr.dftuJ*
- *corr.VDW*
- *corr.VDWType*
- *corr.dipolEfield*
- *corr.dipolPosition*
- *corr.coreEnergy*

- *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*
- *phonon.eigenVectors*

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

- *aimd.ensemble*
- *aimd.thermostat*
- *aimd.andersenProb*
- *aimd.noseMass*
- *aimd.latticeFCoeff*
- *aimd.atomFCoeffElements*
- *aimd.atomFCoeffs*
- *aimd.latticeMass*
- *aimd.pressure*
- *aimd.iniTemp*
- *aimd.finTemp*
- *aimd.timeStep*
- *aimd.totalSteps*

- *wannier.functions*
- *wannier.wannMaxIter*
- *wannier.disMaxIter*
- *wannier.disWin*

- *wannier.disFrozWin*
- *wannier.disEfShift*
- *wannier.interpolatedBand*
- *wannier.kpointsLabel*
- *wannier.kpointsCoord*
- *wannier.kpointsNumber*
- *wannier.kmeshTolerance*
- *wannier.outStep*
- *WannProj*

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/optical/potential/elf/pcharge/neb/frequency/phonon/elastic/aimd/epsilon/wannier`

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/optical/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**）；

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: .as /.h5 /.json

描述: `sys.structure` 参数设置结构文件的路径，支持 `.as` 、`.h5` 和 `.json` 格式，支持绝对路径和相对路径；使用 DS-PAW 进行结构弛豫默认生成 `relax.h5` 文件，可直接设置 `sys.structure = relax.h5` 读取弛豫后的结构进行计算；(.json 文件暂可支持但不建议用户使用，DS-PAW 会在迭代版本过程中完全摒弃 json 格式的输出)

案例: sys.structure = relax.h5

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-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.spinDiff*

默认值: 无

可选值: [0,inf)

描述： 设置上下自旋电子数的差；

案例: sys.spinDiff = 1

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，`sys.hybrid` 只在 task = scf 和 relax 的时候生效，`sys.hybrid` 设置为 true 时 `sys.functional` 不再生效；

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: *sys.sol*

默认值： false

可选值： false/true

描述： `sys.sol` 参数指定是否应用隐式溶剂化模型；

案例： sys.sol = true

参数名称: *sys.solEpsilon*

默认值： 78.4

可选值： real

描述： `sys.solEpsilon` 参数指定溶剂介电常数，默认值为水的介电常数；

案例： sys.solEpsilon = 80

参数名称: *sys.solTAU*

默认值： 5.25E-4

可选值： real

描述： `sys.solTAU` 参数指定单位面积的有效界面张力的大小，单位 eV/Å^2，该参数建议设置为小于 1e-3 的数值；

案例： sys.solTAU = 0

参数名称: *sys.solLambdaD*

默认值： 无

可选值： real

描述： `sys.solLambdaD` 参数指定泊松玻尔兹曼方程中德拜长度（Debye length）的数值，单位为 Å，若不设置，使用泊松方程且忽略双电层离子对静电势的屏蔽效应；

案例： sys.solLambdaD = 3.04

备注

1. 关于德拜长度 sys.solLambdaD, 其表达式为 $\lambda_D = \sqrt{\frac{\varepsilon\varepsilon_0 k_B T}{2c^0 z^2 q^2}}$

   1M 带 +/-1 电荷阴阳离子的水溶液的德拜长度为：3.04 Å

参数名称: *sys.fixedP*

默认值： false

可选值： false/true

描述： sys.fixedP 参数为控制固定电势计算的开关，目前只与 task = scf 兼容；

案例： sys.fixedP = true

参数名称: *sys.fixedPConvergence*

默认值： 0.01

可选值： real

描述： sys.fixedPConvergence 参数指定固定电势计算的收敛精度，当前后两次自洽计算的 delta_electron 小于收敛精度，计算结束；

案例： sys.fixedPConvergence = 0.01

参数名称: *sys.fixedPPotential*

默认值：无

可选值： real

描述： sys.fixedPPotential 参数指定固定电势计算的目标电极电势值，默认以标准氢电极（SHE）为参考电极电势；

案例： sys.fixedPPotential = 5.4723

参数名称: *sys.fixedPType*

默认值： SHE

可选值： SHE/PZC

描述： sys.fixedPType 参数指定 sys.fixedPPotential 给出的电势值的电势类型，SHE 为以标准氢电极的电极电势作为参考值，PZC 以零电荷电位为参考值；

案例： sys.fixedPType = SHE

参数名称: *sys.fixedPMaxIter*

默认值： 60

可选值： int

描述： sys.fixedPMaxIter 参数指定固定电势计算时最大迭代步数；

案例： sys.fixedPMaxIter = 100

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 = ../scf/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 = ../scf/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 480 eV。

案例: cal.cutoff = 480

参数名称: *cal.methods*

默认值: 1 (sys.hybrid = true 时，默认值为 4)

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，Monhkorst-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: *cal.opticalGrid*

默认值: 2000

可选值: int

描述: `cal.opticalGrid` 表示 DS-PAW 计算光学性质时在能量区内的网格点数，只在 io.optical 时生效；

案例: cal.opticalGrid = 2000

参数名称: *cal.iniFixedP*

默认值: 无

可选值: 指定恒电势计算输出的 h5 文件的路径

描述: `cal.iniFixedP` 给出上一次恒电势计算所得 h5 文件的路径，DS-PAW 读取该文件进行恒电势续算；

案例: cal.iniFixedP = ./scf.h5

参数名称: *cal.FFTGrid*

默认值: 取决于参数 cal.cutoff 和 cal.cutoffFactor

可选值: 3*1 int array

描述: `cal.FFTGrid` 给出晶胞在沿着三个晶格方向上的 FFT-Grid 格点数；

案例: cal.FFTGrid = [16,16,16]

参数名称: *cal.supGrid*

默认值: false

可选值: true/false

描述: `cal.supGrid` 为是否使用 support FFTGrid 的开关，可加大 FFT-Grid 的密度；

案例: cal.supGrid = true

参数名称: *io.charge*

默认值: true

可选值: true/false

描述: 控制是否输出电荷密度文件 `rho.bin` 和 `rho.h5`；当 io.charge=true 时，生成 `rho.bin` 和 `rho.h5` 文件；

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/relax 时该参数生效；不支持同时设置 sys.spin=non-collinear

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/relax 时该参数生效；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: task = wannier 且不读取 wave.bin 文件时默认值为 true, 其他 task 下默认值为 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；

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： 控制是否进行光学性质计算；io.optical=true 仅在 task=scf 时生效，当该参数生效时对应的 scf.h5 文件会写入光学性质数据；

案例: io.optical = true

参数名称: *io.bader*

默认值: false

可选值: false/true

描述: 控制是否进行 bader 电荷计算；io.bader=true 仅在 task=scf 时生效，当该参数生效时对应的 scf.h5 文件会写入 bader 电荷数据；

案例: io.bader = true

参数名称: *io.polarization*

默认值: false

可选值: false/true

描述: 控制是否进行铁电极化计算；io.polarization=true 仅在 task=scf 时生效，当该参数生效时对应的 scf.h5 文件会写入铁电极化数据；

案例: io.polarization = true

参数名称: *io.magProject*

默认值: sys.spin=collinear 以及 sys.spin=non-collinear 时默认值为 true，其余情况为 false

可选值: false/true

描述: 在磁矩计算中，控制是否在相应的 h5 输出文件中写入投影磁矩数据；

案例: io.magProject = true

参数名称: *io.boundCharge*

默认值: false

可选值: true/false

描述: 控制引入隐式溶剂模型时是否输出溶剂束缚电荷密度文件；

案例: io.boundCharge = true

参数名称: *io.outJsonFile*

默认值: true

可选值: true/false

描述: 控制是否输出 json 格式的输出文件；

案例: io.outJsonFile = false

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: Pulay

可选值: Broyden/Kerker/Pulay

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**、**Kerker 方法**和 **Pulay 方法**;

案例: scf.mixType = Pulay

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: *scf.timeStep*

默认值: 0.4

可选值: real

描述: `scf.timeStep` 参数控制 cal.methods=4/5 时的步长; 当 cal.methods =4 时, scf.timeStep 决定了 MD 的步长; 步长太小会让计算收敛花费较多步数, 步长太大会让 scf 计算发散; 当 cal.methods =5 时, scf.timeStep 决定了初始步长的大小，步长太大可能会让 scf 计算不稳定，步长太小可能会让计算不够准确;

案例: scf.timeStep = 0.4

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/volume/all/atom&shape

描述: `relax.freedom` 表示 DS-PAW 结构弛豫的自由度，atom 表示只弛豫原子位置; volume 表示只弛豫晶格体积；all 表示弛豫原子位置、晶格体积和晶胞形状;atom&shape 表示弛豫原子位置和晶格形状；

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 结构弛豫的方法，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` 表示弛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 结构弛豫时，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` 表示结构弛豫时，结构弛豫的中的缩放常数；

案例: relax.stepRange = 0.2

参数名称: *relax.pressure*

默认值: 0

可选值: real

描述: `relax.pressure` 表示将在特定外压下进行结构优化，也可用于修正 Pullay stress error，单位 kbar；

案例: relax.pressure = 100

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.h5` 文件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: *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 数组的大小是 band.kpointsCoord 数组大小的 **1/3** ；比 band.kpointsNumber 数组大小大 **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/ 1*1 int array

描述: 该参数只有在能带计算时生效；**band.kpointsNumber** 为每相邻两个高对称点之间的 K 点个数

- 当参数长度为 (n-1)*1 int array 时 **band.kpointsNumber** 比 **band.kpointsLabel** 数据大小少 **1**
- 当参数长度为 1*1 int array 时，以给定参数为基准，对所有高对称点进行等密度撒点，等密的最终撒点数可从 DS-PAW.log 的 **band.kpointsNumber** 读取；

案例: band.kpointsNumber = [100]

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, 若设置 band.project= ture , 则 `band.h5` 文件中会保存投影能带的信息；若不打开投影，设置 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 , `band.h5` 文件中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: *band.EfShift*

默认值: task=band 时为 true，其他 task 时为 false

可选值: true/false

描述： `band.EfShift` 参数表示 task=band 时，是否从 rho.bin 中读取 EFermi，仅在 task=band 时生效；

案例: band.EfShift = true

参数名称: *optical.grid*

默认值: 2000

可选值: int

描述： `optical.grid` 表示 DS-PAW 计算光学性质时在能量区内的网格点数，只在 io.optical 和 task=optical 时生效；

案例: optical.grid = 2000

参数名称: *optical.KKEta*

默认值: EnergyAxe resolution*0.99

可选值: real

描述： `optical.KKEta` 利用 Kramers-Kroning 关系求解介电函数实部时的 η 值。使用默认值时，结果可能很不平滑。增大 η 可以让结果更加平滑，但会让低频区域介电函数值计算结果出现一定程度错误。不建议使用过大的 η，建议通过增加格点数（optical.grid）让结果更平滑。（未增加此参数的旧版本中, η 值为 0.1）

案例: optical.KKEta = 0.1

参数名称: *optical.smearing*

默认值: 1

可选值: 1/2/3

描述： `optical.smearing` 决定在 optical 计算时对能量展宽时的展宽算法。1/2/3 分别对应 Gaussian smearing/Fermi smearing/Methfessel-Paxton order 1;

案例: optical.smearing = 1

参数名称: *optical.sigma*

默认值: 0.05

可选值: real

描述： `optical.sigma` 决定使用 optical.smearing 决定的展开算法时的展宽宽度；

案例: optical.sigma = 0.05

参数名称: *optical.Emax*

默认值: 未占据态的最大能量 *1.2 (eV)

可选值: real

描述： `optical.Emax` 决定 optical 计算时频率（EnergyAxe）的最大值；

案例: optical.Emax = 20

参数名称: *potential.type*

默认值: total

可选值: total/hartree/all

描述: `potential.type` 控制静电势的输出类型；当 potential.type = hartree，`potental.h5` 文件写入静电势（离子势和 hartree 势之和），当 potential.type = total，`potental.h5` 文件写入局域势（静电势和交换关联势之和）数据，当 potential.type = all，`potental.h5` 文件同时写入两种势；

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/all

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，all 表示所有方向，适用于孤立分子计算；

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.dftuForm*

默认值: 2

可选值: 1/2

描述: corr.dftuForm 表示选择哪一种 DFT+U 方法。1 对应 DFT+U+J 方法 (Liechtenstein' s formulation)，2 对应 DFT+U 方法 (Dudarev' s formulation)；

案例: corr.dftuForm = 2

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: *corr.coreEnergy*

默认值: false

可选值: true/false

描述: `corr.coreEnergy` 表示设置是否用初态近似计算芯电子能级;

案例: corr.coreEnergy = true

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，选用 LBFGS/CG/QM2/FIRE 方法时只能以力作为收敛判据；

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: real

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.h5 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/phonon.h5

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: *phonon.eigenVectors*

默认值: false

可选值: false/true

描述: `phonon.eigenVectors` 控制是否输出 dynamical matrix 的本征矢量。phonon.eigenVectors=true，将在 phonon 输出文件的 BandInfo 下增加 EigenVectors 的输出；其中 EigenVectors>Size 给出 dynamical matirx 本征矢量矩阵的大小（大小为：[NumberOfQPoints,(NumberOfAtoms*3),NumberOfBand,(real,imag)]），EigenVectors>RowMajor 表示是否以行优先模式输出，EigenVectors>Values 给出本征矢量矩阵的值；

案例: phonon.eigenVectors = true

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/NPT/NPH/SA

描述: `aimd.ensemble` 表示分子动力学模拟时选用的系综; SA 为 Simulated Annealing 缩写, 对应模拟退火过程;

案例: aimd.ensemble = NVE

参数名称: *aimd.thermostat*

默认值: 取决于 *aimd.ensemble*

可选值: andersen/noseHoover/langevin

描述: `aimd.thermostat` 表示分子动力学模拟时选用的恒温器或恒压器;

案例: aimd.thermostat = andersen

Thermostat/Ensemble	NVE	NVT	NPT	NPH	SA
andersen	compatible*	compatible	incompatible	incompatible	incompatible
noseHoover	incompatible	compatible*	incompatible	incompatible	incompatible
langevin	incompatible	compatible	compatible*	compatible*	incompatible

Note: * denotes default thermostat

参数名称: *aimd.andersenProb*

默认值: aimd.ensemble = NVE 时默认值为 0

可选值: NVE 时可选值为 0, NVT 时可选值为 real ($0 < x \leq 1$)

描述: `aimd.andersenProb` 控制 Andersen 恒温器下原子受到“碰撞”的概率;

案例: aimd.andersenProb = 0

参数名称: *aimd.noseMass*

默认值: 0

可选值: real ($x \geq 0$)

描述: `aimd.noseMass` 控制 Nose-Hoover 恒温器的有效质量;

案例: aimd.noseMass = 0

参数名称: *aimd.latticeFCoeff*

默认值: aimd.ensemble = NPH 时默认值为 0

可选值: NPH 时可选值为 0，NPT 时可选值为 real (x > 0)

描述: `aimd.latticeFCoeff` 表示 NPT/NPH 系综下 langevin 恒温器中的晶胞摩擦系数大小, 单位 ps-1；

案例: aimd.latticeFCoeff = 10

参数名称: *aimd.atomFCoeffElements*

默认值: 无

可选值: n*1 string array

描述: `aimd.atomFCoeffElements` 表示选用 langevin 恒温器时考虑为 langevin 原子的元素名称，命名规则为“原元素名 + 下划线 + 自定义字段”，如命名为 Hf_1，structure.as 文件需同步修改元素名；

案例: aimd.atomFCoeffElements = [Hf_1,O_1]

参数名称: *aimd.atomFCoeffs*

默认值: 无

可选值: n*1 string array

描述: `aimd.atomFCoeffs` 表示选用 langevin 恒温器时考虑为 langevin 原子对应的摩擦系数, 单位 ps-1。该数值应与 aimd.atomFCoeffElements 给出的元素名对应，如下案例表示为 Hf_1 原子赋值 10，为 O_1 原子赋值 5；

案例: aimd.atomFCoeffElements = [Hf_1,O_1], aimd.atomFCoeffs = [10,5]

参数名称: *aimd.latticeMass*

默认值: 1000

可选值: real

描述: `aimd.latticeMass` 表示选用 langevin 恒压器进行 NPT/NPH 模拟时晶胞自由度的虚拟质量，单位 **amu**；

案例: aimd.latticeMass = 1000

参数名称: *aimd.pressure*

默认值: 0

可选值: real

描述: `aimd.pressure` 表示进行 NPT/NPH 模拟时体系的目标压强值，单位 **kbar**；

案例: aimd.pressure = 1000

参数名称: *aimd.iniTemp*

默认值: 0

可选值: real

描述: `aimd.iniTemp` 表示分子动力学模拟时的初始温度，单位 K；

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，该参数只在 aimd.ensemble = SA 时生效，单位 K；

案例: aimd.finTemp = 1000

参数名称: *aimd.timeStep*

默认值: 1

可选值: real

描述: `aimd.timeStep` 表示分子动力学模拟时的时间步长，单位 fs；

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

参数名称: *wannier.functions*

默认值: 无

可选值: int

描述: `wannier.functions` 表示 wannier 函数的个数;

案例: wannier.functions = 8

参数名称: *wannier.wannMaxIter*

默认值: 200

可选值: int

描述: `wannier.wannMaxIter` 表示在求解最大局域化 wannier 函数过程中的总迭代次数;

案例: wannier.wannMaxIter = 500

参数名称: *wannier.disMaxIter*

默认值: 100

可选值: int

描述： `wannier.disMaxIter` 表示解纠缠的最大迭代步数；

案例: wannier.disMaxIter = 200

参数名称: *wannier.disWin*

默认值: [**自洽计算所得哈密顿量最低本征值, 最高本征值**]

可选值: 2*1 array

描述： `wannier.disWin` 表示解纠缠的能量窗口，默认包含所有能带；

案例: wannier.disWin = [-1000,1000]

参数名称: *wannier.disFrozWin*

默认值: 无

可选值: 2*1 array

描述： `wannier.disFrozWin` 表示解纠缠窗口，解纠缠时，该窗口中的态将保持不变；

案例: wannier.disFrozWin = [-10,10]

参数名称: *wannier.disEfShift*

默认值: false

可选值: true/false

描述： `wannier.disEfShift` 表示 wannier.disWin 和 wannier.disFrozWin 输入的能量是否是 Ef=0；

案例: wannier.disEfShift = true

参数名称: *wannier.interpolatedBand*

默认值: false

可选值: true/false

描述： `wannier.interpolatedBand` 表示 wannier 计算插值能带的开关；

案例: wannier.interpolatedBand = true

参数名称: *wannier.kpointsLabel*

默认值: 无

可选值: n*1 string array

描述： `wannier.kpointsLabel` 表示插值能带的高对称点标签；

案例: wannier.kpointsLabel = [G,M,K,G]

参数名称: *wannier.kpointsCoord*

默认值: 无

可选值: 3n*1 real array

描述: `wannier.kpointsCoord` 表示插值能带高对称点的分数坐标;

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

参数名称: *wannier.kpointsNumber*

默认值: 无

可选值: (n-1)*1 int array/ 1*1 int array

描述: 该参数只有在插值能带计算时生效; **wannier.kpointsNumber** 为能带每相邻两个高对称点间的 K 点个数

- 当参数长度为 (n-1)*1 int array 时 **wannier.kpointsNumber** 比 **wannier.kpointsNumber** 数据大小少 **1**
- 当参数长度为 1*1 int array 时, 以给定参数为基准, 对所有高对称点进行等密度撒点, 等密的最终撒点数可从 DS-PAW.log 的 **wannier.kpointsNumber** 读取;

案例: wannier.kpointsNumber = [100]

参数名称: *wannier.kmeshTolerance*

默认值: 1e-06

可选值: real

描述: `wannier.kmeshTolerance` 决定两个 k 点是否在同一壳层;

案例: wannier.kmeshTolerance = 1e-06

参数名称: *wannier.outStep*

默认值: 20

可选值: int

描述: `wannier.outStep` 表示 task=wannier 时输出 wannier 的信息的步数间隔;

案例: wannier.outStep = 50

标签名称: *WannProj*

默认值: 无

可选值: n*1 string array

描述: `WannProj` 为 wannier 计算中定义初始投影轨道的标签, 用于 structure.as 中;

案例:

```
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 WannProj
Si -0.125000000 -0.125000000 -0.125000000  [s,p]
Si 0.125000000 0.125000000 0.125000000     [s,p]
```

备注

1. **WannProj** 标签设置在 structure.as 文件的第 7 行
2. 此例投影轨道总数为 2*（1+3）= 8 条

可选值范围: DS-PAW 共支持 **44** 种投影轨道名称，分二类，展示如下：

- 第一类：轨道简称，对应该类型轨道的总数目，两种关系见下表：

name	number of projections
[s]	1
[p]	3
[d]	5
[f]	7
[sp]	2
[sp2]	3
[sp3]	4
[sp3d]	5
[sp3d2]	6

- 第二类：特定轨道的名称，每种（每个数组，[ ]）对应 **1** 条投影轨道：

[px] [py] [pz]
[dxy] [dyz] [dxz] [dz2] [dx2-y2]
[fz3] [fxz2] [fyz2] [fxyz] [fz(x2-y2)] [fx(x2-3y2)] [fy(3x2-y2)]
[sp-1] [sp-2]
[sp2-1] [sp2-2] [sp2-3]
[sp3-1] [sp3-2] [sp3-3] [sp3-4]
[sp3d-1] [sp3d-2] [sp3d-3] [sp3d-4] [sp3d-5]
[sp3d2-1] [sp3d2-2] [sp3d2-3] [sp3d2-4] [sp3d2-5] [sp3d2-6]

备注

1. 当不定义初始轨道时（见快速入门 2.30 节），程序执行随机选择初始投影。

6

输出文件格式说明

DS-PAW 2023A 版本默认生成的输出文件 **JSON** 文件可直接通过 Device Studio 进行分析处理，另输出文件新增了 **hdf5** 格式, 可下载 vitables （python 环境下执行 pip install vitables）或 HDFView 查看 **hdf5** 格式文件，使用辅助工具使用教程 提供的 **python** 脚本进行结果分析。

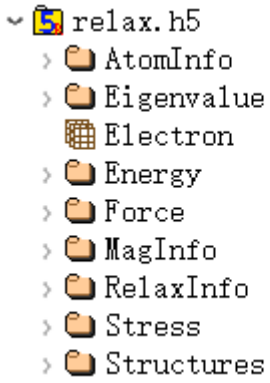
除电荷密度文件 *rho.h5* 、溶剂束缚电荷密度输出文件 *rhoBound.h5* 、其余输出文件的文件名取决于 **task** 类型，现 DS-PAW 支持 task 类型有 14 种，对应的 h5 文件名分别为：*relax.h5* 、*scf.h5* 、*band.h5* 、*dos.h5* 、*potential.h5* 、*elf.h5* 、*pcharge.h5*、*frequency.h5* 、*elastic.h5* 、*neb.h5* 、*phonon.h5* 、*aimd.h5* 、*epsilon.h5* 、*wannier.h5* 。

DS-PAW 2023A 暂可支持 **.json** 格式的输出文件，但不建议用户继续使用该格式分析结果，DS-PAW 会在迭代版本过程中完全摒弃 json 格式的输出，停止对该种格式的维护与升级。用户可通过 `io.outJsonFile` 参数控制 json 文件是否输出。

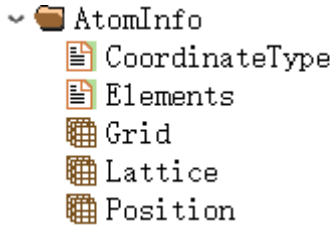
6.1 relax.h5

relax.h5 为 *task* = *relax* 时的输出文件，当 *task* 类型为其他时，该文件不输出。

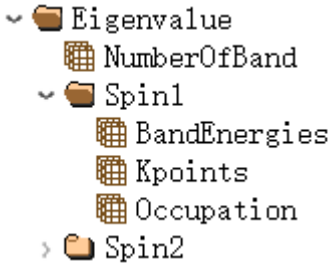
relax.h5 至少包含 9 个基本结构体：



（1）*AtomInfo* 中保存体系的基本结构信息，如晶胞大小，原子位置等；

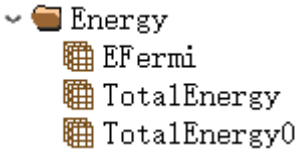


（2）*Eigenvalue* 中保存能带计算数量、自旋信息、k 点数量及坐标、各能带在各 k 点下对应的轨道占据数及能量本征值;

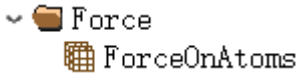


（3）*Electron* 中保存体系总的价电子数;

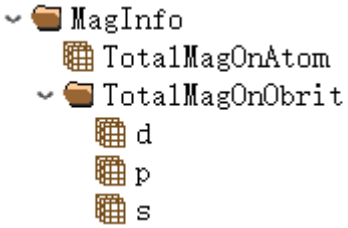
（4）*Energy* 中保存总能及费米能;



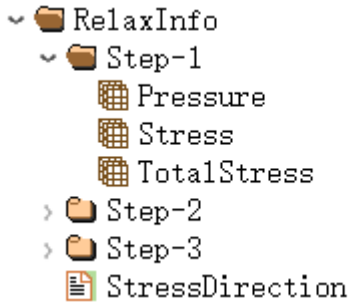
（5）*Force* 中保存弛豫过程每个原子的受力;



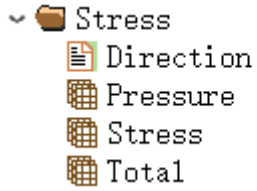
（6）*MagInfo* 中保存原子总的磁矩信息，若打开了投影则保存投影磁矩信息;



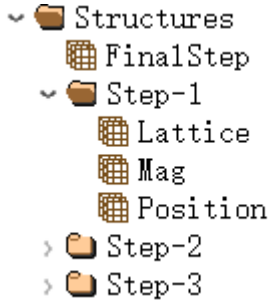
（7）*RelaxInfo* 中保存体系在结构弛豫过程中每步的应力及压力数据;



（8）*Stress* 中保存晶胞各方向的应力大小，体系压力大小;



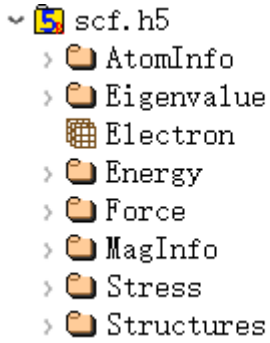
（9）*Structures* 中保存弛豫过程中结构和磁矩数据;



6.2 scf.h5

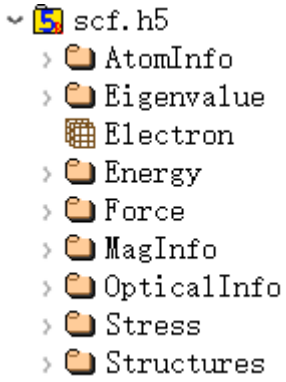
scf.h5 为 *task* = *scf* 时的输出文件，当 *task* 类型为其他时，该文件不输出。

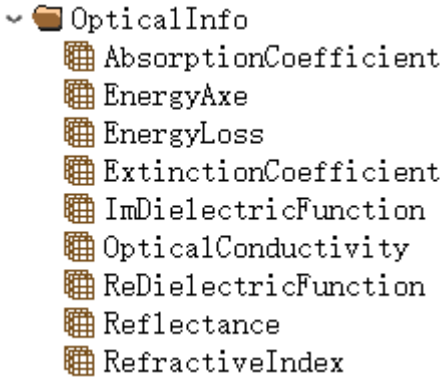
scf.h5 至少包含 8 个基本结构体，其基本信息与 *relax.h5* 一致：

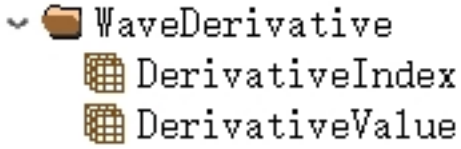


task = *scf* 下可通过 `sys` 和 `io` 等参数控制完成多种功能的计算，不同功能下生成的 *scf.h5* 文件会写入对应功能的数据，具体可分为以下情况：

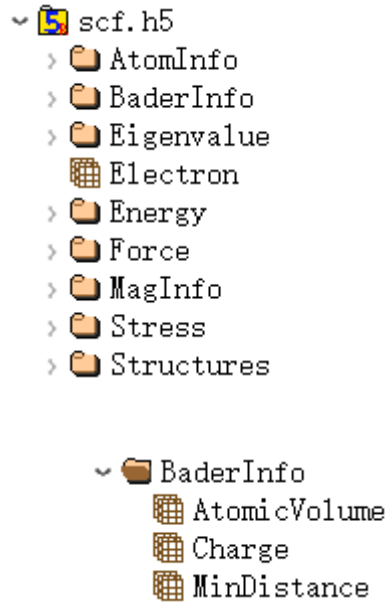
（1）通过设置 `io.optical = true` 在自洽计算的基础上计算线性光学性质：



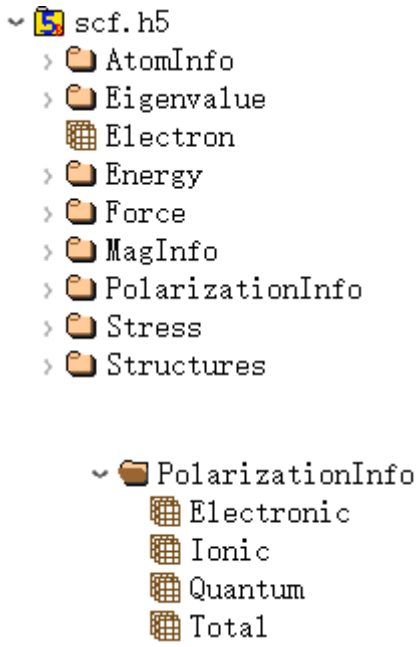




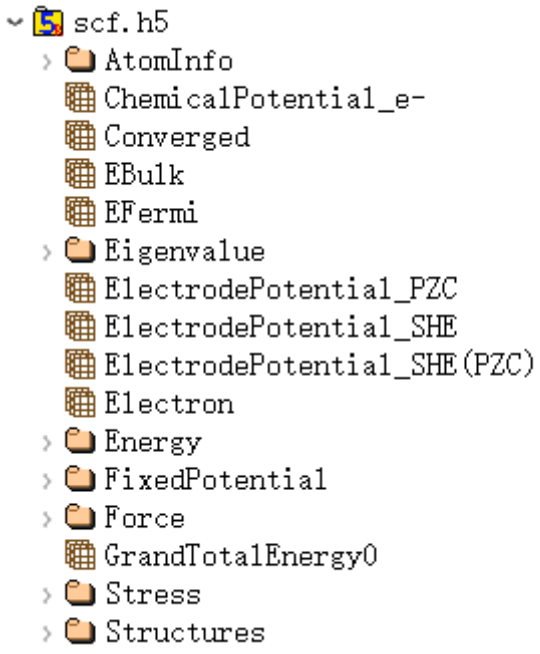
（2）通过设置 `io.bader = true` 在自洽计算的基础上计算 bader 电荷：



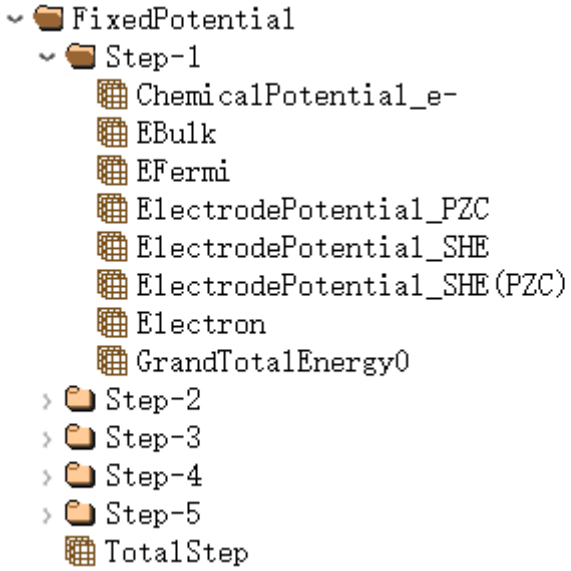
(3) 通过设置 `io.polarization = true` 在自洽计算的基础上进行铁电计算:



(4) 通过设置 `sys.fixedP = true` 在自洽计算中进行固定电势计算:

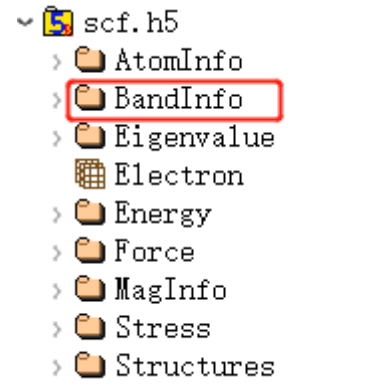


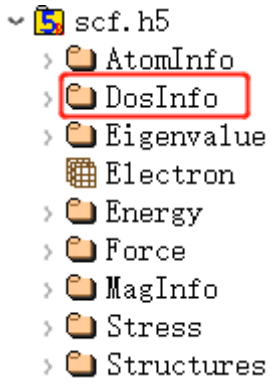
其中 *ChemicalPotential_e* 为体系电子化学势值；*EBulk* 取值为“隐式溶剂模型下费米能级 shift 值 *-1”；*ElectrodePotential* 给出不同标定标准下的电势值；*GrandTotalEnergy0* 给出电子巨正则系综下的体系总能。

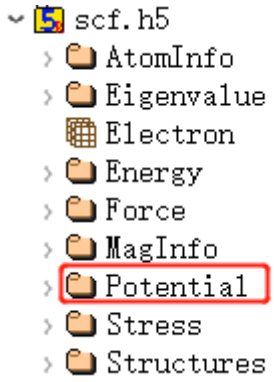


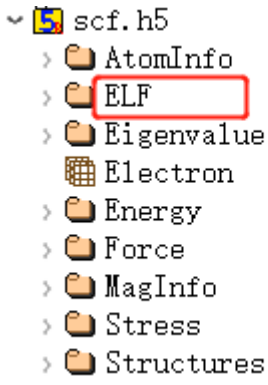
fixedPPotential 标签下展开为电子迭代过程中各重要参数的信息汇总。

(5) **通过设置 `io.band = true`、`io.dos = true`、`io.potential = true`、`io.elf = true`，** 在自洽计算的基础上进行能带计算、态密度计算、势函数计算、电子局域密度计算：


scf.h5
AtomInfo
BandInfo
Eigenvalue
Electron
Energy
Force
MagInfo
Stress
Structures


scf.h5
AtomInfo
DosInfo
Eigenvalue
Electron
Energy
Force
MagInfo
Stress
Structures

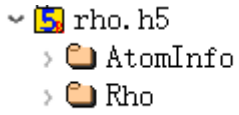

scf.h5
AtomInfo
Eigenvalue
Electron
Energy
Force
MagInfo
Potential
Stress
Structures


scf.h5
AtomInfo
ELF
Eigenvalue
Electron
Energy
Force
MagInfo
Stress
Structures

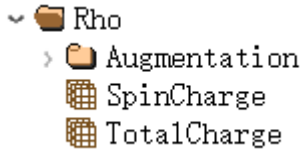
6.3 rho.h5

rho.h5 为各 *task* 下的电荷密度输出文件。

rho.h5 包含 2 个结构体：



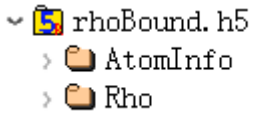
其中 *AtomInfo* 与 *relax.h5* 文件的 *AtomInfo* 结构一致，*Rho* 中保存电荷密度数据：



6.4 rhoBound.h5

rhoBound.h5 为溶剂化模型下的溶剂束缚电荷密度输出文件。

rhoBound.h5 包含 2 个结构体：

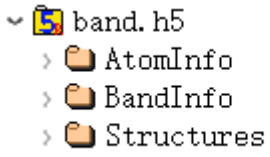


其中 *AtomInfo* 与 *relax.h5* 文件的 *AtomInfo* 结构基本一致，*Rho* 中保存溶剂束缚电荷密度数据：

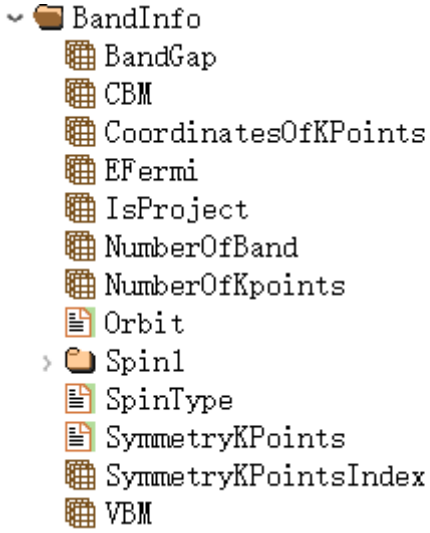
6.5 band.h5

band.h5 为各 *task = band* 下的输出文件，当 *task* 类型为其他时，该文件不输出。

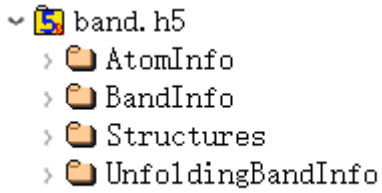
band.h5 至少包含 3 个结构体：



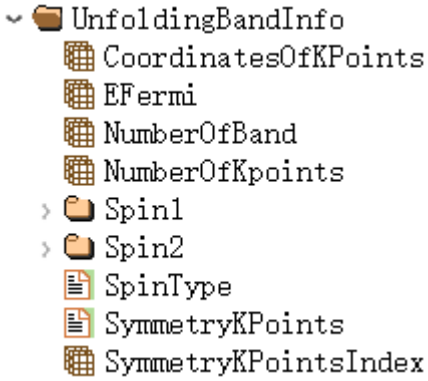
其中 *AtomInfo* 、*Structures* 与 *relax.h5* 文件相对应的结构体结构一致，*BandInfo* 中保存能带数据：



能带去折叠计算对应的 *band.h5* 至少包含 4 个结构体：



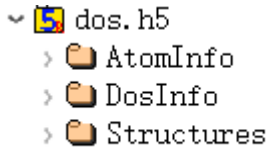
其中 *AtomInfo* 、*Structures* 与 *relax.h5* 文件相对应的结构体结构一致，*BandInfo* 中保存能带数据，*UnfoldingBandInfo* 中保存能带反折叠数据：



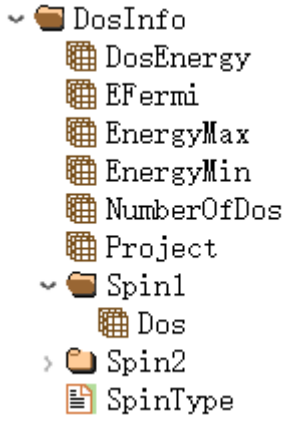
6.6 dos.h5

dos.h5 为 *task* = *dos* 下的输出文件，当 *task* 类型为其他时，该文件不输出。

dos.h5 至少包含 3 个结构体：



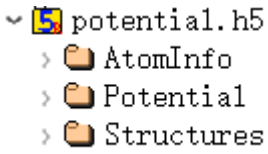
其中 *AtomInfo* 、*Structures* 与 *relax.h5* 文件相对应的结构体结构一致，*DosInfo* 中保存态密度数据：



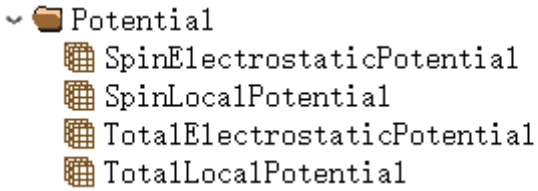
6.7 potential.h5

potential.h5 为 *task* = *potential* 下的输出文件，当 *task* 类型为其他时，该文件不输出。

potential.h5 至少包含 3 个结构体：



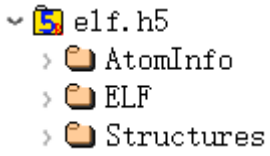
其中 *Potential* 中保存势函数数据：



6.8 elf.h5

elf.h5 为 *task* = *elf* 下的输出文件，当 *task* 类型为其他时，该文件不输出。

elf.h5 至少包含 3 个结构体：

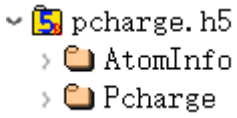


其中 *elf* 中保存局域密度数据：

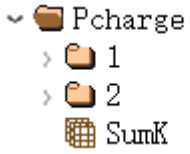
6.9 pcharge.h5

pcharge.h5 为 *task* = *pcharge* 下的输出文件，当 *task* 类型为其他时，该文件不输出。

pcharge.h5 包含 2 个结构体：



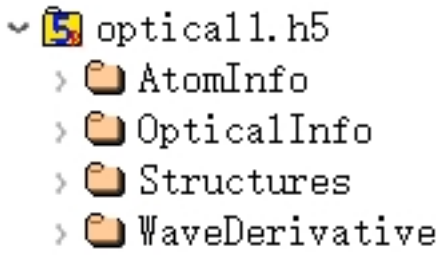
其中 *Pcharge* 中保存部分电荷密度数据：



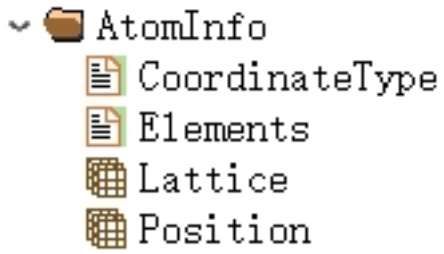
6.10 optical.h5

optical.h5 为 *task* = *optical* 下的输出文件，当 *task* 类型为其他时，该文件不输出。

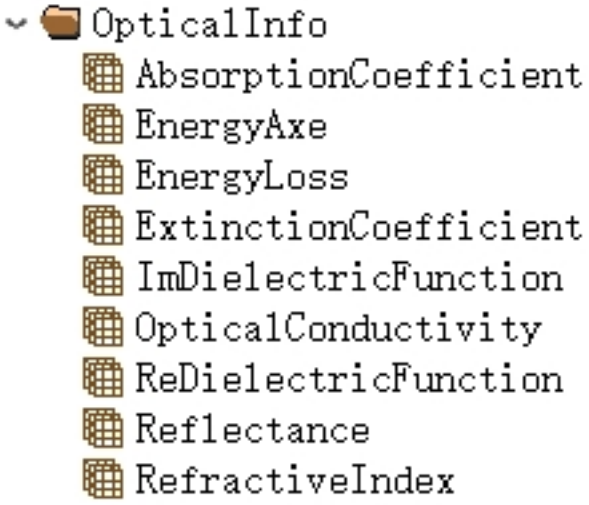
optical.h5 包含 4 个结构体：



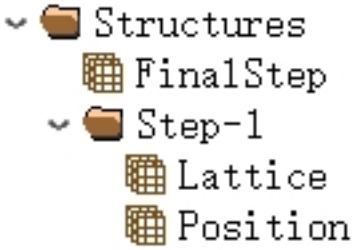
（1）其中 *AtomInfo* 中保存体系的基本结构信息，如晶胞大小，原子位置等：



（2）*opticalInfo* 中保存光学计算各种性质的数据信息：

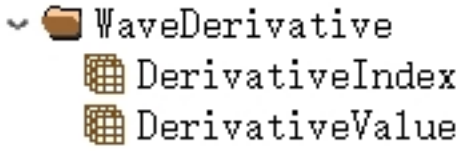


（3）*Structures* 中保存光学计算的结构信息：



（4）*WaveDerivate* 中保存波函数对 k 点的导数数组，大小为：（实部，虚部）*NumberOfBand* 优选后

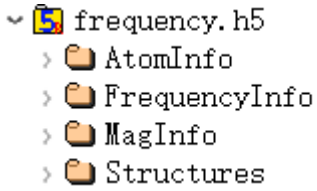
band 数目 *NumberOfKPoints*NumberOfSpin* (x,y,z)；*DerivativeIndex* 给出导数数组的维度；*DerivativeValue* 给出导数数组的值。



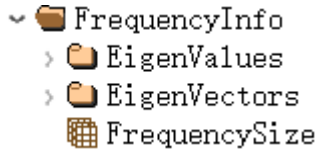
6.11 frequency.h5

frequency.h5 为 *task = frequency* 下的输出文件，当 *task* 类型为其他时，该文件不输出。

在考虑自旋的情况下 *frequency.h5* 包含 4 个结构体：



其中 *FrequencyInfo* 中保存频率数据：

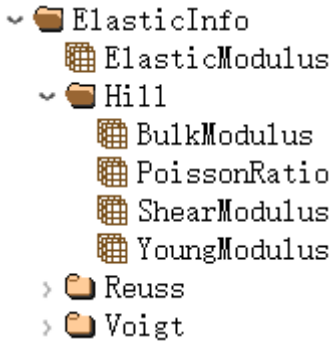


6.12 elastic.h5

elastic.h5 为 *task = elastic* 下的输出文件，当 *task* 类型为其他时，该文件不输出。

在考虑自旋的情况下 *elastic.h5* 包含 4 个结构体：

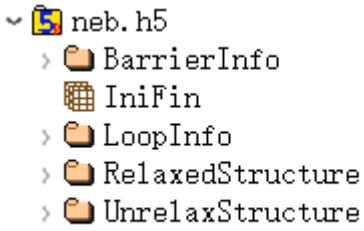
其中 *ElasticInfo* 中保存弹性数据：



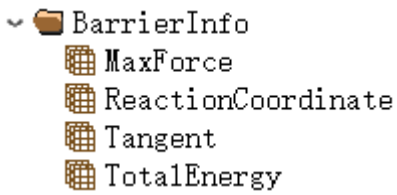
6.13 neb.h5

neb.h5 为 *task* = *neb* 时，外层目录的输出文件。

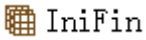
neb.h5 包含 5 个结构体：



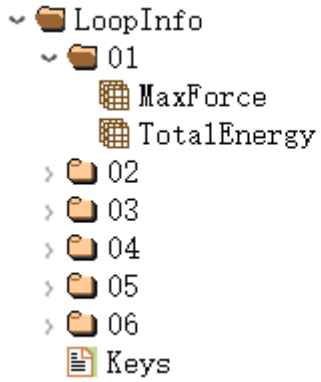
其中 *BarrierInfo* 中保存最大受力、反应坐标（各 image 和初态 00 结构间的反应距离）、最大剪切力、总能数据：



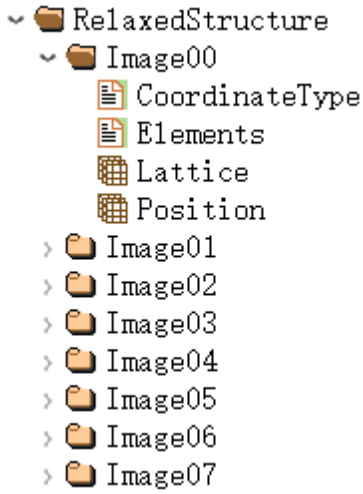
其中 *IniFin* 中保存计算初末态的开关：



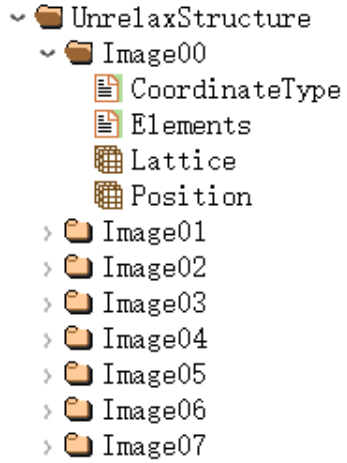
其中 *LoopInfo* 中保存 neb 优化过程中每个 image 的能量与受力变化情况：



其中 *RelaxedStructure* 中保存各 image 优化结束的结构数据：



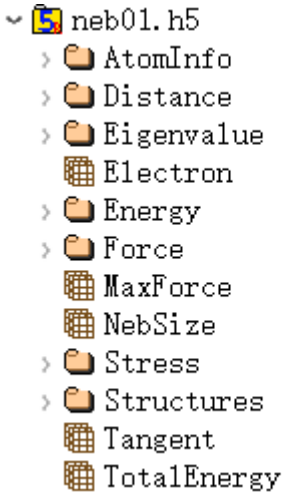
其中 *UnrelaxStructure* 中保存各 image 优化之前的结构数据：



6.14 neb01.h5

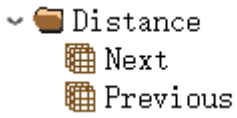
neb01.h5 为 *task* = *neb* 时，*01* 子文件夹下的输出文件，同理 *02* 文件夹下会生成 *neb02.h5* 文件。

在不考虑自旋的情况下 *neb01.h5* 包含 12 个结构体：



其中 *AtomInfo* 、*Eigenvalue* 、*Electron* 、*Energy*、*Force*、*Stress* 、*Structures* 与 *relax.h5* 文件相对应的结构体结构一致；

其中 *Distance* 中保存 image 1 在优化过程中反应原子距离前后 image 的距离变化数据；



其中 *MaxForce* 中保存 image 1 在优化过程中最大受力数据；

其中 *NebSize* 中保存过渡态计算最大步数；

其中 *Tangent* 中保存 image 1 在优化过程中剪切力变化数据；

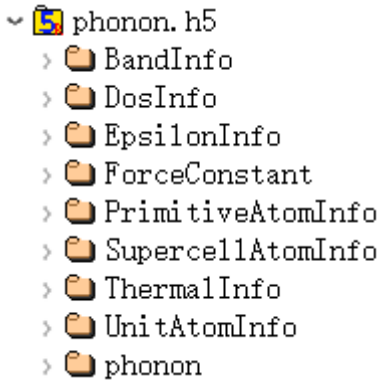
其中 *TotalEnergy* 中保存 image 1 在优化过程中总能变化数据；

6.15 phonon.h5

phonon.h5 为 *task* = *phonon* 下的输出文件，当 *task* 类型为其他时，该文件不输出。

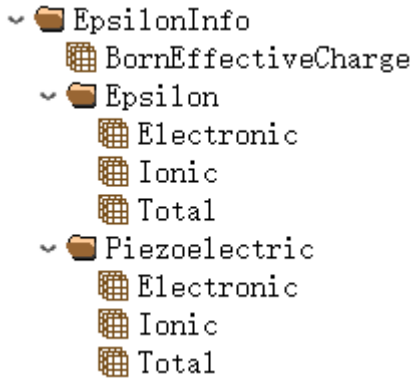
（1）当 `phonon.method = dfpt` 时，phonon.f5 文件如下所示：

dfpt 方法计算声子能带和态密度，打开介电常数、声子热力学计算的开关，*phonon.h5* 包含 9 个结构体：

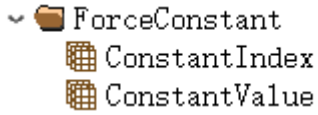


其中 *BandInfo* 、*DosInfo* 分别保存能态和态密度数据，其结构分别与 *band.h5* 、*dos.h5* 文件相对应的结构体结构一致；

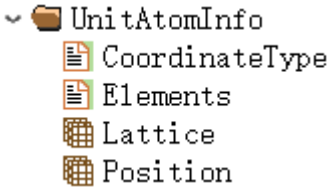
其中 *EpsilonInfo* 中保存介电常数数据：



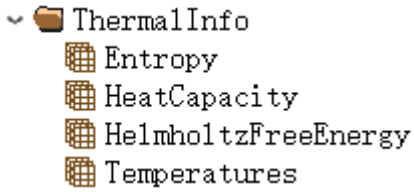
其中 *ForceConstant* 中保存力学常数数据：



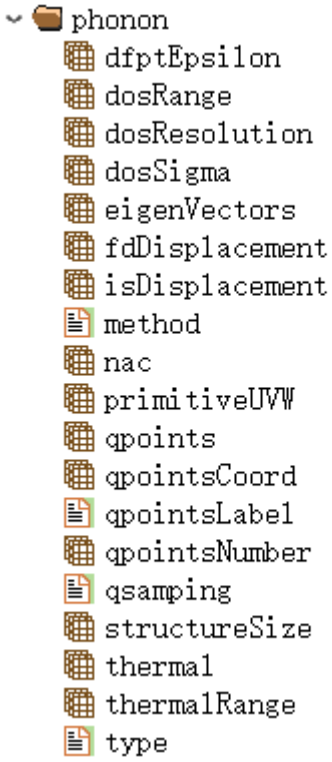
其中 *PrimitiveAtomInfo* 、*SupercellAtomInfo* 、*unitAtomInfo* 分别保存原胞、超胞、单胞的结构信息，以单胞为例结构如下：



其中 *ThermalInfo* 中保存声子热力学数据：

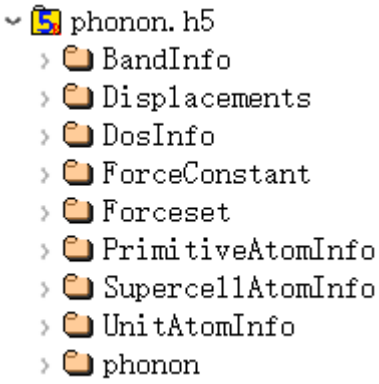


其中 *Phonon* 中保存声子计算输入参数数据：



（2）当 `phonon.method = fd` 时，phonon.f5 文件如下所示：

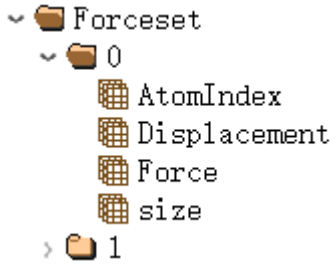
有限位移法计算声子能带和态密度，*phonon.h5* 包含 9 个结构体：



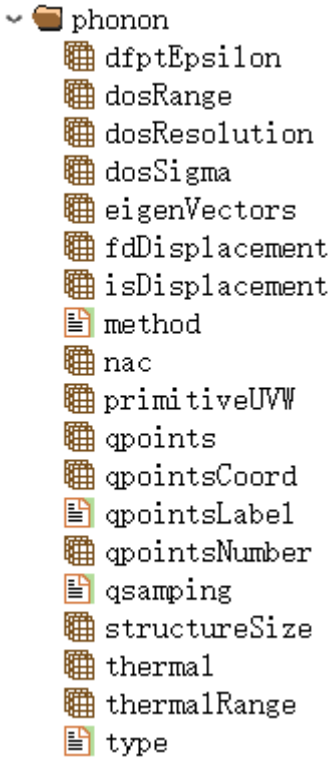
其中 *BandInfo* 、*DosInfo* 分别保存能态和态密度数据，其结构分别与 *band.h5* 、*dos.h5* 文件相对应的结构体结构一致；

其中 *ForceConstant* 中保存力学常数数据，*PrimitiveAtomInfo* 、*SupercellAtomInfo* 、*unitAtomInfo* 分别保存原胞、超胞、单胞的结构信息，其结构与 `phonon.method = dfpt` 时生成的 phonon.f5 文件相对应的结构体结构一致；

其中 *ForceSet* 保存各结构计算所得的力学矩阵数据：



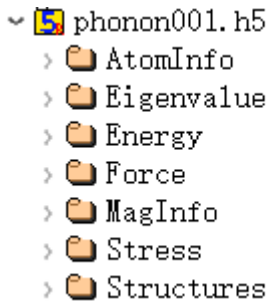
其中 *Phonon* 中保存声子计算输入参数数据：



6.16 phonon001.h5

task = phonon，*phonon.method = fd* 时，*001* 子文件下会输出文件 *phonon.h5*，可将此类 *h5* 文件重命名为 *phonon001.h5*，同理 *002* 文件夹下也会生成 *phonon.h5* 文件。

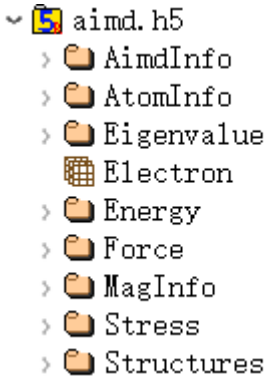
在考虑自旋的情况下 *phonon001.h5* 包含 7 个结构体，结构如下：



6.17 aimd.h5

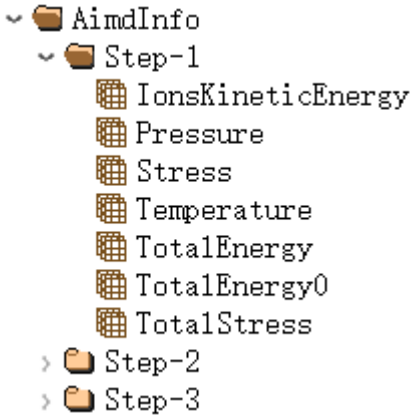
aimd.h5 为 *task* = *aimd* 下的输出文件，当 *task* 类型为其他时，该文件不输出。

在考虑自旋的情况下 *aimd.h5* 包含 9 个结构体：



其中基本结构体信息与 *relax.h5* 一致；

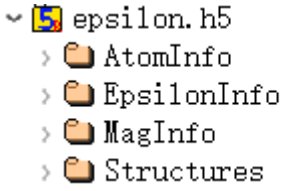
其中新结构体 *AimdInfo* 包含 n 个结构体，每个结构体保存某个离子步下体系的状态信息，如温度、压力、能量、动能等：



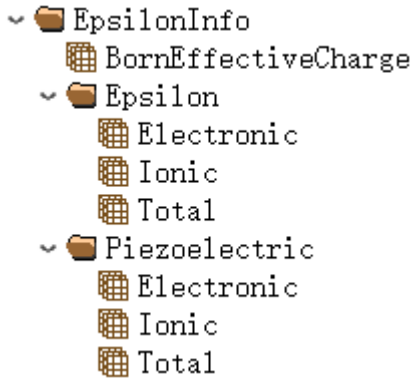
6.18 epsilon.h5

epsilon.h5 为 *task* = *epsilon* 下的输出文件，当 *task* 类型为其他时，该文件不输出。

在考虑自旋的情况下 *epsilon.h5* 包含 4 个结构体：



其中 *EpsilonInfo* 中保存介电常数数据：

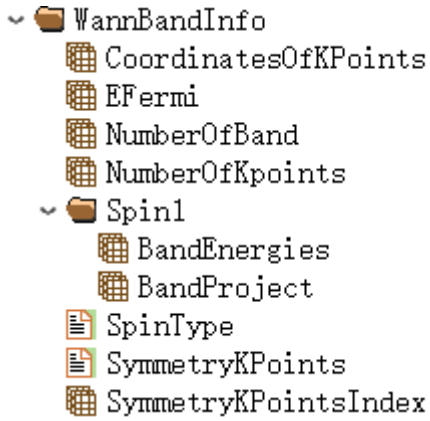


6.19 wannier.h5

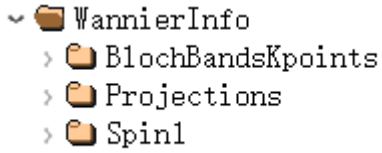
wannier.h5 为 *task* = *wannier* 下的输出文件，当 *task* 类型为其他时，该文件不输出。

在不考虑自旋的情况下，计算插值能带所得的 *wannier.h5* 包含 9 个结构体：

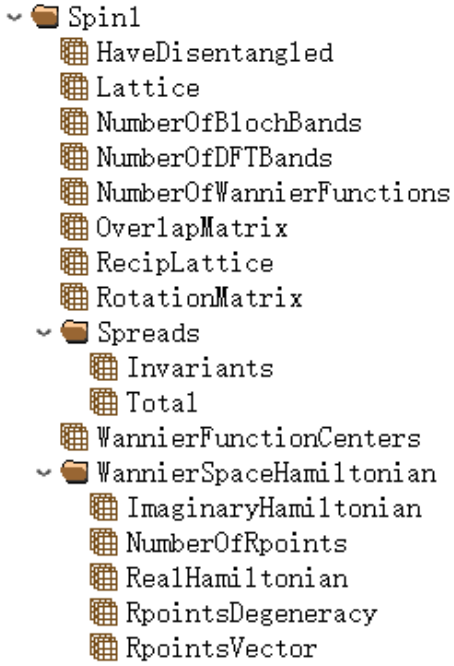
其中 *WannBandInfo* 中保存插值能带数据，可用于绘制能带图：



其中 *WannInfo* 中保存 wannier 函数拟合数据，包括 k 网格点和初始投影信息等：



其中 *WannInfo* 的 *spin1* 下保存 wannier 函数拟合数据，包括计算所得的哈密顿量等数据：



7

续算说明

DS-PAW 目前可支持 **结构弛豫**、**过渡态计算**、**分子动力学模拟**、**恒电势计算**、**读取 rho 和 wave** 五种功能的续算，用户通过指定文件路径读取前次计算所得的末态结构、磁矩、电势等相关信息。

7.1 relax 弛豫计算续算说明

弛豫计算意外中止、或最大步数内未收敛、或想做更高精度的弛豫计算，都需获取上一次计算所得的末态结构（在体系考虑自旋的情况下需获取末态构型的磁矩信息）进行下一次弛豫计算。该情况下程序会输出 *latestStructure.as* 和 *relax.h5* 文件，*latestStructure.as* 和 *relax.h5* 都可作为续算的输入文件。若需在此结构的基础上进行续算，建议按以下步骤完成：

1. 新建干净的目录，准备两个输入文件：*relax.in* 和 *latestStructure.as* （或 *relax.h5* ）；
2. 在 *relax.in* 文件中设置参数 **sys.structure = latestStructure.as** （或 **sys.structure = relax.h5** ），结构文件的名称可自行修改，建议提供醒目的续算提示；
3. 提交任务进行计算。

latestStructure.as 为结构弛豫计算续算可读的文件之一，除此以外 *relax.h5* 文件也可作为末态结构被读入。

7.2 neb 过渡态计算续算说明

过渡态计算意外中止、或最大步数内未收敛、或想做更高精度的过渡态计算，都需获取上一次计算所得的末态结构（在体系考虑自旋的情况下需获取末态构型的磁矩信息）进行下一次过渡态计算。过渡态计算涉及多个子文件夹，该情况下各子文件夹 No 下会默认输出 *latestStructureNo.as* 和 *nebNo.h5* 文件，*.as* 文件可作为续算的输入文件。以插点数 **3** 为例，若需在此结构的基础上进行续算，可直接调用辅助工具使用教程 部分的 neb 续算脚本进行处理：

调用 python 脚本进行数据处理过程演示：

1. 进入 neb 初次计算目录，查看该目录下的文件：

```
(base) [hzw1002@mgt2 neb]$ ls
00  01  02  03  04  dev.slurm  DS-PAW.log  _err.dat  input.in  neb-restart.py  _out.dat
```

2. 在该目录下调用 *neb_restart.py* 脚本，执行如下命令：

```
python neb_restart.py
```

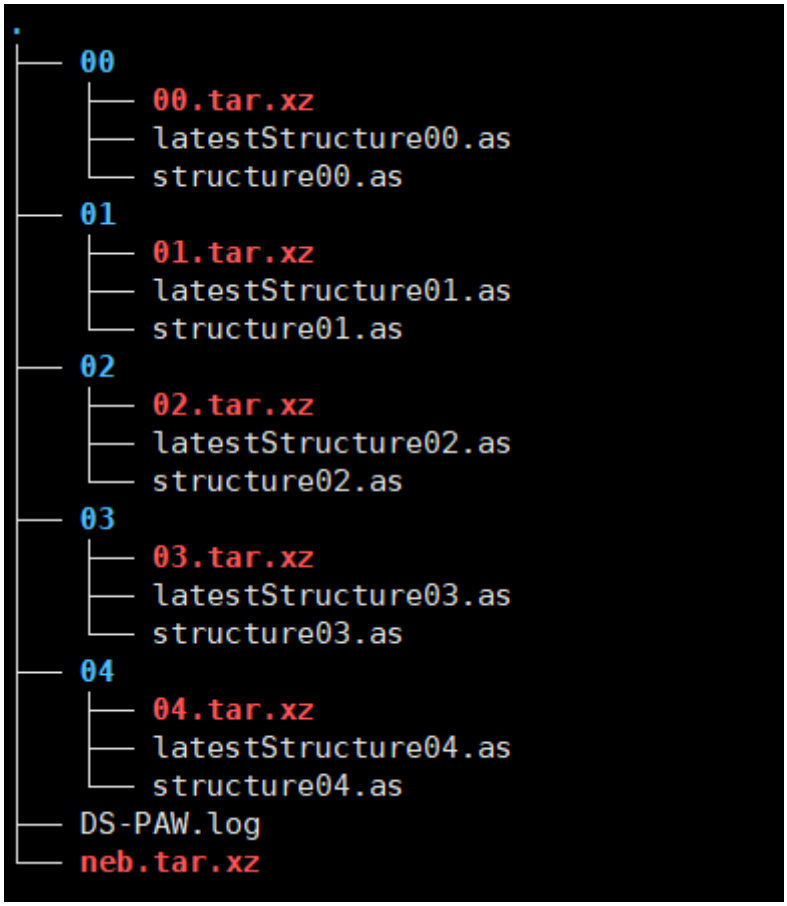
按照提示在交互界面指定 neb 原文件路径、参数文件名及备份文件夹名称，此例指定备份文件夹为 *bakfile*。

3. 再次查看 neb 目录：

```
(base) [hzw1002@mgt2 neb]$ ls
00  01  02  03  04  bakfile  dev.slurm  _err.dat  input.in  neb-restart.py  _out.dat
```

其中 *bakfile* 为备份文件，*00-04* 文件夹存放续算所需的结构文件，在该目录下可直接提交进行续算。

4. 备份文件夹 *bakfile* 结构解析。



备份文件夹下最外层的压缩包 *neb.tar.xz* 存放初次 neb 计算的 *h5* 文件，各子文件下的压缩包为初次 neb 计算子文件夹下所有文件备份，子文件夹外层存放初次计算的初末态结构文件。

用户若自行准备输入文件，建议按以下步骤完成：

1. 新建干净的目录，放入 *neb.in* 文件、初末态结构文件 *structure00.as* 、*structure04.as*，中间构型的末态结构文件 *latestStructure01.as* 、*latestStructure02.as* 、*latestStructure03.as* ；
2. 将中间结构文件 *latestStructureNo.as* 分别重命名为 *structureNo.as* ；
3. 新建文件夹 **00** 、**01** 、**02** 、**03** 、**04** ，将各结构文件放置于对应文件夹下；
4. 提交任务进行计算。

.as 文件为过渡态计算续算的可读文件，不建议使用 *nebNo.h5* 作为续算的输入文件。

7.3 aimd 分子动力学模拟续算说明

分子动力学模拟计算意外中止、或想加大模拟时长，需要获取上一次计算所得的末态结构和速度（在体系考虑自旋的情况下需获取末态构型的磁矩信息）进行更长时间的模拟，分子动力学模拟默认会输出 *latestStructure.as* 和 *aimd.h5* 文件，*latestStructure.as* 和 *aimd.h5* 都可作为续算的输入文件。若需在此结构的基础上进行续算，建议按以下步骤完成：

1. 新建干净的目录，准备两个输入文件：*aimd.in* 和 *latestStructure.as* （或 *aimd.h5* ）；
2. 在 *aimd.in* 文件中设置参数 **sys.structure = latestStructure.as** （或 **sys.structure = aimd.h5** ），结构文件的名称可自行修改，建议提供醒目的续算提示；
3. 提交任务进行计算。

latestStructure.as 为分子动力学计算续算可读的文件之一，除此以外 *aimd.h5* 文件也可作为末态结构被读入。

Note：

1. 若需修改系综进行续算，需删除 latestStructure.as 文件中 **Next positions** 部分信息，否则续算可能报错。

7.4 fixedPotential 恒电势计算续算说明

恒电势计算采用的是最速下降法，通过多步自洽求解目标电荷及电势值，可将整个过程看作 n 个前后依赖的自洽计算，若在电荷收敛之前某处计算意外中止，可使用续算功能，以中断前获取的电荷及电势值作为搜寻起点逼近目标电势，恒电势续算建议按以下步骤完成：

1. 在原计算目录下修改 *fixedPotential.in* 文件，指定初次计算所得的 h5 文件所在目录即可进行续算，对应参数 `cal.iniFixedP = ./scf.h5` 。

Note：

1. 若需保留初次计算的 scf.h5 文件，可将原文件重命名，如重命名为 readscf.h5，设置 *cal.iniFixedP = ./readscf.h5* 。

2. 续算时从指定文件获取电子数和目标电极电势值，在 in 文件中修改此类参数无效。

7.5 读取 rho 和 wave 续算说明

杂化泛函计算耗时长，在一步计算未收敛或想提高收敛精度再次计算时，可读取已得到的电荷密度和波函数文件，通过 `cal.iniCharge` 和 `cal.iniWave` 参数指定文件路径即可。如下 Resatrt-HSE.in 文件列出杂化泛函续算的关键参数：

```
# task type
task = scf

#hybrid related
sys.hybrid=true
sys.hybridType=HSE06

#read related
cal.iniCharge = ../01/rho.bin
cal.iniWave = ../01/wave.bin

#outputs related
io.charge = true
io.wave = true
```

Note：

1. 杂化泛函计算的续算需同时提供电荷密度和波函数文件，缺一不可。
2. 杂化泛函计算建议输出 rho.bin 和 wave.bin 文件，可作于续算输入。

8

辅助工具使用教程

备注

用 DSPAW 完成 DFT 计算后，想快速分析结果、画图，或者完成一些常见的数据处理任务？

dspawpy （Python ≥ 3.9 ）就是这样一款辅助工具，它可以编程调用（参考下文示例脚本），也提供了一个命令行交互程序

参考下方教程安装 后，命令行输入 dspawpy 后回车即可使用交互程序：

```
... loading dspawpy cli ...

********这是dspawpy命令行交互小工具，预祝您使用愉快********
   ( )
  _| |  ___  _ _      _ _  _   _   _  _ _    _   _
/'_` |/',__)( '_`\  /'_` )( ) ( ) ( )( '_`\ ( ) ( )
( (_| |\__, \| (_) )( (_| || \_/ \_/ || (_) )| (_) |
\__,_)(____/| ,__/'`\__,_) \___x___/ | ,__/  \__, |
            | |                     | |    ( )_| |
            (_)                     (_)     \___/

[版本号]: [安装路径]

=====================================
|  1: update更新
|  2: structure结构转化
|  3: volumetricData数据处理
|  4: band能带数据处理
|  5: dos态密度数据处理
|  6: bandDos能带和态密度共同显示
|  7: optical光学性质数据处理
|  8: neb过渡态计算数据处理
|  9: phonon声子计算数据处理
|  10: aimd分子动力学模拟数据处理
|  11: Polarization铁电极化数据处理
|  12: ZPE零点振动能数据处理
```

```
|  13: TS的热校正能
|  14: relaxation结构优化日志分析
|  15: hdf5文件探索
|
|  q: 退出
=======================================
--> 输入数字后回车选择功能:
```

亮点:

- 自动补全：按 Tab 键即可生效，有助于快速且正确地输入程序所需要的参数
- 多线程懒加载：在等待用户输入时后台加载模块，大幅缩短等待时间；仅加载必要模块，减少内存占用

注意:

- 在远程服务器上使用时，由于机器的磁盘读写性能不佳，可能需要较长的启动时间，极端情况下可能长达半分钟（与服务器当时的磁盘读写性能直接相关）。如果无法忍受，请在自己的电脑上安装 dspawpy 使用
- 输入 *dspawpy* 回车后，python 会先加载内建模块，完成后显示…loading dspawpy cli …提示语句，表明进入第二阶段（开始加载第三方依赖库）
- 等待第二阶段完成后，会显示欢迎界面，此时 dspwapy 已完成前期加载，进入第三阶段。后续将动态根据选择的功能模块加载相应的依赖库，从而最大限度减少等待时间

8.1 安装与更新

1. 在 HZW 机器上，已提前安装 dspawpy，使用以下命令激活虚拟环境即可使用:

```
source /data/hzwtech/profile/dspawpy.env
```

2. 在其他机器上，请自行安装 dspawpy（下列两种方式二选一）:

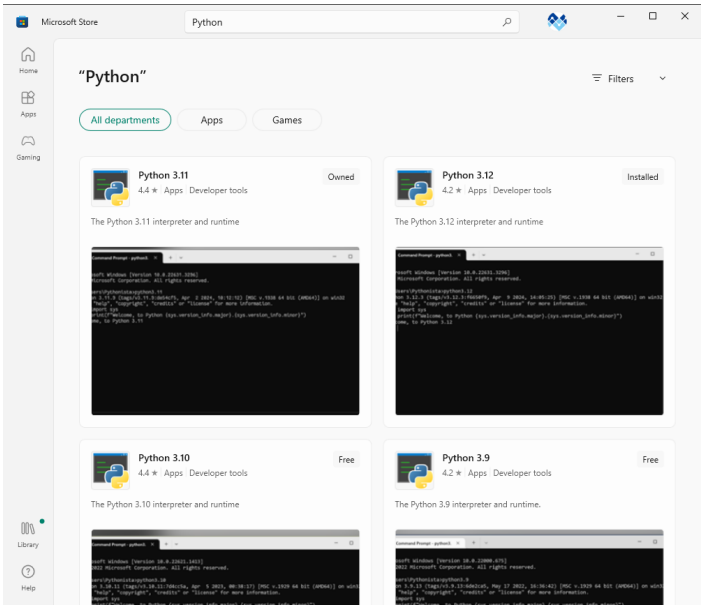
- 使用 mamba 或 conda ，安装包可前往 https://conda-forge.org/download/ 下载

```
mamba install dspawpy -c conda-forge
#conda install dspawpy -c conda-forge
```

- 或使用 pip3 (部分操作系统中没有 pip3 可执行程序，此时请尝试 pip)

pip

- pip3 是 python3 自带的包管理器
- Linux 和 Mac 一般自带 python3 和 pip3。
- Windows 平台打开 microsoft store 搜索 python 安装即可



然后打开 cmd 或者 powershell 即可使用 pip。

```
pip3 install dspawpy
```

关于如何设置 pip 和 conda 镜像地址以加速安装过程，可参考 https://mirrors.tuna.tsinghua.edu.cn/help/pypi/ 和 https://mirrors.tuna.tsinghua.edu.cn/help/anaconda/

如果安装依然失败，请尝试上面的 mamba/conda 安装法。

警告

在集群上，由于权限问题，公共路径中的 pip 很可能不支持全局安装 python 库，必须在 pip install 后面追加 `--user` 选项安装到家目录 `~/.local/lib/python3.x/site-packages/` 下，其中 3.x 表示 python 解释器版本，x 可能是 9-13 之间的任意整数

python 将优先读取家目录下的 dspawpy，即使公共环境中的 dspawpy 的版本比家目录中的 dspawpy 的版本新！因此，如果以前用 `--user` 安装过 dspawpy，又忘记手动更新家目录下的老版本 dspawpy，即使 source 了公共环境，也无法调用公共环境中的 dspawpy，依旧会使用老版本 dspawpy，导致一些 BUG。

因此，**鉴于 HZW 集群每周自动更新 dspawpy，建议不要在家目录下重复安装，已安装的建议删除**。如果在其他集群上，请确保及时手动更新家目录下的 dspawpy

如果不想删除家目录中的 dspawpy，又不想更新它，可以在运行 python 脚本时，尝试 -s 选项避免导入家目录中的 dspawpy：`python -s your-script.py`。

8.1.1 更新 dspawpy

如果 dspawpy 是通过 mamba/conda 安装的，使用以下命令更新：

```
mamba update dspawpy
#conda update dspawpy
```

如果 dspawpy 是通过 pip 安装的，那么：

```
pip install dspawpy -U # -U 表示安装最新版
```

备注

如果 pip 使用了国内的镜像网站，可能由于镜像网站尚未同步最新版 dspawpy dspawpy 而导致无法顺利升级，请使用如下命令告诉 pip 从 pypi 官网下载安装：

```
pip install dspawpy -i https://pypi.org/simple --user -U # -i 指定下载地址，--user␣
↪表示仅为当前用户安装，-U 表示安装最新版
```

如果运行时出现 **dspawpy 相关**错误信息，请先检查是否已正确导入最新版本的 dspawpy 并检查安装路径：

```
$ python3 # 或 python
>>> import dspawpy
>>> dspawpy.__version__ # 将输出版本号
>>> dspawpy.__file__ # 将输出安装路径
```

8.2 structure 结构转化

读取结构信息使用 `read` 函数；将结构信息写入文件，使用 `write` 函数；快速结构转化，使用 `convert` 函数：

API: read(), write(), convert()

dspawpy.io.structure.**convert**(*infile*, *si=None*, *ele=None*, *ai=None*, *infmt: str | None = None*, *task: str = 'scf'*, *outfile: str = 'temp.xyz'*, *outfmt: str | None = None*, *coords_are_cartesian: bool = True*)

convert from infile to outfile.

- multi -> single, only keep last step
- crystal -> molecule, will lose lattice info
- molecule -> crystal, will add a box of twice the maximum xyz
- pdb, dump may suffer decimal precision loss

参数

- **infile** –
  - h5/json/as/hzw/cif/poscar/cssr/xsf/mcsqs/prismatic/yaml/fleur-inpgen file path
  - If a folder is given, will read {task}.h5/json files

  - If structures are given, will read multiple structures.
- **si** (*int, list, or str*) –
  - Structure index, starting from 1
    - si=1, read the 1st
    - si=[1,2], read the 1st and 2nd
    - si=’:’, read all
    - si=’-3:’, read the last 3
  - If empty, for multi-configuration files, all configurations will be read; for single-configuration files, the latest configuration will be read.
  - This parameter is only valid for h5/json files.
- **ele** –
  - Element symbol, written as ‘H’ or [‘H’,’O’]
  - If empty, atomic information for all elements will be read.
  - This parameter is only valid for h5/json files.
- **ai** –
  - Atom index, starting from 1
  - Usage is the same as si
  - If empty, atomic information for all atoms will be read.
  - This parameter is only valid for h5/json files.
- **infmt** –
  - Input structure file type, e.g., ‘h5’. If None, the file extension will determine the format.
- **task** –
  - Used when datafile is a folder path to locate the internal {task}.h5/json file.
  - Calculation task type, including ‘scf’, ‘relax’, ‘neb’, ‘aimd’. Other values will be ignored.
- **outfile** –
  - Output filename
- **outfmt** –
  - Output structure file type, e.g., ‘xyz’. If None, the file extension will determine the format.
- **coords_are_cartesian** –
  - Whether to write coordinates in Cartesian form (default: True); otherwise, fractional coordinates will be used.
  - This option is currently only valid for as and json formats.

示例

```
>>> from dspawpy.io.structure import convert
>>> convert('tests/supplement/PtH.as', outfile='tests/doctest_out/PtH.hzw')
==> .../PtH...hzw
```

batch test

```
>>> for readable in ['relax.h5', 'system.json', 'aimd.pdb', 'latestStructure.as',
↪'CuO.hzw', 'POSCAR']:
...     for writable in ['pdb', 'xyz', 'dump', 'as', 'hzw', 'POSCAR']:
...         convert('tests/supplement/stru/'+readable, outfile=f"tests/doctest_
↪out/{readable.split('.')[0]}.{writable}")
==> .../relax...pdb
==> .../relax...xyz
==> .../relax...dump
==> .../relax...as
==> .../relax...hzw
==> .../system...pdb
==> .../system...xyz
==> .../system...dump
==> .../system...as
==> .../system...hzw
==> .../aimd...pdb
==> .../aimd...xyz
==> .../aimd...dump
==> .../aimd...as
==> .../aimd...hzw
==> .../latestStructure...pdb
==> .../latestStructure...xyz
==> .../latestStructure...dump
==> .../latestStructure...as
==> .../latestStructure...hzw
==> .../CuO...pdb
==> .../CuO...xyz
==> .../CuO...dump
==> .../CuO...as
==> .../CuO...hzw
==> .../POSCAR...pdb
==> .../POSCAR...xyz
==> .../POSCAR...dump
==> .../POSCAR...as
==> .../POSCAR...hzw
```

dspawpy.io.structure.**read**(*datafile: str | list*, *si=None*, *ele=None*, *ai=None*, *fmt: str | None = None*, *task: str | None = 'scf'*)

Read one or more h5/json files and return a list of pymatgen Structures.

参数

- **datafile** –
  - file paths for h5/json/as/hzw/cif/poscar/cssr/xsf/mcsqs/prismatic/yaml/fleur-inpgen files;
  - If a directory path is given, it can be combined with the task parameter to read the {task}.h5/json files inside
  - If a list of strings is given, it will sequentially read the data and merge them into a list of Structures

- **si** (*int, list or str*) –
  - Configuration number, starting from 1
    - si=1, reads the first configuration
    - si=[1,2], reads the first and second configurations
    - si=’:’, reads all configurations
    - si=’-3:’, reads the last three configurations
  - If empty, it reads all configurations for multi-configuration files and the latest configuration for single-configuration files
  - This parameter is only valid for h5/json files
- **ele** –
  - Element symbol, format reference: ‘H’ or [‘H’,’O’]
  - If empty, it will read atomic information for all elements
  - This parameter is only valid for h5/json files
- **ai** –
  - Atom index, starting from 1
  - Same as si
  - If empty, it will read all atom information
  - This parameter is only valid for h5/json files
- **fmt** –
  - File format, including ‘as’, ‘hzw’, ‘xyz’, ‘pdb’, ‘h5’, ‘json’ 6 types, other values will be ignored.
  - If empty, the file type will be determined based on file name conventions.
- **task** –
  - Used when datafile is a directory path to find the internal {task}.h5/json file.
  - Determine the task type, including ‘scf’, ‘relax’, ‘neb’, ‘aimd’ four types, other values will be ignored.

返回
Structure list

返回类型
pymatgen_Structures

示例

```
>>> from dspawpy.io.structure import read
```

Reads a single file to generate a list of Structures

```
>>> pymatgen_Structures = read(datafile='tests/supplement/PtH.as')
>>> len(pymatgen_Structures)
1
>>> pymatgen_Structures = read(datafile='tests/supplement/PtH.hzw')
```

(续下页)

(接上页)

```
>>> len(pymatgen_Structures)
1
>>> pymatgen_Structures = read(datafile='tests/supplement/Si2.xyz')
>>> len(pymatgen_Structures)
1
>>> pymatgen_Structures = read(datafile='tests/aimd.pdb')
>>> len(pymatgen_Structures)
1000
>>> pymatgen_Structures = read(datafile='tests/2.1/relax.h5')
>>> len(pymatgen_Structures)
3
>>> pymatgen_Structures = read(datafile='tests/2.1/relax.json')
>>> len(pymatgen_Structures)
3
```

Note that pymatgen_Structures is a list composed of multiple Structure objects, each corresponding to a structure. If there is only one structure, it will also return a list. Please use pymatgen_Structures[0] to obtain the Structure object.

When datafile is a list, it reads multiple files sequentially and merges them into a Structures list

```
>>> pymatgen_Structures = read(datafile=['tests/supplement/aimd1.h5','tests/
↪supplement/aimd2.h5'])
```

dspawpy.io.structure.**write**(*structure*, *filename: str*, *fmt: str | None = None*, *coords_are_cartesian: bool = True*)

Write information to the structure file

参数

- **structure** –*A pymatgen Structure object*
- **filename** –Structure filename
- **fmt** –
  - Structure file type, natively supports ‘json’, ‘as’, ‘hzw’, ‘pdb’, ‘xyz’, ‘dump’, ‘png’, ‘gif’ eight types
- **coords_are_cartesian** –
  - Whether to write in Cartesian coordinates, default is True; otherwise write in fractional coordinate format
  - This option is currently only effective for ‘as’ and ‘json’ formats

示例

First, read the structure information:

```
>>> from dspawpy.io.structure import read, write
>>> s = read('tests/2.15/01/neb01.h5')
>>> len(s)
17
```

Writing structure information to a file:

```
>>> write(s, filename='tests/doctest_out/PtH.json', coords_are_cartesian=True)
==> .../PtH...json
>>> write(s, filename='tests/doctest_out/PtH.as', coords_are_cartesian=True)
==> .../PtH...as
>>> write(s, filename='tests/doctest_out/PtH.hzw', coords_are_cartesian=True)
==> .../PtH...hzw
```

PDB, XYZ, and DUMP file types can write multiple conformations to form a "trajectory." The generated XYZ trajectory files can be opened and visualized using visualization software like OVITO.

```
>>> write(s, filename='tests/doctest_out/PtH.pdb', coords_are_cartesian=True)
==> .../PtH...pdb
>>> write(s, filename='tests/doctest_out/PtH.xyz', coords_are_cartesian=True)
==> .../PtH...xyz
>>> write(s, filename='tests/doctest_out/PtH.dump', coords_are_cartesian=True)
==> .../PtH...dump
```

PNG and GIF formats can be used to export structure images using ASE's io module. PNG format exports a single structure image, while GIF format can create an animation for multiple structures.

```
>>> write(s, filename='tests/doctest_out/PtH.png')
==> .../PtH...png
>>> write(s, filename='tests/doctest_out/PtH.gif')
==> .../PtH...gif
```

The recommended format for storing single structure information is as format. If the Structure contains magnetic moment or degree of freedom information, it will be written in the most complete format, such as Fix_x, Fix_y, Fix_z, Mag_x, Mag_y, Mag_z. The default value for degree of freedom information is F, and the default value for magnetic moment is 0.0. You can manually delete this default information from the generated as file as needed. Writing to other types of structure files will ignore magnetic moment and degree of freedom information.

可参考 `2conversion.py` 脚本完成转化：

```python
# coding:utf-8
from dspawpy.io.structure import convert

convert(
    infile="tests/2.1/relax.h5",  # Structure to be converted, if in the current path, you can just write the filename
    si=None,  # Select configuration number, if not specified, read all by default
    ele=None,  # Filter element symbol, default reads atomic information for all elements
    ai=None,  # Filter atomic indices, starting from 1, default to read all atomic information
    infmt=None,  # Input structure file type, e.g., 'h5'. If None, it will be matched ambiguously based on the filename rule.
    task="relax",  # Task type, this parameter is only valid when infile is a folder rather than a filename
    outfile="tests/outputs/us/relaxed.xyz",  # Structure file name
    outfmt=None,  # Output structure file type, e.g., 'xyz'. If None, it will be fuzzy matched according to filename rules.
    coords_are_cartesian=True,  # Written in Cartesian coordinates by default
)
```

convert 函数的几个关键参数设置规则见下表：

表 1: **dspawpy 支持读写的结构文件列表**

infmt（输入文件格式）	infile（输入文件名模糊匹配）	outfmt（输出文件格式）	outfile（输出文件名模糊匹配）	说明
h5	**.h5*	X	X	DS-PAW 计算完成后保存的 hdf5 类型文件
json	**.json*	json	**.json*	DS-PAW 计算完成后保存的 json 类型文件
pdb	**.pdb*	pdb	**.pdb*	Protein Data Bank
as	**.as*	as	**.as*	DS-PAW 记载原子坐标等信息的结构文件
hzw	**.hzw*	hzw	**.hzw*	DeviceStudio 默认的结构文件
xyz	**.xyz*	xyz	**.xyz*	读取时仅支持分子结构的单构型，写入时则是包含晶胞的 extended-xyz 类型轨迹文件
X	X	dump	**.dump*	lammps-dump 类型轨迹文件
X	**.cif*/*.mcif**	cif/mcif	**.cif*/*.mcif**	Crystallographic Information File
X	**POSCAR*/*CONTCAR*/*.vas*	poscar	**POSCAR**	VASP 文件
X	**.cssr**	cssr	**.cssr**	Crystal Structure Standard Representation
X	**.yaml/*.yml*	yaml/yml	**.yaml/*.yml*	YAML Ain’t Markup Language
X	**.xsf**	xsf	**.xsf**	eXtended Structural Format
X	**rndstr.in*/*lat.in*/*bestsqs**	mcsqs	**rndstr.in*/*lat.in*/*	Monte Carlo Special Quasirandom Structure
X	*inp*.xml/*.in*/inp_**	fleur-inpgen	**.in**	FLEUR 结构文件，须额外安装 pymatgen-io-fleur 库
X	**.res*	res	**.res*	ShelX res 结构文件
X	**.config*/*.pwmat**	pwmat	**.config/*.pwmat*	PWmat 文件
X	X	prismatic	**prismatic**	用于 STEM 模拟的一种文件格式
X	*CTRL**	X	X	Stuttgart LMTO-ASA 文件

备注

- 上述表格中 * 号表示任意字符，X 表示不支持
- h5, json, pdb, xyz, dump, CONTCAR 等格式支持多个结构组成的轨迹信息（常见于结构优化或者 NEB 或者 AIMD 任务）
- in(out)fmt 参数优先级高于文件名模糊匹配规则；例如，指定 in(out)fmt=’h5’ 后，文件名可以是任意值，甚至可以是 a.json
- 将结构信息写为 *json* 格式时，仅支持可视化 NEB 链任务，详见观察 *NEB* 链 节相关说明
- DS-PAW 输出的 neb.h5、phonon.h5、phonon.json、neb.json 以及 wannier.json 暂时无法读取结构信息

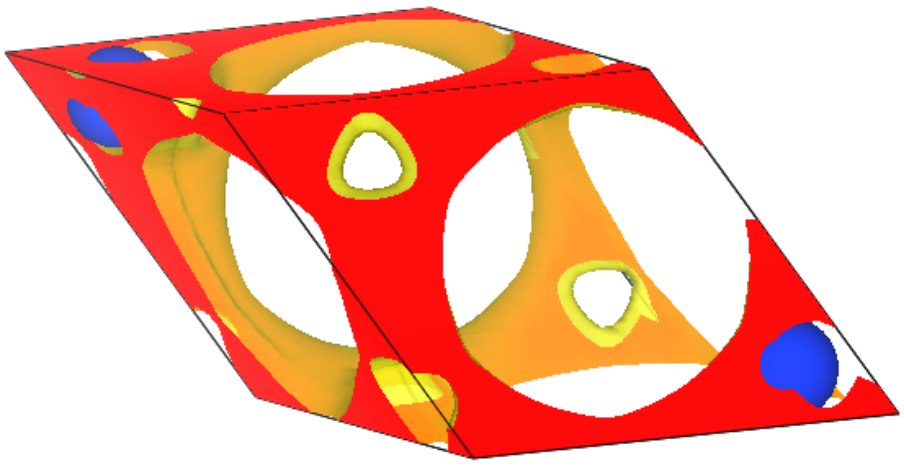
8.3 volumetricData 数据处理

8.3.1 volumetricData 可视化

- 参考 3vis_vol.py：

```python
# coding:utf-8
from dspawpy.io.write import write_VESTA

# Read data file (in h5 or json format), process it, and output to a cube file
write_VESTA(
    in_filename="tests/2.2/rho.h5",  # Path to the json or h5 file containing electronic system information
    data_type="rho",  # Data type, supported values are "rho", "potential", "elf", "pcharge", "rhoBound"
    out_filename="tests/outputs/us/DS-PAW_rho.cube",  # Output file path
    gridsize=(10, 10, 10),  # Specifies the interpolation grid size
    format="cube",  # Supported formats: cube, vesta, and txt (xyz grid coordinates + values)
)
```

将转换后的文件 *DS-PAW_rho.cube* 拖入 **VESTA** 软件中显示：



晶体硅单胞的电荷密度分布示意图

8.3.2 差分 volumetricData 可视化

- 参考 3dvol.py：

```
# coding:utf-8
from dspawpy.io.write import write_delta_rho_vesta

# Read the data file (h5 or json format), process it, and output it to a cube file, which can be directly opened with Vesta and has a small volume
write_delta_rho_vesta(
    total="tests/supplement/AB.h5",  # Data file for the system containing all components
    individuals=[
        "tests/supplement/A.h5",
        "tests/supplement/B.h5",
    ],  # Data files for the system containing each component
    output="tests/outputs/us/3delta_rho.cube",  # Output file path
)
```

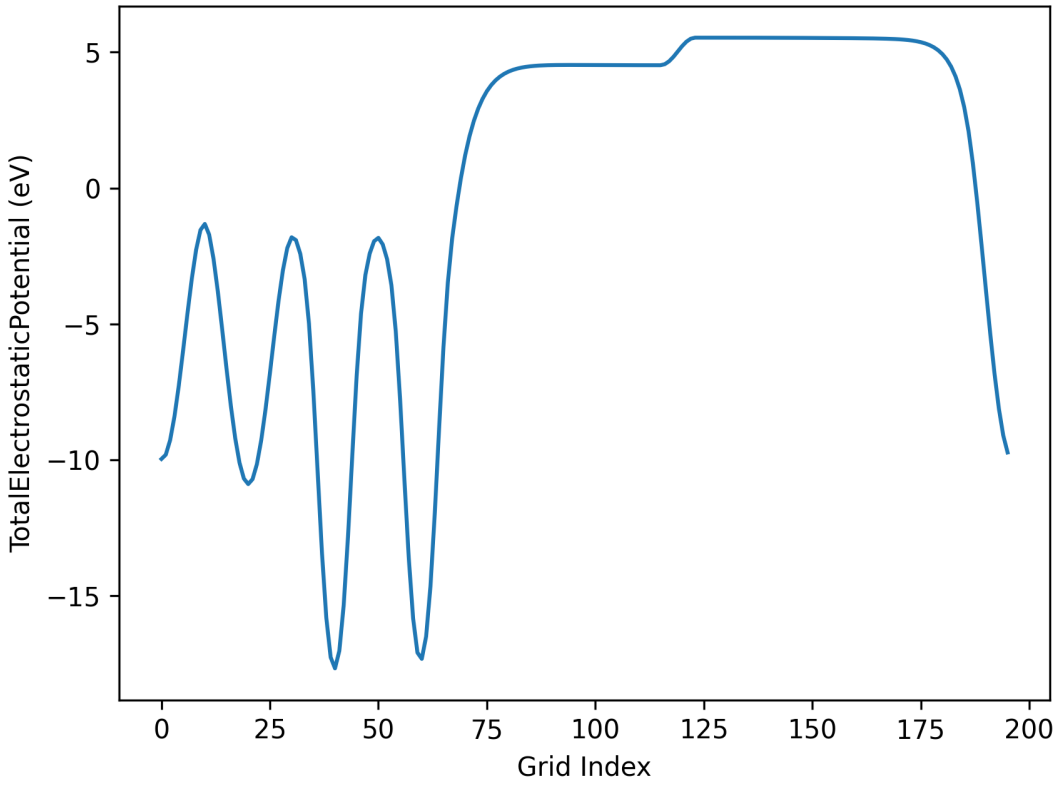
上述脚本支持处理多元体系的电荷密度差，示例以二元体系为例，得到了 AB.h5 与 A.h5 和 B.h5 之间的电荷密度差文件 *delta_rho.cube* ，该文件可直接使用 **VESTA** 打开。

8.3.3 volumetricData 面平均

- 参考 3planar_ave.py：

```
# coding:utf-8
from dspawpy.plot import average_along_axis

axes = [
    "2"
]  # "0", "1", "2" correspond to the x, y, z axes respectively; select which axes to average along
axes_indices = [int(i) for i in axes]
for ai in axes_indices:
    plt = average_along_axis(
        datafile="tests/3.3/scf.h5",  # Data file path
        task="potential",  # Task name, can be 'rho', 'potential', 'elf', 'pcharge', 'rhoBound'
        axis=ai,  # Axis along which to plot the potential curve
        smooth=False,  # Whether to smooth
        smooth_frac=0.8,  # Smoothing coefficient
        subtype=None,  # Used to specify the subclass of task data, currently only used for Potential
        label=f"axis{ai}",  # Legend label
    )
if len(axes_indices) > 1:
    plt.legend()

plt.xlabel("Grid Index")
plt.ylabel("TotalElectrostaticPotential (eV)")
plt.savefig("tests/outputs/us/3pot_ave.png", dpi=300)  # Image name
```

处理应用案例 3.3 小节所得静电势文件，可得真空方向势函数曲线如下所示：



AuAl 势函数示意图

API: write_VESTA(), write_delta_rho_vesta(), average_along_axis()

- `write_VESTA` 函数负责处理 volumetricData 可视化：

  `dspawpy.io.write.`**`write_VESTA`**(*in_filename: str*, *data_type: str*, *out_filename: str = 'DS-PAW.cube'*, *subtype: str | None = None*, *format: str | None = 'cube'*, *compact: bool = False*, *inorm: bool = False*, *gridsize: Sequence[int] | None = None*)

  Read data from a json or h5 file containing electronic system information and write to a VESTA formatted file.

  参数

  - **`in_filename`** –Path to a json or h5 file containing electronic system information
  - **`data_type`** –Data type, supported values are "rho", "potential", "elf", "pcharge", "rhoBound"
  - **`out_filename`** –Output file path, default "DS-PAW.cube"
  - **`subtype`** –Used to specify the subtype of data_type, default is None, which will read the TotalElectrostaticPotential data of potential
  - **`format`** –Output data format, supports "cube" and "vesta" ("vasp"), default is "cube", case-insensitive

- **compact** –Each data point for each grid is placed on a new line, reducing the file size by decreasing the number of spaces (this does not affect the parsing of VESTA software), default is False
- **inorm** –Whether to normalize the volume data so that the sum is 1, default is False
- **gridsize** –The redefined number of grid points, in the format (ngx, ngy, ngz), default is None, which uses the original number of grid points

返回
VESTA formatted file

返回类型
out_filename

示例

```
>>> from dspawpy.io.write import write_VESTA
>>> write_VESTA("tests/2.2/rho.json", "rho", out_filename='tests/outputs/
↪doctest/rho.cube')
==> ...rho...cube
```

```
>>> from dspawpy.io.write import write_VESTA
>>> write_VESTA(
...     in_filename="tests/2.7/potential.h5",
...     data_type="potential",
...     out_filename="tests/outputs/doctest/my_potential.cube",
...     subtype='TotalElectrostaticPotential', # or 'TotalLocalPotential'
...     gridsize=(50,50,50), # all integer, can be larger or less than the␣
↪original gridsize
... )
Interpolating volumetric data...
volumetric data interpolated
==> ...my_potential...cube
>>> write_VESTA(
...     in_filename="tests/2.8/elf.h5",
...     data_type="elf",
...     out_filename="tests/outputs/doctest/elf.cube",
... )
==> ...elf...cube
>>> write_VESTA(
...     in_filename="tests/2.9/pcharge.h5",
...     data_type="pcharge",
...     out_filename="tests/outputs/doctest/pcharge.cube",
... )
==> ...pcharge...cube
>>> write_VESTA(
...     in_filename="tests/2.7/potential.h5",
...     data_type="potential",
...     out_filename="tests/outputs/doctest/my_potential.vasp",
...     subtype='TotalElectrostaticPotential', # or 'TotalLocalPotential'
...     gridsize=(50,50,50), # all integer, can be larger or less than the␣
↪original gridsize
... )
Interpolating volumetric data...
volumetric data interpolated
==> ...my_potential...vasp
```

(续下页)

(接上页)

```
>>> write_VESTA(
...     in_filename="tests/2.8/elf.h5",
...     data_type="elf",
...     out_filename="tests/outputs/doctest/elf.vasp",
... )
==> ...elf...vasp
>>> write_VESTA(
...     in_filename="tests/2.9/pcharge.h5",
...     data_type="pcharge",
...     out_filename="tests/outputs/doctest/pcharge.vasp",
... )
==> ...pcharge...vasp
```

```
>>> write_VESTA(
...     in_filename="tests/2.7/potential.h5",
...     data_type="potential",
...     out_filename="tests/outputs/doctest/my_potential.txt",
...     subtype='TotalElectrostaticPotential', # or 'TotalLocalPotential'
...     gridsize=(50,50,50), # all integer, can be larger or less than the␣
↪original gridsize
... )
Interpolating volumetric data...
volumetric data interpolated
==> ...my_potential...txt
>>> with open("tests/outputs/doctest/my_potential.txt") as t:
...     contents = t.readlines()
...     for line in contents[:10]:
...         print(line.strip())
# 2 atoms
# 50 50 50 grid size
# x y z value
0.000 0.000 0.000      0.3279418
0.055 0.055 0.000     -0.0740864
0.110 0.110 0.000     -0.8811762
0.165 0.165 0.000     -2.1283864
0.220 0.220 0.000     -4.0559144
0.275 0.275 0.000     -6.8291031
0.330 0.330 0.000    -10.1550910
```

volumetricData 指的是随空间位置变化的物理量，如电荷密度 rho，势能函数 potential，局域电荷密度 elf，部分电荷密度 pcharge，溶剂束缚电荷密度 rhoBound 等。这些数据在 DS-PAW 中以 volumetricData 类型保存。

- `write_delta_rho_vesta` 函数负责处理差分 volumetricData 可视化：

`dspawpy.io.write.`**`write_delta_rho_vesta`**(*total: str*, *individuals: list[str]*, *output: str = 'delta_rho.cube'*, *format: str = 'cube'*, *compact: bool = False*, *inorm: bool = False*, *gridsize: Sequence | None = None*, *data_type: str | None = 'rho'*, *subtype: str | None = None*)

Charge density differential visualization

DeviceStudio does not currently support large files; it is temporarily written in a format that can be opened with VESTA.

参数

- **`total`** –Path to the total charge density file of the system, can be in h5 or json format

- **individuals** –Paths to the charge density files of each component in the system, can be in h5 or json format
- **output** –Output file path, default “delta_rho.cube”
- **format** –Output data format, supports “cube” and “vasp”, default to “cube”
- **compact** –Each data point for each grid is placed on a new line, and the file size is reduced by reducing the number of spaces (this does not affect the parsing by VESTA software), default is False
- **inorm** –Whether to normalize the volume data so that the sum is 1, default is False
- **gridsize** –Redefined grid number, format as (ngx, ngy, ngz), default is None, use the original grid number

返回

A charge density file after the difference of charges (total - individual1 - individual2 - ⋯)

返回类型

output

示例

```
>>> from dspawpy.io.write import write_delta_rho_vesta
>>> write_delta_rho_vesta(total='tests/supplement/AB.h5',
...     individuals=['tests/supplement/A.h5', 'tests/supplement/B.h5'],
...     output='tests/outputs/doctest/delta_rho.cube')
==> ...delta_rho...cube
```

- `average_along_axis` 函数负责处理 volumetricData 面平均数据：

`dspawpy.plot.`**`average_along_axis`**(*datafile: str = 'potential.h5'*, *task: str = 'potential'*, *axis: int = 2*, *smooth: bool = False*, *smooth_frac: float = 0.8*, *raw: bool = False*, *subtype: str | None = None*, *verbose: bool = False*, ***kwargs*)

Plot the average curve of a physical quantity along a certain axis

参数

- **datafile** –Path to an h5 or json file, or a folder containing any of these files, default ‘potential.h5’
- **task** –Task type, can be ‘rho’, ‘potential’, ‘elf’, ‘pcharge’, ‘rhoBound’
- **axis** –Along which axis to plot the potential curve, default is 2
- **smooth** –Whether to smooth, default False
- **smooth_frac** –Smoothing coefficient, default 0.8
- **raw** –Whether to return plot data to a CSV file
- **subtype** –Used to specify the task data subtype, default None, representing drawing Potential/TotalElectrostaticPotential
- ****kwargs** –Other parameters, passed to matplotlib.pyplot.plot

返回

Can be passed to other functions for further processing

返回类型

axes

示例

```
>>> from dspawpy.plot import average_along_axis
```

Read data from the potential.h5 file, plot, and save the original plot data to a CSV file

```
>>> plt = average_along_axis(datafile='tests/3.3/rho.h5', task='rho', axis=2,␣
↪smooth=True, smooth_frac=0.8)
>>> plt.savefig('tests/outputs/doctest/rho_h5.png')
>>> plt = average_along_axis(datafile='tests/3.3/rho.json', task='rho',␣
↪axis=2, smooth=True, smooth_frac=0.8)
>>> plt.savefig('tests/outputs/doctest/rho_json.png')
```

```
>>> plt = average_along_axis(datafile='tests/2.7/potential.h5', task=
↪'potential', axis=2, smooth=True, smooth_frac=0.8, raw=True)
>>> plt.savefig('tests/outputs/doctest/potential_h5.png')
>>> plt = average_along_axis(datafile='tests/2.7/potential.json', task=
↪'potential', axis=2, smooth=True, smooth_frac=0.8)
>>> plt.savefig('tests/outputs/doctest/potential_json.png')
```

```
>>> plt = average_along_axis(datafile='tests/2.8/elf.h5', task='elf', axis=2,␣
↪smooth=True, smooth_frac=0.8)
>>> plt.savefig('tests/outputs/doctest/elf_h5.png')
>>> plt = average_along_axis(datafile='tests/2.8/elf.json', task='elf',␣
↪axis=2, smooth=True, smooth_frac=0.8)
>>> plt.savefig('tests/outputs/doctest/elf_json.png')
```

```
>>> plt = average_along_axis(datafile='tests/2.9/pcharge.h5', task='pcharge',␣
↪axis=2, smooth=True, smooth_frac=0.8)
>>> plt.savefig('tests/outputs/doctest/pcharge_h5.png')
>>> plt = average_along_axis(datafile='tests/2.9/pcharge.json', task='pcharge
↪', axis=2, smooth=True, smooth_frac=0.8)
>>> plt.savefig('tests/outputs/doctest/pcharge_json.png')
```

```
>>> plt = average_along_axis(datafile='tests/2.28/rhoBound.h5', task='rhoBound
↪', axis=2, smooth=True, smooth_frac=0.8)
>>> plt.savefig('tests/outputs/doctest/rhoBound_h5.png')
>>> plt = average_along_axis(datafile='tests/2.28/rhoBound.json', task=
↪'rhoBound', axis=2, smooth=True, smooth_frac=0.8)
>>> plt.savefig('tests/outputs/doctest/rhoBound_json.png')
```

警告

如果通过 SSH 连接到远程服务器执行上述脚本，出现 QT 相关的报错信息，可能是使用的程序（比如 MobaXterm 等）和 QT 库不兼容，要么更换程序（例如 VSCode 或者系统自带的终端命令行），要么请在 python 脚本第二行开始添加以下代码：

```
import matplotlib
matplotlib.use('agg')
```

8.4 band 能带数据处理

知识点：

1. 脚本中调用 get_band_data() 读取数据，在读取数据时设置 efermi=XX 可将能量零点修改为指定值；设置 zero_to_efermi=True, 可将能量零点修改问所读取的文件中的费米能级处
2. 脚本调用 pymatgen 的 BSPlotter.get_plot() 画图，在画图时可设置 zero_to_efermi=True，将坐标轴能量零点修改到费米能级处。由于 pymatgen 在 2023.8.17 的一处关键更新将绘图函数返回对象从 plt 改成了 axes，导致后续脚本可能出现不兼容。因此用户脚本相关部分已添加一个判断语句加以处理。
3. 能带两步算需从第一步自洽计算获取准确费米能级（从自洽获得的 system.json），若获取失败，用户可在调用 get_band_data 读取数据时，利用参数 efermi 修改能量零点。例如：band_data=get_band_data('band.h5',efermi=-1.5)
4. 脚本中画图调用 pymatgen 中的 BSPlotter.get_plot, 当判断体系为非金属时，设置 zero_to_efermi 会认为 VBM 为费米能级能量而非文件读取时的费米能级。故当体系为非金属时，在数据读取时设置 zero_to_efermi=True 和在画图时设置 zero_to_efermi=True 得到的图会有区别

执行本节所列的 python 脚本，程序会判断是否为金属体系。若为非金属体系，将要求选择是否平移费米能级到能量零点，请按提示操作。

8.4.1 普通能带处理

参考 `4bandplot.py`：

```python
# coding:utf-8
import os
import matplotlib.pyplot as plt
from pymatgen.electronic_structure.plotter import BSPlotter

from dspawpy.io.read import get_band_data

datafile = "tests/supplement/pband.h5"  # Specifies the data file path
band_data = get_band_data(
    band_dir=datafile,
    syst_dir=None,  # path to system.json file, required only when band_dir is a json file
    efermi=None,  # Used for manually correcting the Fermi level
    zero_to_efermi=True,  # For non-metallic systems, the zero point energy should be shifted to the Fermi level
)

bsp = BSPlotter(band_data)
axes_or_plt = bsp.get_plot(
    zero_to_efermi=False,  # The data has already been shifted when read, so this should be turned off
    ylim=[-10, 10],  # Range of the y-axis for the band structure plot
    smooth=False,  # Whether to smooth the band structure plot
    vbm_cbm_marker=False,  # Whether to mark the valence band maximum and conduction band minimum in the band structure plot
    smooth_tol=0,  # Threshold for smoothing
    smooth_k=3,  # Order of the smoothing process
    smooth_np=100,  # Number of points for smoothing
```

(续下页)

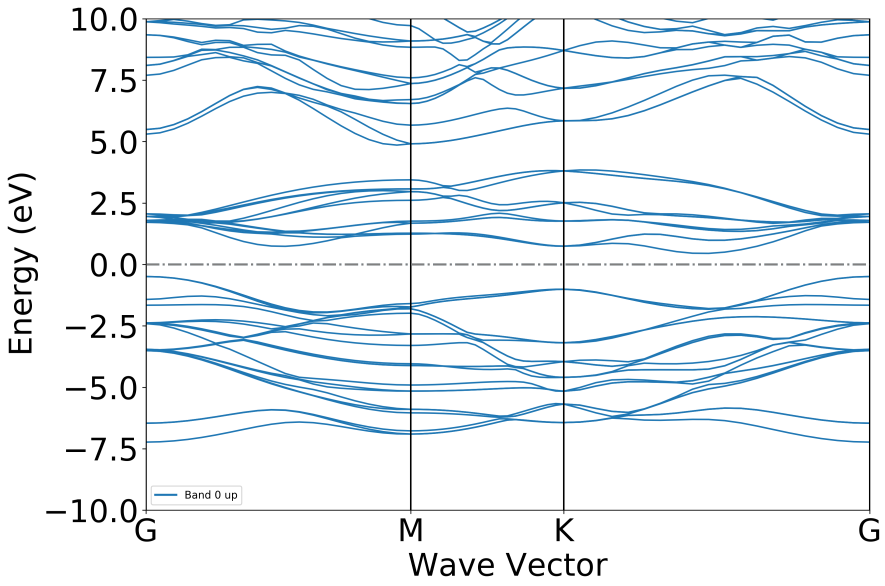
(接上页)

```
)

if isinstance(axes_or_plt, plt.Axes):
    fig = axes_or_plt.get_figure()  # version newer than v2023.8.10
else:
    fig = axes_or_plt.gcf()  # older version pymatgen

# Add a reference line for the energy zero point
for ax in fig.axes:
    ax.axhline(0, lw=2, ls="-.", color="gray")

figname = "tests/outputs/us/4bandplot.png"  # Filename for the output band plot
os.makedirs(os.path.dirname(os.path.abspath(figname)), exist_ok=True)
fig.savefig(figname, dpi=300)
```

知识点：

能带两步算需从自洽计算获取准确的费米能级（从 system.json 获取），若获取失败，用户可在 get_band_data 函数中指定 efermi 参数修改

执行代码可以得到类似以下能带图：



二硫化钼能带示意图

8.4.2 将能带投影到每一种元素分别作图，数据点大小表示该元素对该轨道的贡献

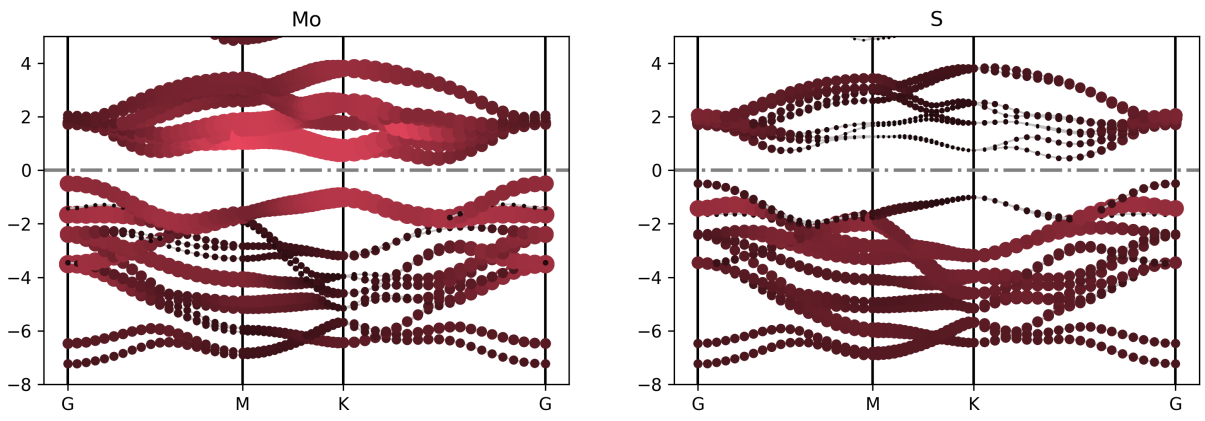
参考 4bandplot_elt.py：

```python
# coding:utf-8
import os

import matplotlib.pyplot as plt
import numpy as np
from pymatgen.electronic_structure.plotter import BSPlotterProjected

from dspawpy.io.read import get_band_data

datafile = "tests/supplement/pband.h5"  # Specify the data file path
band_data = get_band_data(
    band_dir=datafile,
    syst_dir=None,  # path to system.json file, required only when band_dir is a json file
    efermi=None,  # Used to manually adjust the Fermi level
    zero_to_efermi=True,  # For non-metallic systems, shift the zero-point energy to the Fermi level
)

bsp = BSPlotterProjected(bs=band_data)  # Initialize the BSPlotterProjected class
axes_or_plt = bsp.get_elt_projected_plots(
    zero_to_efermi=False,  # The data has already been shifted when read, so this should be disabled
    ylim=[-8, 5],  # Set the energy range
    vbm_cbm_marker=False,  # Whether to mark the conduction band minimum (CBM) and valence band maximum (VBM)
)

if isinstance(axes_or_plt, plt.Axes):
    fig = axes_or_plt.get_figure()  # version newer than v2023.8.10
elif np.iterable(axes_or_plt):
    fig = np.asarray(axes_or_plt).flatten()[0].get_figure()
else:
    fig = axes_or_plt.gcf()  # older version pymatgen

# Add a reference line for the energy zero point
for ax in fig.axes:
    ax.axhline(0, lw=2, ls="-.", color="gray")

figname = "tests/outputs/us/4bandplot_elt.png"  # The filename for the output band plot
os.makedirs(os.path.dirname(os.path.abspath(figname)), exist_ok=True)
fig.savefig(figname, dpi=300)
```

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

执行代码可以得到类似以下能带图：



二硫化钼元素投影能带示意图

警告

如果通过 SSH 连接到远程服务器执行上述脚本，出现 QT 相关的报错信息，可能是使用的程序（比如 MobaXterm 等）和 QT 库不兼容，要么更换程序（例如 VSCode 或者系统自带的终端命令行），要么请在 python 脚本第二行开始添加以下代码：

```
import matplotlib
matplotlib.use('agg')
```

8.4.3 能带投影到不同元素的不同轨道

参考 4bandplot_elt_orbit.py：

```
# coding:utf-8
import os

import matplotlib.pyplot as plt
import numpy as np
from pymatgen.electronic_structure.plotter import BSPlotterProjected

from dspawpy.io.read import get_band_data

datafile = "tests/supplement/pband.h5"  # Specify the data file path
band_data = get_band_data(
    band_dir=datafile,
    syst_dir=None,  # path to system.json file, only required when band_dir is a json file
    efermi=None,  # Used for manually correcting the Fermi level
    zero_to_efermi=True,  # For non-metallic systems, shift the zero point energy to the Fermi level
)

bsp = BSPlotterProjected(bs=band_data)  # Initialize the BSPlotterProjected class
# Select elements and orbitals, create a dictionary
dict_elem_orbit = {"Mo": ["d"], "S": ["s"]}

```

(续下页)

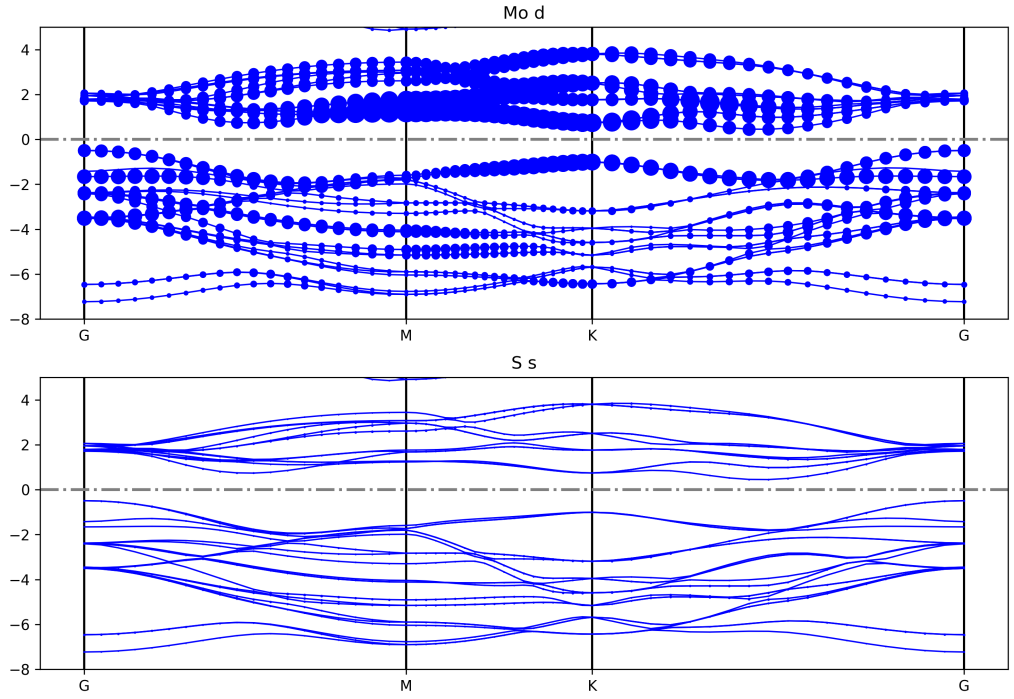
(接上页)

```
axes_or_plt = bsp.get_projected_plots_dots(
    dictio=dict_elem_orbit,
    zero_to_efermi=False,  # The data has already been shifted when read, so this should be turned off
    ylim=[-8, 5],  # Set the energy range
    vbm_cbm_marker=False,  # Whether to mark the conduction band minimum and valence band maximum
)

if isinstance(axes_or_plt, plt.Axes):
    fig = axes_or_plt.get_figure()  # version newer than v2023.8.10
elif np.iterable(axes_or_plt):
    fig = np.asarray(axes_or_plt).flatten()[0].get_figure()
else:
    fig = axes_or_plt.gcf()  # older version pymatgen

# Add a reference line for the energy zero point
for ax in fig.axes:
    ax.axhline(0, lw=2, ls="-.", color="gray")

figname = "tests/outputs/us/4bandplot_elt_orbit.png"  # Filename for the output band plot
os.makedirs(os.path.dirname(os.path.abspath(figname)), exist_ok=True)
fig.savefig(figname, dpi=300)
```

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

执行代码可以得到类似以下能带图：



二硫化钼元素轨道投影能带示意图

警告

如果通过 SSH 连接到远程服务器执行上述脚本，出现 QT 相关的报错信息，可能是使用的程序（比如 MobaXterm 等）和 QT 库不兼容，要么更换程序（例如 VSCode 或者系统自带的终端命令行），要么请在 python 脚本第二行开始添加以下代码：

```
import matplotlib
matplotlib.use('agg')
```

8.4.4 将能带投影到不同原子的不同轨道

参考 4bandplot_patom_porbit.py：

```
# coding:utf-8
import os

import matplotlib.pyplot as plt
import numpy as np
from pymatgen.electronic_structure.plotter import BSPlotterProjected

from dspawpy.io.read import get_band_data

datafile = "tests/supplement/pband.h5"  # Specify the data file path
```

(续下页)

(接上页)

```
band_data = get_band_data(
    band_dir=datafile,
    syst_dir=None,  # path to system.json file, required only when band_dir is a JSON file
    efermi=None,  # Used to manually adjust the Fermi level
    zero_to_efermi=True,  # For non-metallic systems, shift the zero point energy to the Fermi level
)

bsp = BSPlotterProjected(bs=band_data)
# Specify elements, orbitals, and atomic numbers
dict_elem_orbit = {"Mo": ["px", "py", "pz"]}
dict_elem_index = {"Mo": [1]}

axes_or_plt = bsp.get_projected_plots_dots_patom_pmorb(
    dictio=dict_elem_orbit,  # Specify the element-orbit dictionary
    dictpa=dict_elem_index,  # Specify the element-atomic number dictionary
    sum_atoms=None,  # Whether to sum over atoms
    sum_morbs=None,  # Whether to sum orbitals
    zero_to_efermi=False,  # Data has already been shifted during reading, should be turned off here
    ylim=None,  # Set the energy range
    vbm_cbm_marker=False,  # Whether to mark the conduction band minimum and valence band maximum
    selected_branches=None,  # Specify the energy band branches to be plotted
    w_h_size=(12, 8),  # Set image width and height
    num_column=None,  # Number of images displayed per row
)

if isinstance(axes_or_plt, plt.Axes):
    fig = axes_or_plt.get_figure()  # version newer than v2023.8.10
elif np.iterable(axes_or_plt):
    fig = np.asarray(axes_or_plt).flatten()[0].get_figure()
else:
    fig = axes_or_plt.gcf()  # older version pymatgen

# Add a reference line for the energy zero point
for ax in fig.axes:
    ax.axhline(0, lw=2, ls="-.", color="gray")

figname = "tests/outputs/us/4band_patom_porbit.png"  # Output bandpass figure filename
os.makedirs(os.path.dirname(os.path.abspath(figname)), exist_ok=True)
fig.savefig(figname, dpi=300)
```

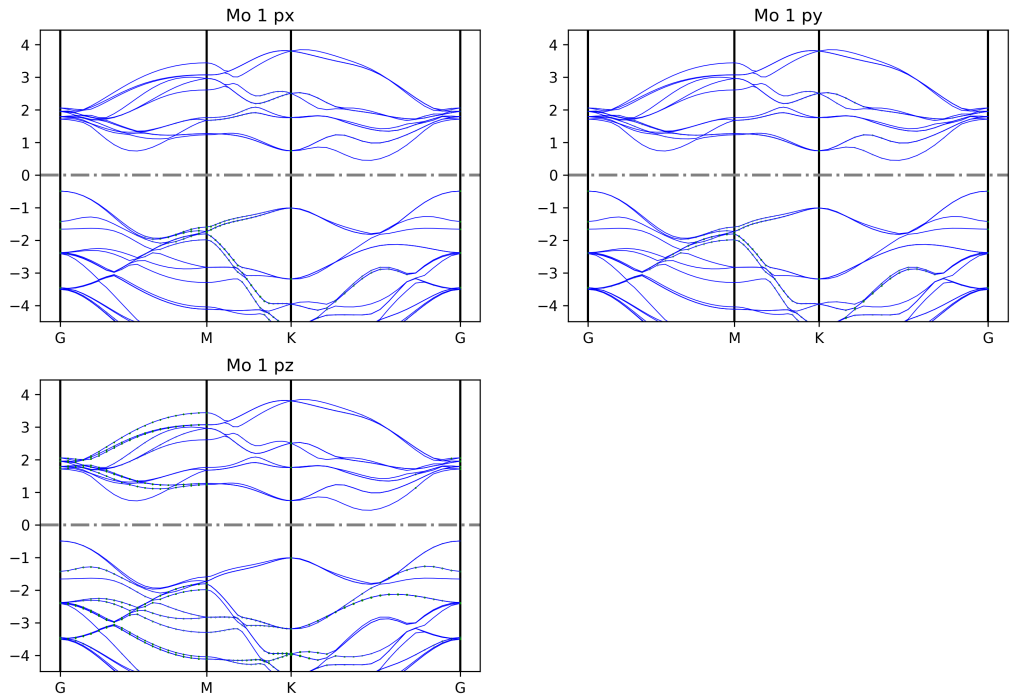
知识点：

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些原子某些轨道的能带图
2. dictpa 指定原子，dictio 指定该原子的轨道
3. 如果要将一些原子或一些轨道的投影分量叠加起来，请根据 get_projected_plots_dots_patom_pmorb 函数文档指定 sum_atoms 或 sum_morbs 参数

警告

1. 如果只选择单个轨道，且轨道名称不止一个字母（例如 px、dxy、dxz），get_projected_plots_dots_patom_pmorb 函数将报错，详情见 此处

执行代码可以得到类似以下能带图：



二硫化钼原子投影能带示意图

警告

如果通过 SSH 连接到远程服务器执行上述脚本，出现 QT 相关的报错信息，可能是使用的程序（比如 MobaXterm 等）和 QT 库不兼容，要么更换程序（例如 VSCode 或者系统自带的终端命令行），要么请在 python 脚本第二行开始添加以下代码：

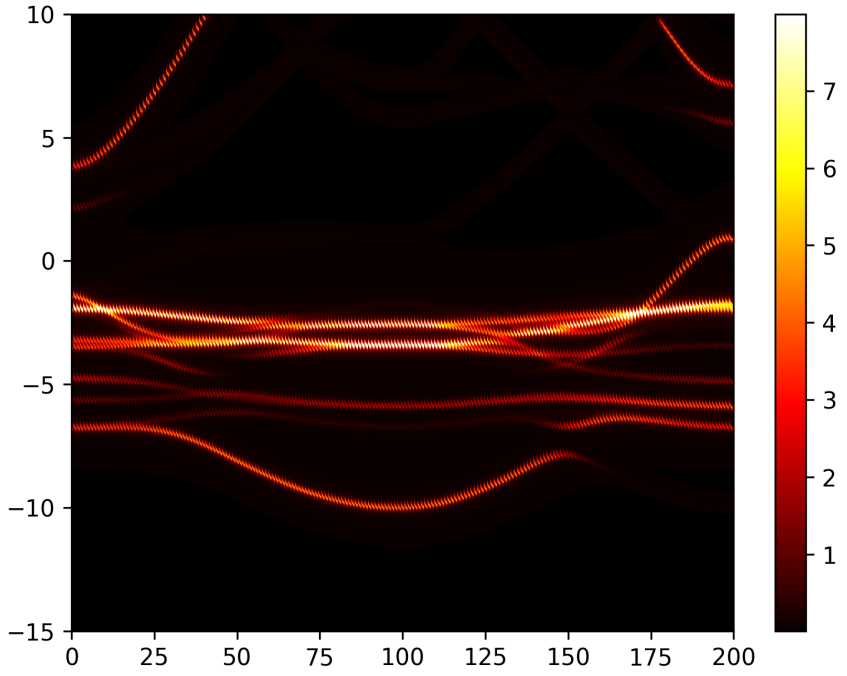
```
import matplotlib
matplotlib.use('agg')
```

8.4.5 能带反折叠处理

参考 4bandunfolding.py：

```
# coding:utf-8
import os

from dspawpy.plot import plot_bandunfolding

plt = plot_bandunfolding(
    datafile="tests/2.22.1/band.h5",  # Read data
    ef=None,  # Fermi level, read from the file
    de=0.05,  # Band width, default 0.05
    dele=0.06,  # Band gap, default 0.06
)

plt.ylim(-15, 10)
figname = "tests/outputs/us/4bandunfolding.png"  # Output band structure plot filename
os.makedirs(os.path.dirname(os.path.abspath(figname)), exist_ok=True)
plt.savefig(figname, dpi=300)
# plt.show()
```

执行代码可以得到类似以下能带图：



Cu3Au 能带反折叠示意图

警告

此功能暂不支持将非金属材料的费米能级设为能量零点（默认价带顶为能量零点）。

警告

如果通过 SSH 连接到远程服务器执行上述脚本，出现 QT 相关的报错信息，可能是使用的程序（比如 MobaXterm 等）和 QT 库不兼容，要么更换程序（例如 VSCode 或者系统自带的终端命令行），要么请在 python 脚本第二行开始添加以下代码：

```
import matplotlib
matplotlib.use('agg')
```

8.4.6 band-compare 能带对比图处理

将普通能带和 wannier 能带绘制在同一张图上

参考 4bandcompare.py：

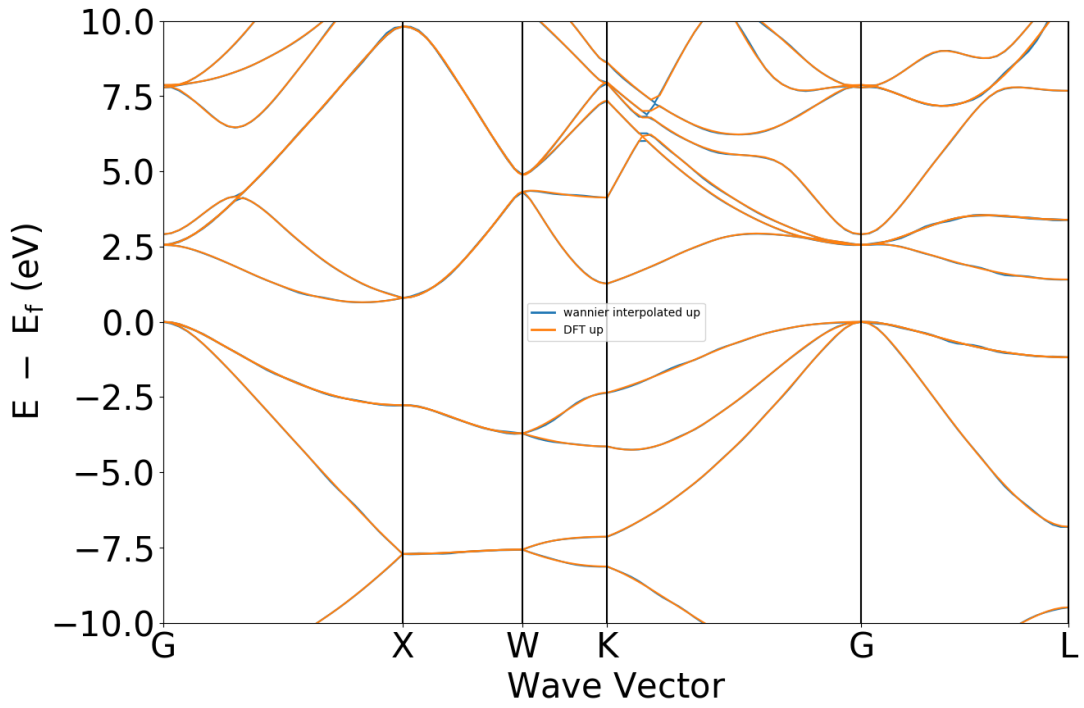
```
# coding:utf-8
import os

from pymatgen.electronic_structure.plotter import BSPlotter

from dspawpy.io.read import get_band_data

band_data = get_band_data(
    band_dir="tests/2.30/wannier.h5",  # Wannier band file path
    syst_dir=None,  # system.json file path, only needed when band_dir is a json file
    efermi=None,  # Used for manually adjusting the Fermi level
    zero_to_efermi=False,  # Whether to shift zero energy to the Fermi level
)
bsp = BSPlotter(bs=band_data)
band_data = get_band_data(
    band_dir="tests/2.3/band.h5",  # Read DFT band structure
    syst_dir=None,  # path to system.json file, required only when band_dir is a json file
    efermi=None,  # Used for manually correcting the Fermi level
    zero_to_efermi=False,  # Whether to shift the zero point energy to the Fermi level
)

bsp2 = BSPlotter(bs=band_data)
bsp.add_bs(bsp2._bs)
axes_or_plt = bsp.get_plot(
    zero_to_efermi=True,  # Move the zero energy level to the Fermi level
    ylim=[-10, 10],  # Energy band plot y-axis range
    smooth=False,  # Whether to smooth the band structure plot
    vbm_cbm_marker=False,  # Whether to mark the valence band maximum and conduction band minimum in the band structure plot
    smooth_tol=0,  # Threshold for smoothing
    smooth_k=3,  # Order of the smoothing process
    smooth_np=100,  # Number of points for smoothing
    bs_labels=["wannier interpolated", "DFT"],  # Band structure labels
)

```

(续下页)

(接上页)

```
import matplotlib.pyplot as plt  # noqa: E402

if isinstance(axes_or_plt, plt.Axes):
    fig = axes_or_plt.get_figure()  # version newer than v2023.8.10
else:
    fig = axes_or_plt.gcf()  # older version pymatgen

# Add a reference line for the energy zero point
for ax in fig.axes:
    ax.axhline(0, lw=2, ls="-.", color="gray")

figname = "tests/outputs/us/4wanierBand.png"  # File name for the output band structure plot
os.makedirs(os.path.dirname(os.path.abspath(figname)), exist_ok=True)
fig.savefig(figname, dpi=300)
```

执行代码可得到类似如下能带对比曲线：



晶体硅瓦尼尔能带与 DFT 能带示意图

API: get_band_data()

- `get_band_data` 函数负责读取能带数据如下：

  `dspawpy.io.read.`**`get_band_data`**(*band_dir: str*, *syst_dir: str | None = None*, *efermi: float | None = None*, *zero_to_efermi: bool = False*, *verbose: bool = False*) → BandStructureSymmLine

  Reads band structure data from an h5 or json file and constructs a BandStructureSymmLine object.

  参数

- **band_dir** –
  - Path to the band structure file, band.h5 / band.json, or a directory containing band.h5 / band.json
  - Note that wannier.h5 can also be read using this function, but band_dir does not support folder types
- **syst_dir** –Path to system.json, prepared only for auxiliary processing of Wannier data (structure and Fermi level are read from it)
- **efermi** –Fermi level, if the Fermi level in the h5 file is incorrect, it can be specified using this parameter
- **zero_to_efermi** –Whether to shift the Fermi level to 0

返回类型
BandStructureSymmLine

示例

```
>>> from dspawpy.io.read import get_band_data
>>> band = get_band_data(band_dir='tests/2.3/band.h5')
>>> band = get_band_data(band_dir='tests/2.4/band.h5')
>>> band = get_band_data(band_dir='tests/2.4/band.json')
```

If you want to process Wannier band structures by specifying wannier.json, you need to additionally specify the syst_dir parameter.

```
>>> band = get_band_data(band_dir='tests/2.30/wannier.h5')
>>> band = get_band_data(band_dir='tests/2.30/wannier.json', syst_dir='tests/
↪2.30/system.json')
```

警告

如果通过 SSH 连接到远程服务器执行上述脚本，出现 QT 相关的报错信息，可能是使用的程序（比如 MobaXterm 等）和 QT 库不兼容，要么更换程序（例如 VSCode 或者系统自带的终端命令行），要么请在 python 脚本第二行开始添加以下代码：

```
import matplotlib
matplotlib.use('agg')
```

8.5 dos 态密度数据处理

8.5.1 总的态密度

参考 5dosplot_total.py：

```
# coding:utf-8
import os

from pymatgen.electronic_structure.plotter import DosPlotter

```

(续下页)

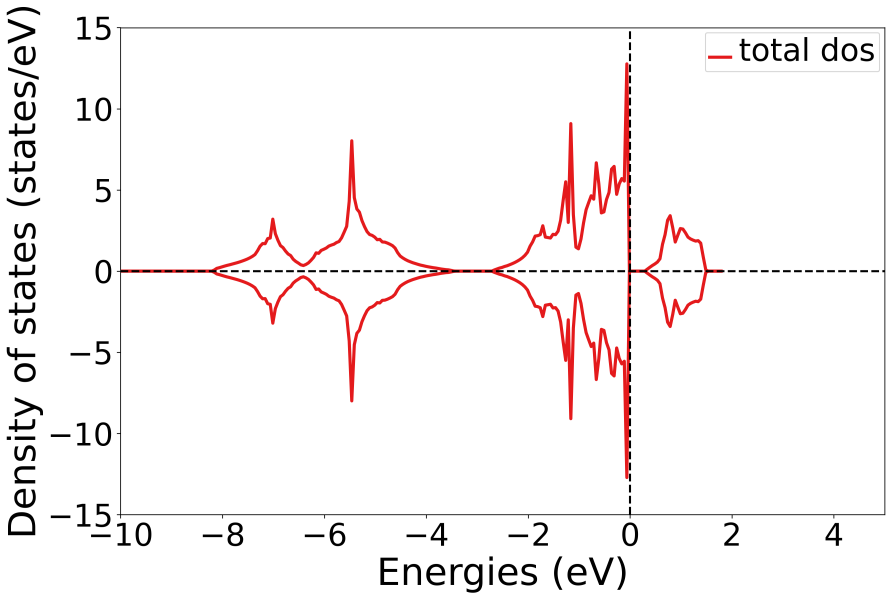
(接上页)

```python
from dspawpy.io.read import get_dos_data
from dspawpy.plot import plot_dos

dos_data = get_dos_data(
    dos_dir="tests/3.2.4/dos.h5",  # Read projected density of states data
    return_dos=False,  # If False, always return a CompleteDos object (regardless of whether projection was enabled during calculation)
)
dos_plotter = DosPlotter(
    zero_at_efermi=True,  # Whether to set the Fermi level as the zero point
    stack=False,  # True indicates drawing an area chart
    sigma=None,  # Gaussian broadening, None indicates no smoothing process
)
dos_plotter.add_dos(
    label="total dos", dos=dos_data
)  # Set the legend for the density of states plot  # Pass the density of states data

ax = plot_dos(
    dosplotter=dos_plotter,
    xlim=[-10, 5],  # Set the energy range
    ylim=[-15, 15],  # Set the density of states range
)
ax.axhline(0, lw=2, ls="-.", color="gray")

filename = "tests/outputs/us/5dos_total.png"  # File name for the output density of states plot
os.makedirs(os.path.dirname(os.path.abspath(filename)), exist_ok=True)

fig = ax.get_figure()
fig.savefig(filename, dpi=300)
```

知识点：

1. 使用 get_dos_data 函数可以将 DS-PAW 计算得到的 dos.h5 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.h5 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 get_plot 函数绘制态密度图

执行代码可以得到类似以下态密度图：



NiO 态密度示意图

警告

如果通过 SSH 连接到远程服务器执行上述脚本，出现 QT 相关的报错信息，可能是使用的程序（比如 MobaXterm 等）和 QT 库不兼容，要么更换程序（例如 VSCode 或者系统自带的终端命令行），要么请在 python 脚本第二行开始添加以下代码：

```
import matplotlib
matplotlib.use('agg')
```

8.5.2 将态密度投影到不同的轨道上

参考 5dosplot_spd.py：

```
# coding:utf-8
import os

from pymatgen.electronic_structure.plotter import DosPlotter

from dspawpy.io.read import get_dos_data
from dspawpy.plot import plot_dos

dos_data = get_dos_data(
    dos_dir="tests/3.2.4/dos.h5",  # Read projected DOS data
    return_dos=False,  # If False, always return a CompleteDos object (regardless of␣
↪whether projection was enabled during calculation)
)
dos_plotter = DosPlotter(
    zero_at_efermi=True,  # Whether to set the Fermi level as the zero point
    stack=False,  # True indicates drawing an area chart
    sigma=None,  # Gaussian broadening, None indicates no smoothing is applied
)
```

(续下页)

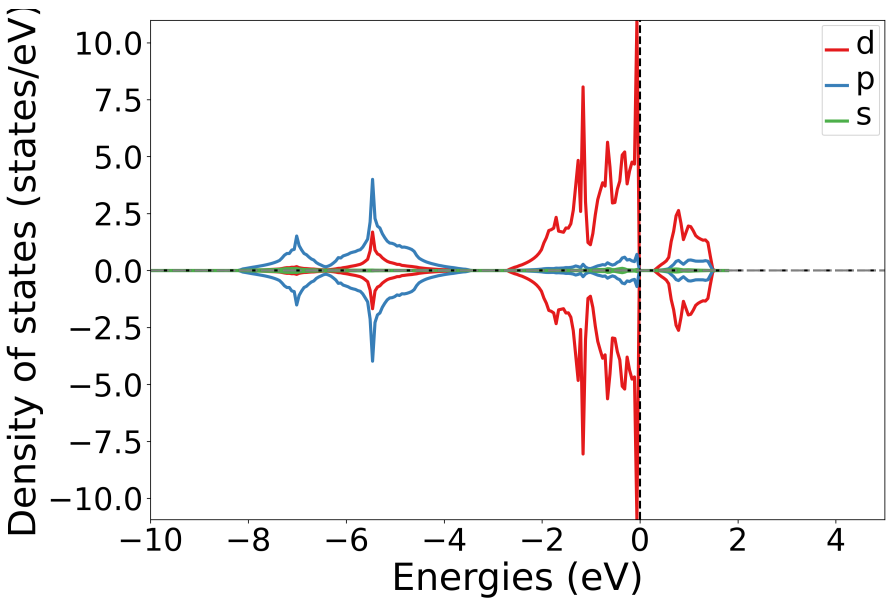
(接上页)

```
dos_plotter.add_dos_dict(
    dos_dict=dos_data.get_spd_dos(),
    key_sort_func=None,  # Orbital projection  # Specifies the sorting function
)
ax = plot_dos(
    dosplotter=dos_plotter,
    xlim=[-10, 5],  # Set the energy range
    ylim=None,  # Set the density of states range
)

ax.axhline(0, lw=2, ls="-.", color="gray")

filename = "tests/outputs/us/5dos_spd.png"  # Filename of the output density of states plot
os.makedirs(os.path.dirname(os.path.abspath(filename)), exist_ok=True)

fig = ax.get_figure()
fig.savefig(filename, dpi=300)
```

知识点：

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

执行代码可以得到类似以下态密度图：



NiO 轨道投影态密度示意图

警告

如果通过 SSH 连接到远程服务器执行上述脚本，出现 QT 相关的报错信息，可能是使用的程序（比如 MobaXterm 等）和 QT 库不兼容，要么更换程序（例如 VSCode 或者系统自带的终端命令行），要么请

在 python 脚本第二行开始添加以下代码：

```
import matplotlib
matplotlib.use('agg')
```

8.5.3 将态密度投影到不同的元素上

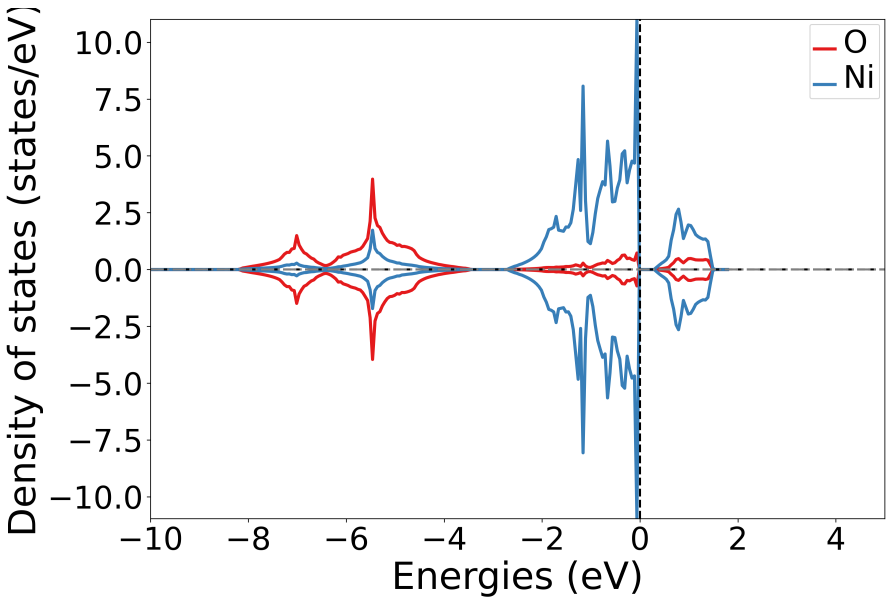
参考 5dosplot_elt.py：

```
# coding:utf-8
import os

from pymatgen.electronic_structure.plotter import DosPlotter

from dspawpy.io.read import get_dos_data
from dspawpy.plot import plot_dos

dos_data = get_dos_data(
    dos_dir="tests/3.2.4/dos.h5",  # Reads projected DOS data
    return_dos=False,  # If False, always returns a CompleteDos object (regardless of whether projection was enabled during calculation)
)
dos_plotter = DosPlotter(
    zero_at_efermi=True,  # Whether to set the Fermi level as the zero point
    stack=False,  # True indicates drawing an area chart
    sigma=None,  # Gaussian broadening, None indicates no smoothing is applied
)
dos_plotter.add_dos_dict(
    dos_dict=dos_data.get_element_dos(),
    key_sort_func=None,  # Projected DOS for elements  # Specify the sorting function
)

ax = plot_dos(
    dosplotter=dos_plotter,
    xlim=[-10, 5],  # Set the energy range
    ylim=None,  # Set the density of states range
)
ax.axhline(0, lw=2, ls="-.", color="gray")

filename = "tests/outputs/us/5dos_elt.png"  # Filename for the output density of states plot
os.makedirs(os.path.dirname(os.path.abspath(filename)), exist_ok=True)

fig = ax.get_figure()
fig.savefig(filename, dpi=300)
```

知识点：

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

执行代码可以得到类似以下态密度图：



NiO 元素投影态密度示意图

警告

如果通过 SSH 连接到远程服务器执行上述脚本，出现 QT 相关的报错信息，可能是使用的程序（比如 MobaXterm 等）和 QT 库不兼容，要么更换程序（例如 VSCode 或者系统自带的终端命令行），要么请在 python 脚本第二行开始添加以下代码：

```
import matplotlib
matplotlib.use('agg')
```

8.5.4 将态密度投影到不同原子的不同轨道上

参考 5dosplot_atom_orbit.py：

```
# coding:utf-8
import os

from pymatgen.electronic_structure.core import Orbital
from pymatgen.electronic_structure.plotter import DosPlotter

from dspawpy.io.read import get_dos_data
from dspawpy.plot import plot_dos

dos_data = get_dos_data(
    dos_dir="tests/3.2.4/dos.h5",  # Reads projected density of states data
    return_dos=False,  # If False, always return a CompleteDos object (regardless of whether projection was enabled during calculation)
)
dos_plotter = DosPlotter(
    zero_at_efermi=True,  # Whether to set the Fermi level as the zero point
    stack=False,  # True indicates drawing an area plot
    sigma=None,  # Gaussian broadening, None indicates no smoothing treatment
```

(续下页)

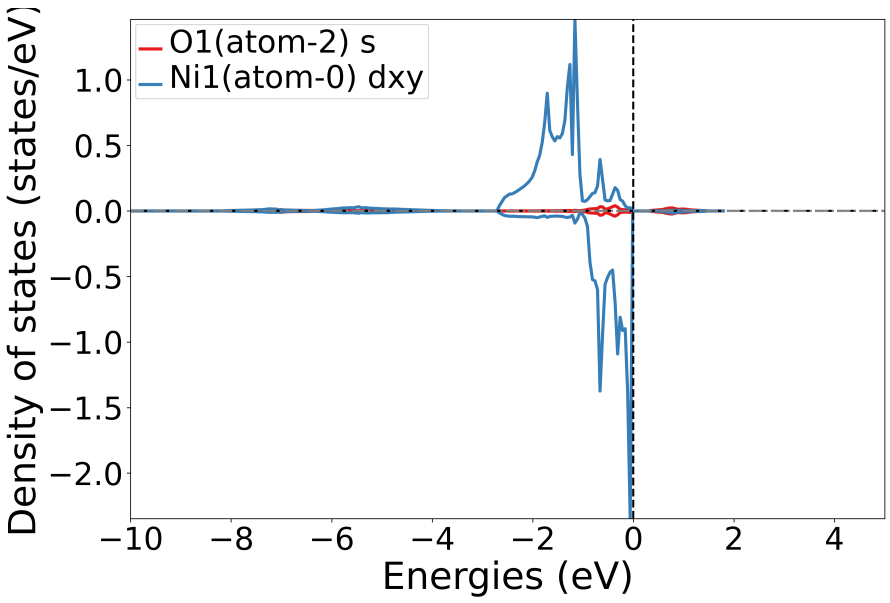
(接上页)

```
)

# ! Specify atomic number and orbital
dict_index_orbit = {0: ["dxy"], 2: ["s"]}

print("Plotting...")
for index in dict_index_orbit:
    _os = dict_index_orbit[index]
    _e = str(dos_data.structure.sites[index].species)
    for _orb in _os:
        dos_plotter.add_dos(
            f"{_e}(atom-{index}) {_orb}",  # label
            dos_data.get_site_orbital_dos(
                dos_data.structure[index],
                getattr(Orbital, _orb),
            ),
        )

ax = plot_dos(
    dosplotter=dos_plotter,
    xlim=[-10, 5],  # Set the energy range
    ylim=None,  # Set the density of states range
)
ax.axhline(0, lw=2, ls="-.", color="gray")

figname = "tests/outputs/us/5dos_atom_orbit.png"  # Output density of states figure␣
↪filename
os.makedirs(os.path.dirname(os.path.abspath(figname)), exist_ok=True)

fig = ax.get_figure()
fig.savefig(figname, dpi=300)
```

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，dos_data.structure[0],Orbital(4) 表示获取第 1 个原子的 dxy 轨道的态密度，get_site_orbital_dos 函数中序号从 0 开始
2. 运行此脚本，根据提示选择元素和轨道，可以得到相应的态密度图

执行代码可以得到类似以下态密度图：



NiO 原子轨道态密度示意图

警告

如果通过 SSH 连接到远程服务器执行上述脚本，出现 QT 相关的报错信息，可能是使用的程序（比如 MobaXterm 等）和 QT 库不兼容，要么更换程序（例如 VSCode 或者系统自带的终端命令行），要么请在 python 脚本第二行开始添加以下代码：

```
import matplotlib
matplotlib.use('agg')
```

8.5.5 将态密度投影到不同原子的分裂 d 轨道 (t2g, eg) 上

参考 5dosplot_t2g_eg.py：

```
# coding:utf-8
import os

from pymatgen.electronic_structure.plotter import DosPlotter

from dspawpy.io.read import get_dos_data
from dspawpy.plot import plot_dos

dos_data = get_dos_data(
    dos_dir="tests/3.2.4/dos.h5",  # Read projected density of states data
    return_dos=False,  # If False, always returns a CompleteDos object (regardless of whether projection was enabled during calculation)
)
dos_plotter = DosPlotter(
    zero_at_efermi=True,  # Whether to set the Fermi level as the zero point
    stack=False,  # True indicates drawing an area chart
    sigma=None,  # Gaussian broadening, None indicates no smoothing is applied
)
```

(续下页)

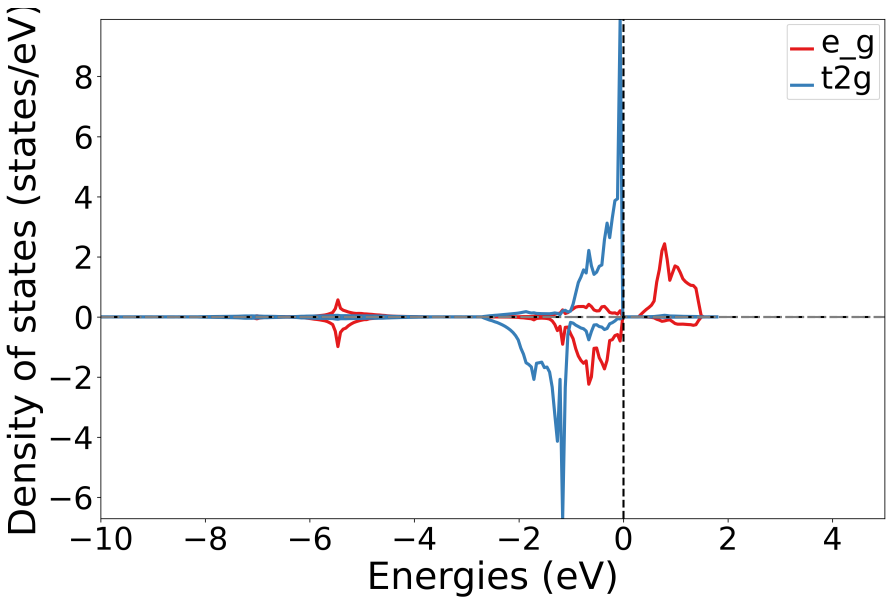
(接上页)

```python
# print(dos_data.structure)

# Specify the atomic number, starting from 0
ais = [1]

print("Plotting...")
atom_indices = [int(ai) for ai in ais]
for atom_index in atom_indices:
    dos_plotter.add_dos_dict(
        dos_data.get_site_t2g_eg_resolved_dos(dos_data.structure[atom_index]),
    )

ax = plot_dos(
    dosplotter=dos_plotter,
    xlim=[-10, 5],  # Set the energy range
    ylim=None,  # Set the density of states range
)
ax.axhline(0, lw=2, ls="-.", color="gray")

filename = "tests/outputs/us/5dos_t2g_eg.png"  # Output density of states plot filename
os.makedirs(os.path.dirname(os.path.abspath(filename)), exist_ok=True)

fig = ax.get_figure()
fig.savefig(filename, dpi=300)
```

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 2 个原子的 t2g 和 eg 轨道的贡献
2. 运行此脚本，根据提示选择原子序号，可以得到相应的态密度图

执行代码可以得到类似以下态密度图：



NiO 分裂 d 轨道原子投影态密度示意图

知识点：

如果元素不含 d 轨道，会画成空白图片

警告

如果通过 SSH 连接到远程服务器执行上述脚本，出现 QT 相关的报错信息，可能是使用的程序（比如 MobaXterm 等）和 QT 库不兼容，要么更换程序（例如 VSCode 或者系统自带的终端命令行），要么请在 python 脚本第二行开始添加以下代码：

```
import matplotlib
matplotlib.use('agg')
```

8.5.6 d-带中心分析

以 Pb-slab 体系为例，对 Pt 原子进行 d-band 中心分析：

参考 `5center_dband.py`：

```
# coding:utf-8
from dspawpy.io.read import get_dos_data
from dspawpy.io.utils import d_band

dos_data = get_dos_data(
    dos_dir="tests/supplement/dos.h5",  # Read projected density of states data
    return_dos=False,  # If False, always returns a CompleteDos object (regardless of whether projection was enabled during calculation)
)
for spin in dos_data.densities:
    print("spin=", spin)
    c = d_band(spin, dos_data)
    print(c)
```

执行代码可以得到类似以下结果：

```
spin=1
-1.785319344084034
```

备注

目前仅支持全部原子平均的 d 轨道中心，不支持元素、原子投影或其他轨道，也不支持选择自旋方向或能量范围。

`get_dos_data` 函数负责处理态密度数据：

API: get_dos_data()

dspawpy.io.read.**get_dos_data**(*dos_dir: str*, *return_dos: bool = False*, *verbose: bool = False*)

Read density of states (DOS) data from an h5 or json file, and construct a CompleteDos or DOS object

参数

- **dos_dir** –Path to the density of states file, dos.h5 / dos.json, or a folder containing dos.h5 / dos.json
- **return_dos** (*bool, optional*) –Whether to return the DOS object. If False, a CompleteDos object is returned uniformly (regardless of whether projection was enabled during calculation)

返回类型

CompleteDos or Dos

示例

```
>>> from dspawpy.io.read import get_dos_data
>>> dos = get_dos_data(dos_dir='tests/2.5/dos.h5')
>>> dos = get_dos_data(dos_dir='tests/2.5/dos.h5', return_dos=True)
```

8.6 bandDos 能带和态密度共同显示

以应用教程中 Si 体系为例：

8.6.1 将能带和态密度显示在一张图上

参考 6bandDosplot.py：

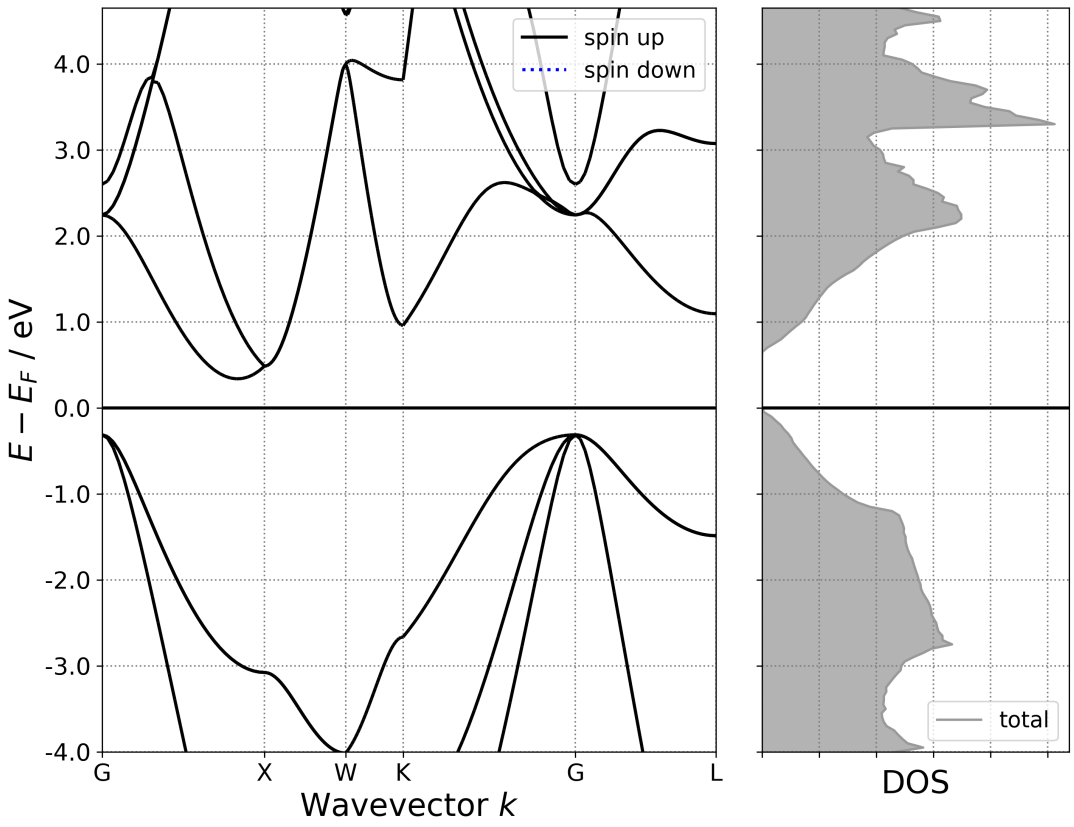
```
# coding:utf-8
import os

import numpy as np
from matplotlib.axes import Axes
from pymatgen.electronic_structure.plotter import BSDOSPlotter

from dspawpy.io.read import get_band_data, get_dos_data

bandfile = "tests/2.3/band.h5"  # Normal band data
band_data = get_band_data(
    band_dir=bandfile,
    syst_dir=None,  # path to system.json file, required only when band_dir is a json file
    efermi=None,  # Used for manually correcting the Fermi level
)
band_efermi = band_data.efermi
dosfile = "tests/2.5/dos.h5"  # Density of states data
dos_data = get_dos_data(
    dos_dir=dosfile,
    return_dos=False,  # If False, always return a CompleteDos object (regardless of whether projection was enabled during calculation)
```

(续下页)

(接上页)

```
)
dos_efermi = dos_data.efermi
bdp = BSDOSPlotter(
    bs_projection=None,  # Band structure projection method, None means no projection
    dos_projection=None,  # Projection method for density of states, None means no projection
    vb_energy_range=4,  # Valence band energy range
    cb_energy_range=4,  # Conduction band energy range
    fixed_cb_energy=False,  # Whether to fix the conduction band energy range
    egrid_interval=1,  # Energy grid interval
    font="DejaVu Sans",  # Default is Times New Roman, change to DejaVu Sans to avoid warnings due to missing font on Linux
    axis_fontsize=20,  # Axis font size
    tick_fontsize=15,  # Tick label font size
    legend_fontsize=14,  # Legend font size
    bs_legend="best",  # Band structure legend position
    dos_legend="best",  # Density of States legend position
    rgb_legend=True,  # Use colored legend
    fig_size=(11, 8.5),  # Figure size
)
if band_efermi != dos_efermi:
    print(f"{band_efermi=:.4f} eV")
    print(f"{dos_efermi=:.4f} eV")
    d_efermi = band_efermi - dos_efermi

    print(
        "! Band and DOS Fermi levels are inconsistent, using DOS Fermi level as reference"
    )
    band_data.bands = {spin: v + d_efermi for spin, v in band_data.bands.items()}

    # ! Band and DOS Fermi levels are inconsistent, using Band level as the reference
    # dos_data.energies -= d_efermi

axes_or_plt = bdp.get_plot(
    bs=band_data, dos=dos_data
)  # Pass band data  # Pass density of states data

if isinstance(axes_or_plt, Axes):
    fig = axes_or_plt.get_figure()  # version newer than v2023.8.10
elif np.iterable(axes_or_plt):
    fig = np.asarray(axes_or_plt).flatten()[0].get_figure()
else:
    fig = axes_or_plt.gcf()  # older version pymatgen

filename = "tests/outputs/us/6bandDos.png"  # Filename for the band structure - density of states plot output
os.makedirs(os.path.dirname(os.path.abspath(filename)), exist_ok=True)
fig.savefig(filename, dpi=300)
print("==> Saved", filename)
```

执行代码可以得到类似以下能带态密度图：



单晶硅能带-态密度示意图

警告

如果通过 SSH 连接到远程服务器执行上述脚本，出现 QT 相关的报错信息，可能是使用的程序（比如 MobaXterm 等）和 QT 库不兼容，要么更换程序（例如 VSCode 或者系统自带的终端命令行），要么请在 python 脚本第二行开始添加以下代码：

```
import matplotlib
matplotlib.use('agg')
```

8.6.2 将能带和投影态密度显示在一张图上

参考 6bandPdosplot.py：

```
# coding:utf-8
import os

import numpy as np
from matplotlib.axes import Axes
```

(续下页)

(接上页)

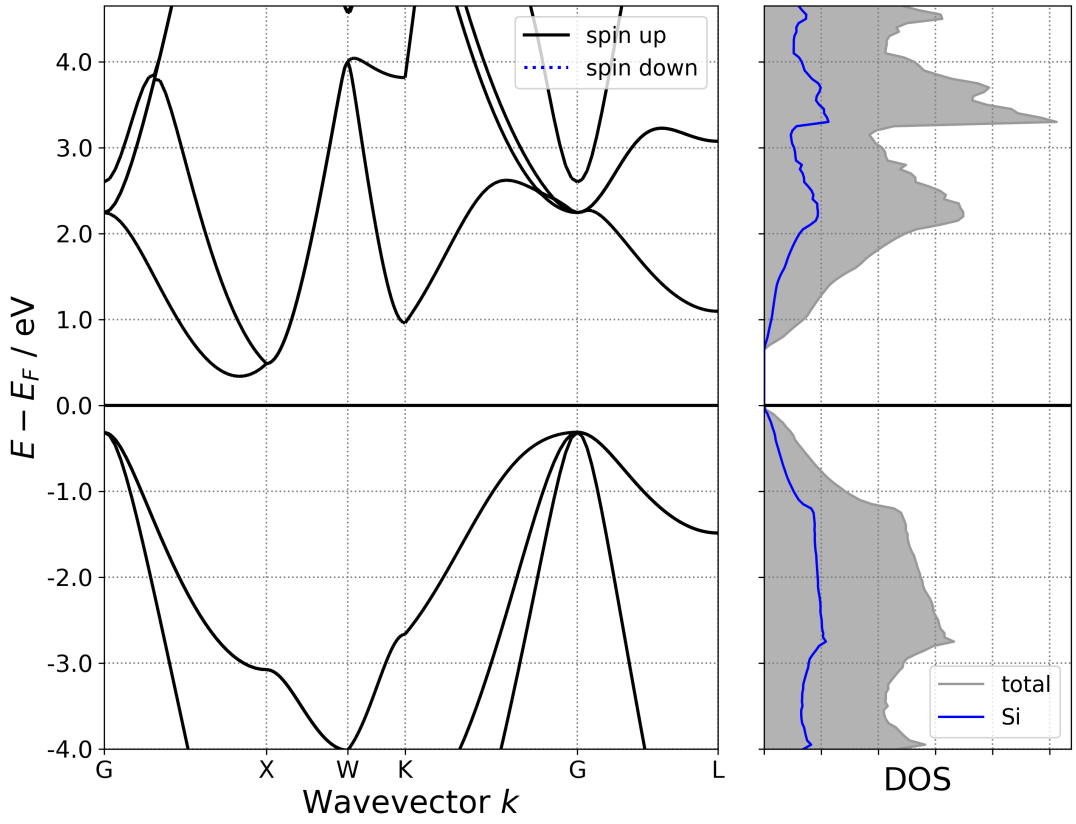
```
from pymatgen.electronic_structure.plotter import BSDOSPlotter

from dspawpy.io.read import get_band_data, get_dos_data

bandfile = "tests/2.4/band.h5"  # Normal band data
band_data = get_band_data(
    band_dir=bandfile,
    syst_dir=None,  # path to system.json file, required only when band_dir is a json file
    efermi=None,  # Used for manually correcting the Fermi level
)
band_efermi = band_data.efermi
dosfile = (
    "tests/2.6/dos.h5"  # DOS data for projected states
)
dos_data = get_dos_data(
    dos_dir=dosfile,
    return_dos=False,  # If False, always return a CompleteDos object (regardless of whether projection was enabled during calculation)
)
dos_efermi = dos_data.efermi
bdp = BSDOSPlotter(
    bs_projection="elements",  # Projection method for band structure, None means no projection
    dos_projection="elements",  # Project DOS onto elements
    vb_energy_range=4,  # Valence band energy range
    cb_energy_range=4,  # Conduction band energy range
    fixed_cb_energy=False,  # Whether to fix the conduction band energy range
    egrid_interval=1,  # Energy grid interval
    font="DejaVu Sans",  # Default is Times New Roman, can be changed to DejaVu Sans to avoid warnings due to font not being installed on Linux
    axis_fontsize=20,  # Axis font size
    tick_fontsize=15,  # Tick label font size
    legend_fontsize=14,  # Legend font size
    bs_legend="best",  # Band structure legend position
    dos_legend="best",  # Position of the projected density of states legend
    rgb_legend=True,  # Use colored legend
    fig_size=(11, 8.5),  # Figure size
)
if band_efermi != dos_efermi:
    print(f"{band_efermi=:.4f} eV")
    print(f"{dos_efermi=:.4f} eV")
    d_efermi = band_efermi - dos_efermi

    print(
        "! Band and DOS Fermi levels are inconsistent, using DOS Fermi level as reference"
    )
    band_data.bands = {spin: v + d_efermi for spin, v in band_data.bands.items()}

    # ! Band and DOS Fermi levels are inconsistent, using Band level as reference
    # dos_data.energies -= d_efermi

axes_or_plt = bdp.get_plot(
    bs=band_data,
    dos=dos_data,
)  # Pass band structure data  # Pass projected density of states data
```

(续下页)

(接上页)

```

if isinstance(axes_or_plt, Axes):
    fig = axes_or_plt.get_figure()  # version newer than v2023.8.10
elif np.iterable(axes_or_plt):
    fig = np.asarray(axes_or_plt).flatten()[0].get_figure()
else:
    fig = axes_or_plt.gcf()  # older version pymatgen

filename = "tests/outputs/us/6bandPdos.png"  # filename for the band structure-projected density of states plot
os.makedirs(os.path.dirname(os.path.abspath(filename)), exist_ok=True)
fig.savefig(filename, dpi=300)
print("==> Saved", filename)
```

执行代码可以得到类似以下能带态密度图：



单晶硅能带-投影态密度示意图

警告

1. 给定投影能带数据，默认将沿着元素投影；给定普通能带数据（或体系所含元素种类超过 4 种），则

不投影，并输出警告

2. 给定投影态密度数据，默认也将沿着元素投影，可以换成沿着轨道投影，或者不投影；给定普通态密度数据且未关闭态密度投影选项 BSDOSPlotter(dos_projection=None)，pymatgen 绘图程序将报错，这就是上面特意准备了一个 6bandDosplot.py 文件的原因。

警告

如果通过 SSH 连接到远程服务器执行上述脚本，出现 QT 相关的报错信息，可能是使用的程序（比如 MobaXterm 等）和 QT 库不兼容，要么更换程序（例如 VSCode 或者系统自带的终端命令行），要么请在 python 脚本第二行开始添加以下代码：

```
import matplotlib
matplotlib.use('agg')
```

8.7 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 *scf.h5* 为例（注：输出文件名与 task 一致，task = scf，io.optical = true 可计算光学性质）：

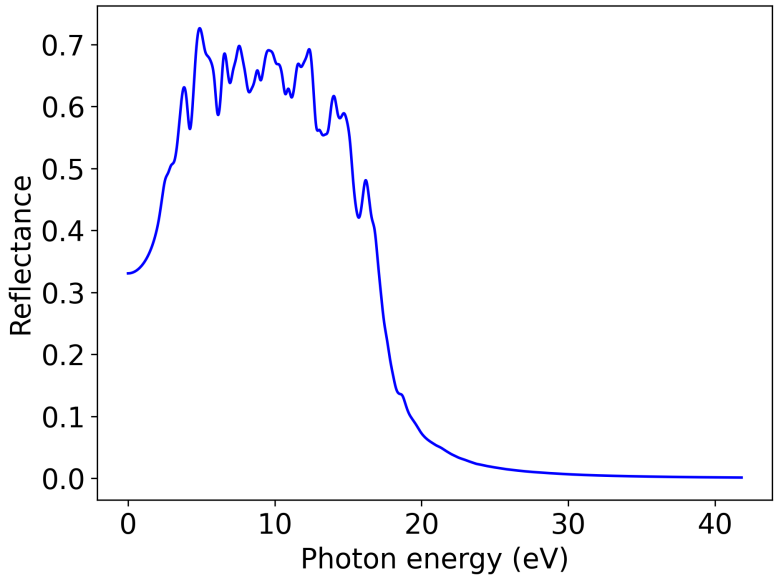
反射率数据处理，参考 `7optical.py`：

```
# coding:utf-8
from dspawpy.plot import plot_optical

plot_optical(
    datafile="tests/2.12/scf.h5",
    keys=["ExtinctionCoefficient", "Reflectance"],
    axes=["X"],  # ["X", "Y", "Z", "XY", "YZ", "ZX"]
    prefix="tests/outputs/optical",  # Where to save, if empty, it means the current folder
    save=True,  # Whether to save the image with the tool's name, if False, please refer to the script below to save manually
)

# The above function will plot and save the images of ExtinctionCoefficient and Reflectance separately
# To plot multiple properties on the same figure, uncomment the following code and set the save parameter above to False

# import os
# import matplotlib.pyplot as plt
#
# plt.tick_params(labelsize=16)
# plt.tight_layout()
# filename = "outputs/us/7optical.png"  # Filename for the output optical properties plot
# os.makedirs(os.path.dirname(os.path.abspath(filename)), exist_ok=True)
# plt.savefig(filename, dpi=300)
```

知识点：

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

执行代码可以得到类似以下反射率随能量变化的曲线：



单晶硅 Reflectance 随光子能量变化趋势示意图

API: plot_optical()

`dspawpy.plot.`**`plot_optical`**(*datafile: str = 'optical.h5', keys: List[str] = ['AbsorptionCoefficient', 'ExtinctionCoefficient', 'RefractiveIndex', 'Reflectance'], axes: List[str] = ['X', 'Y', 'Z', 'XY', 'YZ', 'ZX'], raw: bool = False, prefix: str = '', save: bool = True, verbose: bool = False*)

After the optical property calculation task is completed, read the data and draw a preview image

optical.h5/optical.json -> optical.png

参数

- **`datafile`** –Path to an h5 or json file, or a folder containing any of these files, default ‘optical.h5’
- **`keys`** –One of “AbsorptionCoefficient”, “ExtinctionCoefficient”, “RefractiveIndex”, “Reflectance”, default “AbsorptionCoefficient”
- **`axes`** –Index, default “X”, “Y”, “Z”, “XY”, “YZ”, “ZX”
- **`raw`** –Whether to save plot data to CSV
- **`prefix`** –Folder path to save images, if empty, saves in the current directory
- **`save`** –Whether to save the image, default is True

示例

Plot and save the plot data to rawoptical.csv

```
>>> from dspawpy.plot import plot_optical
>>> plot_optical("tests/2.12/scf.h5", "AbsorptionCoefficient", ['X', 'Y'], prefix=
↪'tests/outputs/doctest')
>>> plot_optical("tests/2.12/optical.json", ["AbsorptionCoefficient"], ['X', 'Y'],
↪ prefix='tests/outputs/doctest', raw=True)
```

警告

如果通过 SSH 连接到远程服务器执行上述脚本，出现 QT 相关的报错信息，可能是使用的程序（比如 MobaXterm 等）和 QT 库不兼容，要么更换程序（例如 VSCode 或者系统自带的终端命令行），要么请在 python 脚本第二行开始添加以下代码：

```
import matplotlib
matplotlib.use('agg')
```

8.8 neb 过渡态计算数据处理

以快速入门 H 在 Pt(100) 表面扩散为例进行介绍：

8.8.1 输入文件之生成中间构型

- 参考 8neb_interpolate_structures.py：

```
# coding:utf-8
from dspawpy.diffusion.neb import NEB, write_neb_structures
from dspawpy.diffusion.nebtools import write_json_chain
from dspawpy.io.structure import read

# Read initial configuration
init_struct = read("tests/2.15/00/structure00.as")[0]
# Read final state configuration
final_struct = read("tests/2.15/04/structure04.as")[0]

neb = NEB(
    initial_structure=init_struct,  # Initial structure
    final_structure=final_struct,  # Final state configuration
    nimages=8,  # Total of 8 configurations, including initial and final states
)
structures = neb.linear_interpolate()  # Linear interpolation
# structures = neb.idpp_interpolate()   # IDPP interpolation

# Save as structure file to dest path
write_neb_structures(
    structures=structures,  # Insert interpolated structure chains
    coords_are_cartesian=True,  # Whether to save in Cartesian coordinates
    fmt="as",  # Save format, supported formats: 'json', 'as', 'hzw', 'pdb', 'xyz
↪', 'dump'
    path="tests/outputs/us/8neb_interpolate_structures",  # Save path
```

(续下页)

(接上页)

```
    prefix="structure",  # File name prefix
)

# Preview initial structure chain
write_json_chain(
    preview=True,  # whether to enable preview mode
    directory="tests/outputs/us/8neb_interpolate_structures",  # Directory for␣
↪NEB calculations
    step=-1,  # Default to saving the structure chain of the last ion step␣
↪(latest)
    dst="tests/outputs/us/8neb",  # Save path
)
# write_xyz_chain(preview=True, # Whether to run in preview mode
#                 directory="tests/outputs/us/8neb_interpolate_structures", #␣
↪NEB calculation directory
#                 step=-1, # Default to saving the structure chain of the last␣
↪ionic step (latest)
#                 dst='tests/outputs/us/8neb' # save path
# )
```

知识点：

1. 用户可以根据需要自行修改插点个数，设置为 8 得到包含 8 个结构文件的文件夹，其中中间构型为 6 个
2. neb.linear_interpolate 为线性插值方法，pbc 参数为 True 时将锁定寻找最短扩散路径，默认为 False 以增加用户控制的灵活性，这是因为
3. 举个例子，初态某原子的分数坐标是 0.2，终态变成 0.8。pbc = True 将强制设置扩散路径为 0.2 -> -0.2。pbc = False 则用户可以令程序按照 0.2 -> 0.8 的扩散路径进行插值；如果要用最短路径，手动将 0.8 改为 -0.2 即可，从而确保程序按照使用者的意图完成插值初猜。

8.8.2 绘制能垒图

8.8.2.1 neb.iniFin = true/false

当设置 neb.iniFin = true/false 时，都可使用读取 neb 计算的路径进行势垒分析（确保初末态计算文件在 neb 计算路径下）：

- 参考 `8neb_barrier_CubicSpline.py`：

```
# coding:utf-8
from dspawpy.diffusion.nebtools import plot_barrier

directory_of_neb_task = (
    "tests/2.15"  # <-- Please modify to the actual NEB path
)

# Plotting the energy barrier using CubicSpline interpolation
plot_barrier(
    directory=directory_of_neb_task,  # path of the neb task
    method="CubicSpline",  # Cubic spline interpolation
    figname="tests/outputs/us/8neb_barrier_CubicSpline.png",  # Output filename␣
```

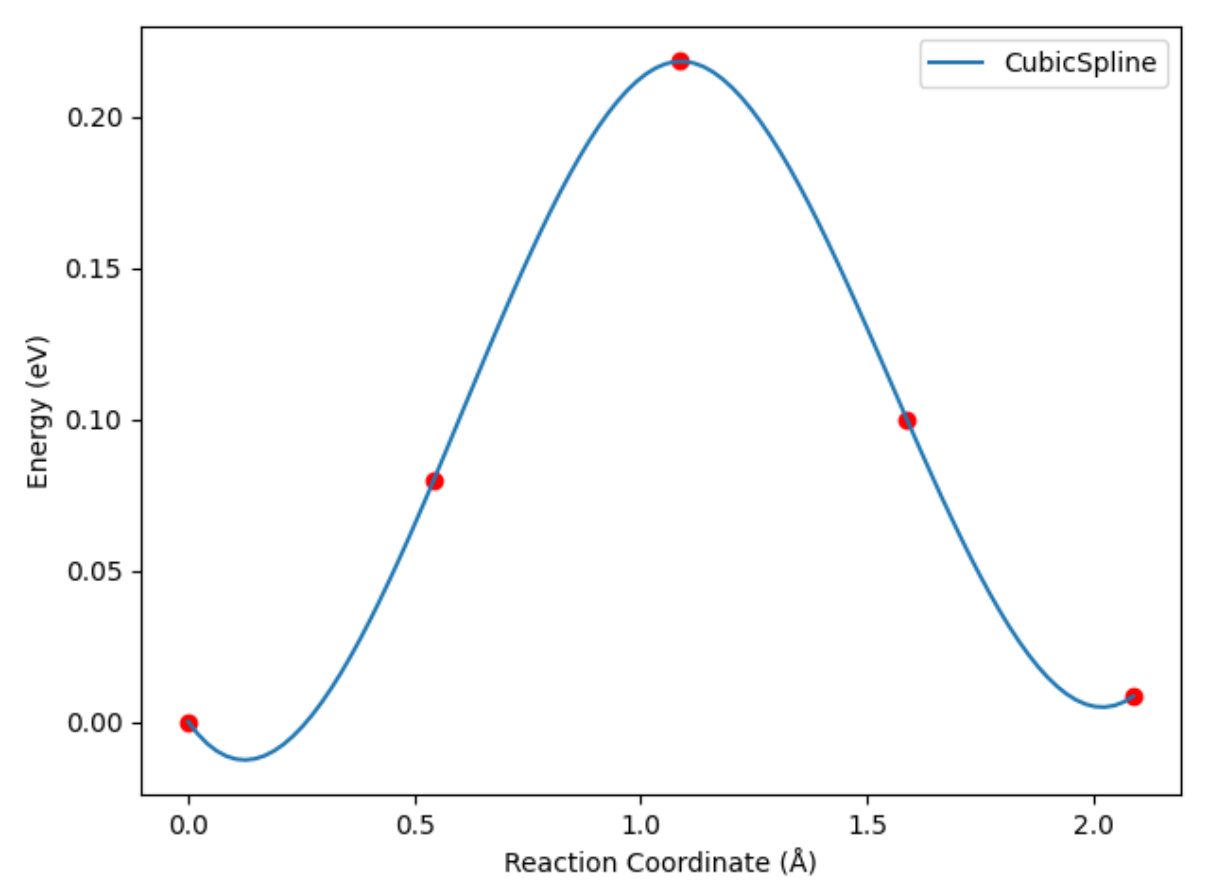
(续下页)

(接上页)

```
    ↪for the energy barrier plot
    show=False,  # Whether to display the energy barrier plot
)
```

备注

运行上述脚本后，可得到类似以下三次样条插值的势垒曲线：



对于这个算例，三次样条插值后，曲线会出现不合理的“下坠”区域，这是三次样条插值算法的特性所决定的。

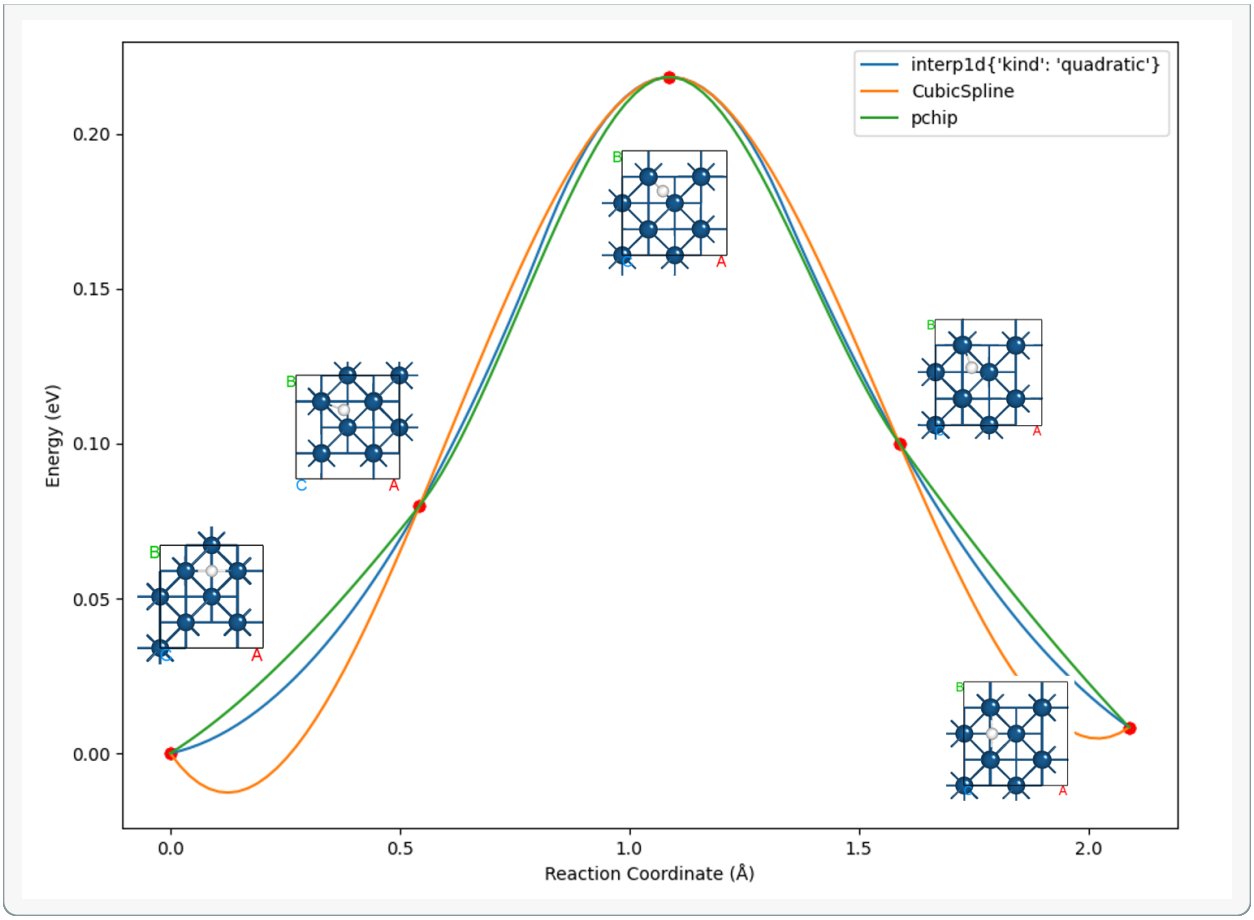
dspawpy 内部调用 scipy 的插值算法，上面这个脚本我们以三次样条插值算法为例，它在 scipy 文档中的定义为：

```
class scipy.interpolate.CubicSpline(x, y, axis=0, bc_type='not-a-knot',␣
↪extrapolate=None)
```

关键词参数包括 axis, bc_type, extrapolate，具体含义见 scipy.interpolate.CubicSpline 。我们可以往 plot_barrier 函数中指定相应的关键词参数（axis, bc_type, extrapolate），将其传递给 scipy.interpolate.CubicSpline 这个类处理。

下面我们采用 8neb_barrier.py 脚本，对比三种算法插值绘制出的曲线：

```python
# coding:utf-8
import os

import matplotlib.pyplot as plt

from dspawpy.diffusion.nebtools import plot_barrier

# Compare the differences in energy barrier curves drawn by different interpolation methods, where show should be set to False
# 1. interp1d
plot_barrier(
    directory="tests/2.15",  # path for NEB calculation
    ri=None,  # Reaction coordinate between the initial structure and the second structure, required when the NEB task only calculated intermediate structures
    rf=None,  # Reaction coordinate between the last configuration and the second-to-last configuration, when the NEB task only calculated intermediate configurations
    ei=None,  # Energy of the initial configuration, required when the NEB task only calculated intermediate configurations
    ef=None,  # Energy of the final configuration, required when the NEB task only calculated intermediate configurations
    method="interp1d",  # Interpolation method
    figname=None,  # Name of the output energy barrier plot file
    show=False,  # Whether to display the energy barrier plot
    kind="quadratic",  # Parameter of the interpolation method
)
# 2. CubicSpline
plot_barrier(
    directory="tests/2.15",
    method="CubicSpline",
    figname=None,
    show=False,
)
# 3. pchip
plot_barrier(
    directory="tests/2.15",
    method="pchip",
    figname=None,
    show=False,
)

filename = "tests/outputs/us/8neb_barrier_comparison.png"  # Filename for the energy barrier plot output
os.makedirs(os.path.dirname(os.path.abspath(filename)), exist_ok=True)
plt.savefig(filename, dpi=300)
# plt.show()
```



知识点：

1. 选择合适的插值算法对于优化最终曲线的呈现效果至关重要
2. 多数情况下，选择 pchip 这种单调三次样条插值算法即可达到较好效果，这也是默认调用的插值算法

8.8.2.2 neb.iniFin = true

当设置 neb.iniFin = true 时，读取 neb 计算所得的 neb.h5/neb.json 文件可快速进行势垒分析：

- 参考 `8neb_barrier_CubicSpline.py`：

```
# coding:utf-8
from dspawpy.diffusion.nebtools import plot_barrier

# Plot energy barrier using CubicSpline interpolation
plot_barrier(
    datafile="tests/2.15/neb.h5",  # Path to neb.h5
    method="CubicSpline",  # Cubic spline interpolation
    figname="tests/outputs/us/8neb_barrier_.png",  # Output file name for the␣
↪energy barrier plot
```

(续下页)

(接上页)

```
    show=False,  # Whether to display the energy barrier plot
)
```

处理得到的势垒图与之前读取路径效果一致。

备注

1. neb.h5 和 neb.json 文件所存能量为 TotalEnergy，如需得精确的势垒值，建议采用读取 neb 计算路径的方法进行处理（取 TotalEnergy0）

警告

如果通过 SSH 连接到远程服务器执行上述脚本，出现 QT 相关的报错信息，可能是使用的程序（比如 MobaXterm 等）和 QT 库不兼容，要么更换程序（例如 VSCode 或者系统自带的终端命令行），要么请在 python 脚本第二行开始添加以下代码：

```
import matplotlib
matplotlib.use('agg')
```

8.8.3 过渡态计算数据处理

NEB 计算完成后，一般要画出能垒图观察，并检查各插值构型最终的受力大小，从而确保受力小于指定的阈值。如果结果异常，还需要检查各插值构型在结构优化过程中的受力与能量变化趋势，判断是否真正“收敛”。上述这些操作至少需要三次循环，为简化操作，我们提供了一个一步到位的总结函数 `summary`：

- 参考 `8neb_check_results.py`：

```
# coding:utf-8
from dspawpy.diffusion.nebtools import summary

# Import the neb calculation directory, a complete folder after neb calculation needs to be provided
summary(
    directory="tests/2.15",
    show_converge=False,  # Whether to display the convergence plots of energy and force
    outdir="tests/outputs/us/8neb",  # Path to save convergence plots of energy and forces
    figname="tests/outputs/us/8neb/neb_barrier_summary.png",  # Path to save the energy barrier plot
)
# Additional keyword arguments can be set for plotting the barrier diagram, such as:
# summary(directory='tests/2.15', method='CubicSpline') # Change to CubicSpline for spline interpolation
```

知识点：

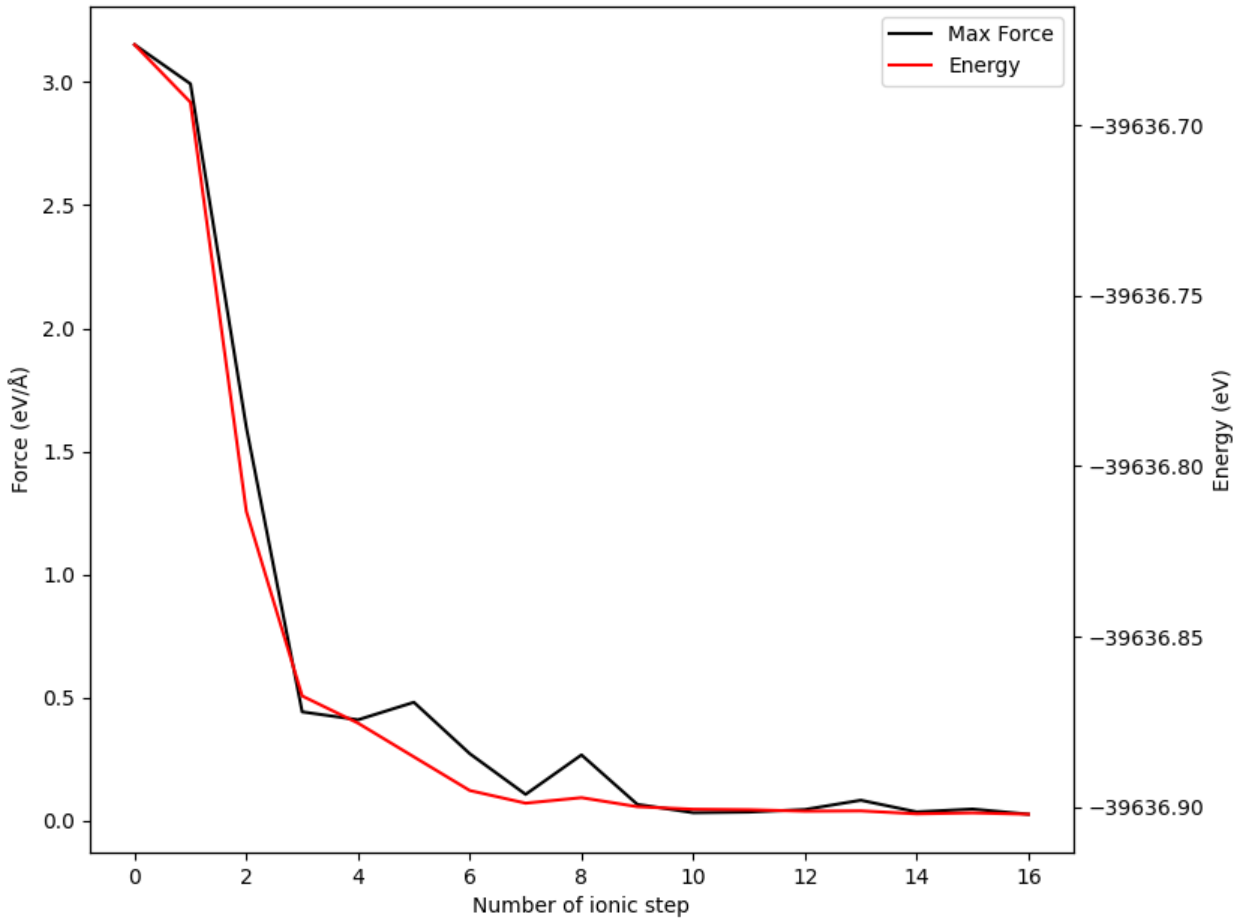
1. 此脚本将以表格形式打印各构型的能量和受力、绘制能垒图，并绘制中间构型的能量与受力收敛过程图

2. 若 `neb.iniFin = false`，用户必须将自洽计算的结果文件 *scf.h5* 或 *system.json* 文件复制到对应的初末态子文件夹中，否则程序无法读取初末态能量和受力信息，将报错退出
3. 默认情况下，能垒图存储在 neb 计算目录外层，各中间构型的能量与受力收敛图存储在各子文件夹

执行代码可以得到类似如下 NEB 计算各构型的能量和受力表格：

Image	Force (eV/Å)	Reaction coordinate (Å)	Energy (eV)	Delta energy (eV)
00	0.1803	0.0000	-39637.0984	0.0000
01	0.0263	0.5428	-39637.0186	0.0798
02	0.0248	1.0868	-39636.8801	0.2183
03	0.2344	1.5884	-39636.9984	0.1000
04	0.0141	2.0892	-39637.0900	0.0084

除了可以获得能垒图外，还可以得到各中间构型的能量和受力收敛曲线（以 02 构型为例）



警告

如果通过 SSH 连接到远程服务器执行上述脚本，出现 QT 相关的报错信息，可能是使用的程序（比

如 MobaXterm 等）和 QT 库不兼容，要么更换程序（例如 VSCode 或者系统自带的终端命令行），要么请在 python 脚本第二行开始添加以下代码：

```
import matplotlib
matplotlib.use('agg')
```

8.8.4 观察 NEB 链

此处所说 NEB 链指的是各插值构型（structure00.as, structure01.as, …）之间的几何关系，而不是单个构型在优化过程中的变化。

- NEB 任务计算量较大，观察 NEB 链有助于判断 NEB 计算的收敛速度。另外，通过插值算法生成中间结构后，经常需要预览 NEB 链。这些需求可以通过 `8neb_visualize.py` 脚本实现：

```
# coding:utf-8
from dspawpy.diffusion.nebtools import write_json_chain, write_xyz_chain

# Convert the configuration chain under the NEB calculation path to a JSON format file
write_json_chain(
    preview=False,  # If the NEB calculation is already completed, preview mode is not required
    directory="tests/2.15",  # NEB calculation directory
    step=-1,  # Default to saving the configuration chain of the last ion step (latest)
    dst="tests/outputs/us/8neb",  # Save path
    ignorels=False,  # Set to True to ignore latestStructureXX.as files
)

# Convert the configuration chain in the NEB calculation path to xyz format files
write_xyz_chain(
    preview=False,  # If the NEB calculation is already completed, preview mode is not required
    directory="tests/2.15",  # NEB calculation directory
    step=-1,  # Default to saving the configuration chain of the last ionic step (latest)
    dst="tests/outputs/us/8neb",  # Save path
    ignorels=False,  # Set to True to ignore latestStructureXX.as files
)
```

知识点：

1. 此脚本生成 neb_movie*.json 文件后，通过 `Device Studio -> Simulator -> DS-PAW -> Analysis Plot` 打开 json 文件即可观察
2. directory 指定为 NEB 计算主路径，需提供 neb 计算完成后的完整文件夹
3. 该脚本支持处理正在进行中的（即未完成的）neb 计算文件，方便用户监测实时轨迹
4. xyz 文件可通过 OVITO 软件打开查看：通过 `Device Studio -> Simulator -> OVITO` 打开可视化界面，将 xyz 文件拖入即可
5. 结构信息读取优先级：latestStructureXX.as > h5 > json；设置 ignorels 为 True 后，先尝试读取 h5 中的数据，失败则读取 json 中的数据

8.8.5 计算构型间距

- 参考这个脚本 `8calc_dist.py`：

```python
# coding:utf-8
from dspawpy.diffusion.nebtools import get_distance
from dspawpy.io.structure import read

# Please modify the paths of structure01.as and structure02.as structure files according to the actual situation
# First read the fractional coordinates, element list, and cell information of the two configurations
s1 = read("tests/2.15/01/structure01.as")[0]
s2 = read("tests/2.15/02/structure02.as")[0]
# Calculate the distance between the two configurations, note that this function only accepts fractional coordinates
dist = get_distance(
    spo1=s1.frac_coords,
    spo2=s2.frac_coords,
    lat1=s1.lattice.matrix,
    lat2=s2.lattice.matrix,
)
print("The distance between the two configurations is:", dist, "Angstrom")
```

8.8.6 neb 续算

- 如果需对 neb 进行续算，可参考 `8neb_restart.py`：

```python
# coding:utf-8
import os
from shutil import copytree, rmtree

from dspawpy.diffusion.nebtools import restart

if os.path.isdir("tests/outputs/us/neb4bk"):
    rmtree("tests/outputs/us/neb4bk")

copytree(
    "tests/2.15",
    "tests/outputs/us/neb4bk",
)
restart(
    directory="tests/outputs/us/neb4bk",  # NEB task path
    output="tests/outputs/us/8neb_restart",  # Backup destination
)
```

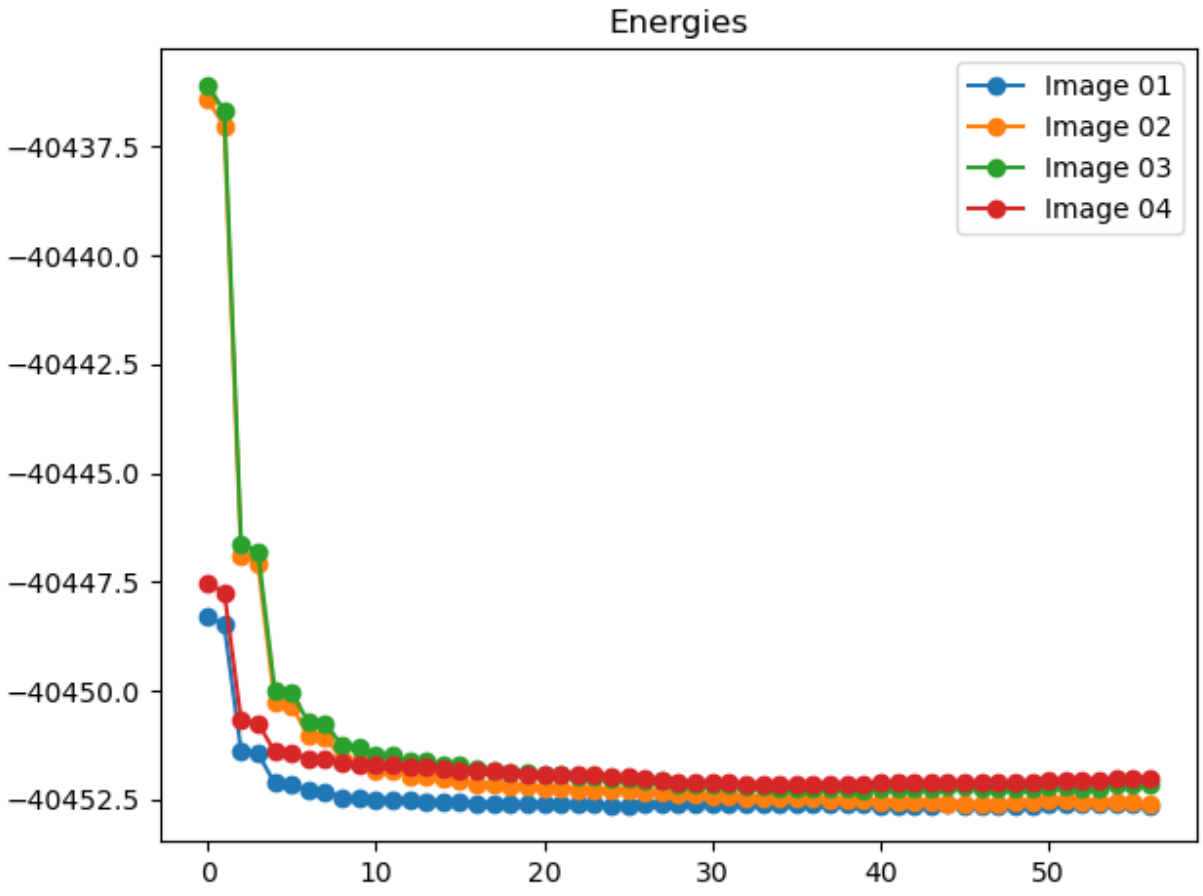
具体效果见*neb 过渡态计算续算说明*。

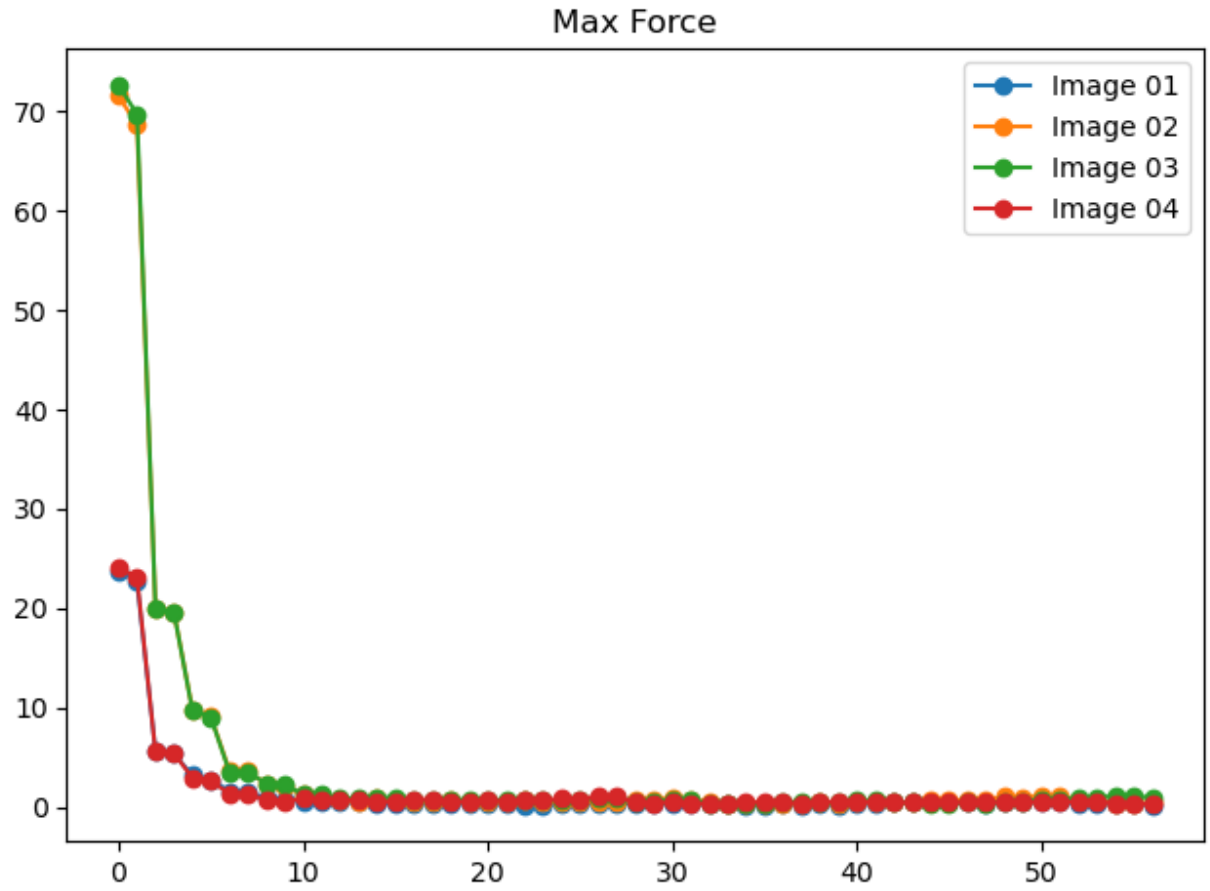
8.8.7 neb 计算过程中能量和最大原子受力的变化趋势图

- 要查看 neb 计算过程中能量和最大原子受力的变化趋势图，可参考 8neb_energy_force_curves.py：

```
# coding:utf-8
from dspawpy.diffusion.nebtools import monitor_force_energy

# Specify the path to the NEB calculation folder; after running, Energies.png and MaxForce.png images will be generated in the specified directory
unfinished_neb_folder = "tests/supplement/neb_unfinished"
monitor_force_energy(
    directory=unfinished_neb_folder,
    outdir="imgs",  # Output image path
)
```

将生成能量和受力变化趋势图：





API: write_neb_structures(), plot_barrier(), summary(), get_distance(), write_movie_json(), write_xyz(), restart()

- `write_neb_structures` 函数负责生成中间构型：

  `dspawpy.diffusion.neb.`**`write_neb_structures`**(*structures: list*, *coords_are_cartesian: bool = True*, *fmt: str = 'as'*, *path: str = '.'*, *prefix='structure'*)

  Interpolate and generate intermediate configuration files

  参数

  - **`structures`** –Structure list
  - **`coords_are_cartesian`** –Is the coordinate Cartesian
  - **`fmt`** –Structure file type, default to "as"
  - **`path`** –Save path
  - **`prefix`** –Filename prefix, default to "structure", which will generate files like structure00.as, structure01.as, ···

  返回

  Saves the configuration file

返回类型
file

示例

First, read the .as file to create a structure object

```
>>> from dspawpy.io.structure import read
>>> init_struct = read("tests/2.15/00/structure00.as")[0]
>>> final_struct = read("tests/2.15/04/structure04.as")[0]
```

Then, interpolate and generate intermediate structure files

```
>>> from dspawpy.diffusion.neb import NEB,write_neb_structures
>>> neb = NEB(init_struct,final_struct,8)
>>> structures = neb.linear_interpolate()    # Linear interpolation
```

Interpolated structures can be saved to the neb folder.

```
>>> write_neb_structures(structures, path="tests/outputs/doctest/11neb_
↪interpolate_structures")
==> .../structure00...as
==> .../structure01...as
==> .../structure02...as
==> .../structure03...as
==> .../structure04...as
==> .../structure05...as
==> .../structure06...as
==> .../structure07...as
```

- `plot_barrier` 函数负责绘制能垒图：

`dspawpy.diffusion.nebtools.`**`plot_barrier`**(*datafile: str = 'neb.h5'*, *directory: str | None = None*, *ri: float | None = None*, *rf: float | None = None*, *ei: float | None = None*, *ef: float | None = None*, *method: str = 'PchipInterpolator'*, *figname: str | None = 'neb_barrier.png'*, *show: bool = True*, *raw: bool = False*, *verbose: bool = False*, ***kwargs*)

Call the scipy.interpolate interpolation algorithm to fit the NEB barrier and plot

参数

- **`datafile`** –Path to neb.h5 or neb.json file
- **`directory`** –NEB calculation path
- **`ri`** –Initial reaction coordinate
- **`rf`** –Final state reaction coordinate
- **`ei`** –Initial state self-consistent energy
- **`ef`** –Final state self-consistent energy
- **`method`** (*`str, optional`*) –Interpolation algorithm, default 'PchipInterpolator'
- **`figname`** (*`str, optional`*) –Barrier image name, default 'neb_barrier.png'
- **`show`** (*`bool, optional`*) –Whether to display the interactive interface, default True
- **`raw`** (*`bool, optional`*) –Whether to return plotting data to CSV

抛出

- **ImportError** –The specified interpolation algorithm does not exist in scipy.interpolate
- **ValueError** –The parameters passed to the interpolation algorithm do not meet the requirements of the algorithm

示例

```
>>> from dspawpy.diffusion.nebtools import plot_barrier
>>> import matplotlib.pyplot as plt
```

Comparing different interpolation algorithms

```
>>> plot_barrier(directory='tests/2.15', method='interp1d', kind=2,␣
↪figname=None, show=False)
>>> plot_barrier(directory='tests/2.15', method='interp1d', kind=3,␣
↪figname=None, show=False)
>>> plot_barrier(directory='tests/2.15', method='CubicSpline', figname=None,␣
↪show=False)
>>> plot_barrier(directory='tests/2.15', method='pchip', figname='tests/
↪outputs/doctest/barrier_comparison.png', show=False)
==> .../barrier_comparison...png
```

Attempt to read neb.h5 file or neb.json file

```
>>> plot_barrier(datafile='tests/2.15/neb.h5', method='pchip', figname='tests/
↪outputs/doctest/barrier_h5.png', show=False)
==> .../barrier_h5...png
>>> plot_barrier(datafile='tests/2.15/neb.json', method='pchip', figname=
↪'tests/outputs/doctest/barrier_json.png', show=False)
==> .../barrier_json...png
```

- summary 函数负责总结 NEB 计算任务的说明文档:

dspawpy.diffusion.nebtools.**summary**(*directory: str = '.'*, *raw=False*, *show_converge=False*, *outdir: str | None = None*, ***kwargs*)

Summary of NEB task completion, execute the following steps in order:

1. Print the forces, reaction coordinates, energy, and energy differences from the initial configuration for each structure
2. Plot the energy barrier diagram
3. Plot and save the convergence processes of energy and forces during the structure optimization

参数

- **directory** –NEB path, default to the current path
- **raw** –Whether to save the plot data to a CSV file
- **show_converge** –Whether to display energy and force convergence plots of the structural optimization process, default is not displayed
- **outdir** –Path to save the convergence process figure, default to directory
- ****kwargs** (*dict*) –Parameters passed to plot_barrier

示例

```
>>> from dspawpy.diffusion.nebtools import summary
>>> directory = 'tests/2.15' # Path for NEB calculation, default to current␣
↪path
>>> summary(directory, show=False, figname='tests/outputs/doctest/neb_barrier.
↪png')
shape: (5, 5)
```

FolderName	Force(eV/Å)	RC(Å)	Energy(eV)	E-E0(eV)
00	0.180272	0.0	-39637.097656	0.0
01	0.014094	0.542789	-39637.019531	0.079814
02	0.026337	1.0868	-39636.878906	0.218265
03	0.024798	1.588367	-39637.0	0.100043
04	0.234429	2.089212	-39637.089844	0.008414

```
==> .../neb_barrier...png
==> .../converge...png
==> .../converge...png
==> .../converge...png
```

```
>>> summary(directory, show=False, figname='tests/outputs/doctest/neb_barrier.
↪png', outdir="tests/outputs/doctest/neb_summary")
shape: (5, 5)
```

FolderName	Force(eV/Å)	RC(Å)	Energy(eV)	E-E0(eV)
00	0.180272	0.0	-39637.097656	0.0
01	0.014094	0.542789	-39637.019531	0.079814
02	0.026337	1.0868	-39636.878906	0.218265
03	0.024798	1.588367	-39637.0	0.100043
04	0.234429	2.089212	-39637.089844	0.008414

```
==> .../neb_barrier...png
==> .../converge...png
==> .../converge...png
==> .../converge...png
```

If inifin=False, the user must place a converged scf.h5 or system.json in the initial and final state subfolders.

- `get_distance` 函数可以计算两个构型间的距离：

`dspawpy.diffusion.nebtools.`**`get_distance`**(*spo1*, *spo2*, *lat1*, *lat2*)

Calculate the distance between two structures based on their fractional coordinates and cell parameters

参数

- **spo1** (*np.ndarray*) –Scores coordinate list 1
- **spo2** (*np.ndarray*) –Fractional coordinate list 2
- **lat1** (*np.ndarray*) –Cell 1
- **lat2** (*np.ndarray*) –Cell 2

返回

Distance

返回类型

float

示例

First, read the structure information

```
>>> from dspawpy.io.structure import read
>>> s1 = read('tests/2.15/01/structure01.as')[0]
>>> s2 = read('tests/2.15/02/structure02.as')[0]
```

Calculate the distance between two configurations

```
>>> from dspawpy.diffusion.nebtools import get_distance
>>> dist = get_distance(s1.frac_coords, s2.frac_coords, s1.lattice.matrix, s2.
↪lattice.matrix)
>>> print('The distance between the two configurations is:', dist, 'Angstrom')
The distance between the two configurations is: 0.476972808803491 Angstrom
```

- `write_movie_json` 和 `write_xyz` 函数可以将中间构型写入 json 或者 xyz 文件：
- `restart` 函数负责重启 NEB 计算：

`dspawpy.diffusion.nebtools.`**`restart`**(*directory: str = '.'*, *output: str = 'bakfile'*)

Archive and compress old NEB tasks, and prepare for continuation at the original path

参数

- **`directory`** –Old NEB task path, default current path
- **`output`** –Backup folder path, default is to create a bakfile folder in the current path for backup; Alternatively, you can specify any path, but it cannot be the same as the current path

示例

```
>>> from dspawpy.diffusion.nebtools import restart
>>> from shutil import copytree
>>> copytree('tests/2.15', 'tests/outputs/doctest/neb4bk2', dirs_exist_
↪ok=True)
'tests/outputs/doctest/neb4bk2'
>>> restart(directory='tests/outputs/doctest/neb4bk2', output='tests/outputs/
↪doctest/neb_backup')
==> .../neb_backup...
```

The preparation for the continuation calculation may take a long time to complete, please be patient

- `monitor_force_energy` 函数负责绘制 NEB 计算过程的能量和受力变化趋势图：

`dspawpy.diffusion.nebtools.`**`monitor_force_energy`**(*directory: str*, *outdir: str = '.'*, *relative: bool = False*)

Read forces and energies during NEB calculations from xx/DS-PAW.log and plot curves

No JSON files are output during the calculation, and only force information is present in nebXX.h5 files, so DS-PAW.log must be read.

Energy matching mode, should hit -40521.972259

8.8.7.1 LOOP 1:

iter | Etot(eV) dE(eV) time # 1 | -35958.655378 -3.595866e+04 47.784 s # 2 | -40069.322436 -4.110667e+03 15.146 s # 3 | -40490.281166 -4.209587e+02 15.114 s # 4 | -40521.972259 -3.169109e+01 17.936 s

示例

```
>>> from dspawpy.diffusion.nebtools import monitor_force_energy
>>> monitor_force_energy(
...     directory="tests/supplement/neb_unfinished",
...     outdir="imgs"
... )
Max Force shape: (57, 4)
```

Folder 01	Folder 02	Folder 03	Folder 04
23.775228	71.547767	72.641234	24.147289
22.683711	68.595607	69.704747	23.0549
5.624252	20.071221	20.049429	5.567894
5.354774	19.631643	19.599093	5.425462
3.188546	9.840143	9.748006	2.943709
…	…	…	…
0.293867	0.812679	0.920251	0.573649
0.27249	0.7475	0.921836	0.540239
0.299767	0.360673	1.174016	0.416171
0.249903	0.288985	1.169237	0.366117
0.204396	0.518356	0.913792	0.300884

```
Energies shape: (57, 4)
```

Folder 01	Folder 02	Folder 03	Folder 04
-40448.281556	-40436.419243	-40436.084611	-40447.527434
-40448.491374	-40437.026948	-40436.685178	-40447.73947
-40451.391617	-40446.884408	-40446.613158	-40450.686918
-40451.448662	-40447.079933	-40446.803281	-40450.743777
-40452.126865	-40450.274376	-40449.978142	-40451.405157
…	…	…	…
-40452.620987	-40452.538682	-40452.230568	-40452.056262
-40452.621777	-40452.544298	-40452.231776	-40452.055815
-40452.620701	-40452.565649	-40452.164604	-40452.035357
-40452.621371	-40452.569113	-40452.164784	-40452.037426
-40452.622418	-40452.577864	-40452.141919	-40452.037885

```
==> ...MaxForce.png...
==> ...Energies.png...
```

8.9 phonon 声子计算数据处理

以 MgO 体系的声子能带态密度计算得到的 *phonon.h5* 为例：

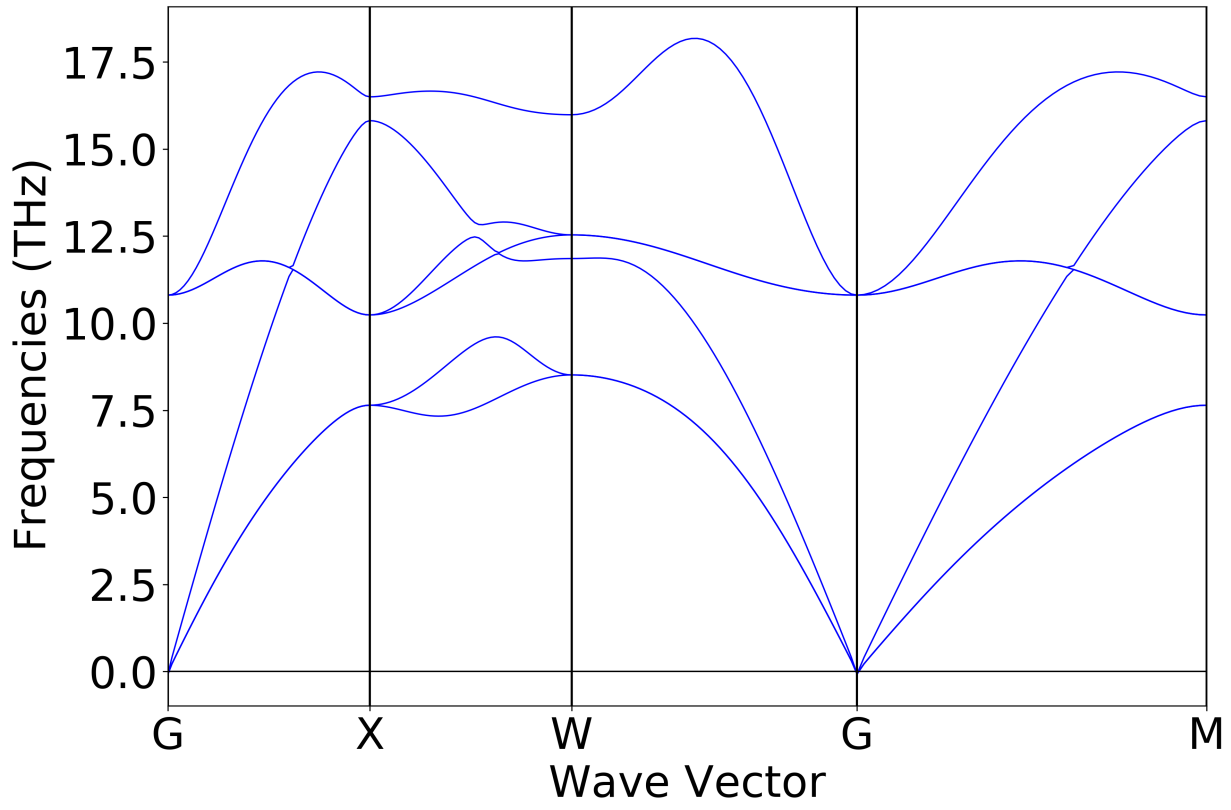
如果未安装 phonopy，运行下列脚本时会弹出 `no module named 'phonopy'` 信息，不影响程序正常运行

8.9.1 声子能带数据处理

- 参考 `9phonon_bandplot.py`：

```python
# coding:utf-8
import os

from pymatgen.phonon.plotter import PhononBSPlotter

from dspawpy.io.read import get_phonon_band_data

band_data = get_phonon_band_data(
    "tests/2.16.1/phonon.h5",
)  # Read phonon band structure
bsp = PhononBSPlotter(band_data)
axes_or_plt = bsp.get_plot(ylim=None, units="thz")  # Y-axis range # Units
import matplotlib.pyplot as plt  # noqa: E402

if isinstance(axes_or_plt, plt.Axes):
    fig = axes_or_plt.get_figure()  # version newer than v2023.8.10
elif isinstance(axes_or_plt, tuple):
    fig = axes_or_plt[0].get_figure()
else:
    fig = axes_or_plt.gcf()  # older version pymatgen

filename = "tests/outputs/us/9phonon_bandplot.png"  # File name for the output phonon band plot
os.makedirs(os.path.dirname(os.path.abspath(filename)), exist_ok=True)
fig.savefig(filename, dpi=300)
```

执行代码可以得到类似以下声子能带曲线：



警告

如果通过 SSH 连接到远程服务器执行上述脚本，出现 QT 相关的报错信息，可能是使用的程序（比如 MobaXterm 等）和 QT 库不兼容，要么更换程序（例如 VSCode 或者系统自带的终端命令行），要么请在 python 脚本第二行开始添加以下代码：

```
import matplotlib
matplotlib.use('agg')
```

8.9.2 声子态密度数据处理

- 参考 9phonon_dosplot.py：

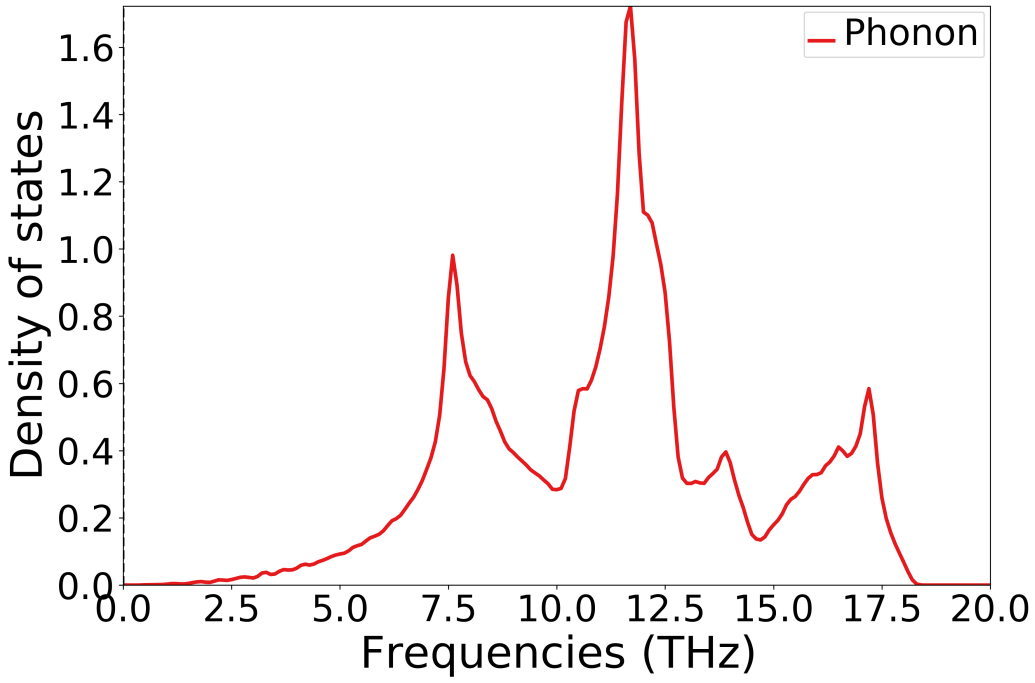
```
# coding:utf-8
import os

from pymatgen.phonon.plotter import PhononDosPlotter

from dspawpy.io.read import get_phonon_dos_data

dos = get_phonon_dos_data("tests/2.16.1/phonon.h5")
dp = PhononDosPlotter(
    stack=False,  # True indicates drawing an area plot
    sigma=None,  # Gaussian blur parameter
```

(续下页)

(接上页)

```
)
dp.add_dos(
    label="Phonon", dos=dos
)  # Legend  # The phonon density of states to be plotted
axes_or_plt = dp.get_plot(
    xlim=[0, 20],  # x-axis range
    ylim=None,  # y-axis range
    units="THz",  # Unit
)
import matplotlib.pyplot as plt  # noqa: E402

if isinstance(axes_or_plt, plt.Axes):
    fig = axes_or_plt.get_figure()  # version newer than v2023.8.10
elif isinstance(axes_or_plt, tuple):
    fig = axes_or_plt[0].get_figure()
else:
    fig = axes_or_plt.gcf()  # older version pymatgen

filename = "tests/outputs/us/9phonon_dosplot.png"  # Energy barrier plot output filename
os.makedirs(os.path.dirname(os.path.abspath(filename)), exist_ok=True)
fig.savefig(filename, dpi=300)
```

执行代码可以得到类似以下声子态密度曲线：



单晶硅声子态密度示意图

警告

如果通过 SSH 连接到远程服务器执行上述脚本，出现 QT 相关的报错信息，可能是使用的程序（比

如 MobaXterm 等）和 QT 库不兼容，要么更换程序（例如 VSCode 或者系统自带的终端命令行），要么请在 python 脚本第二行开始添加以下代码：

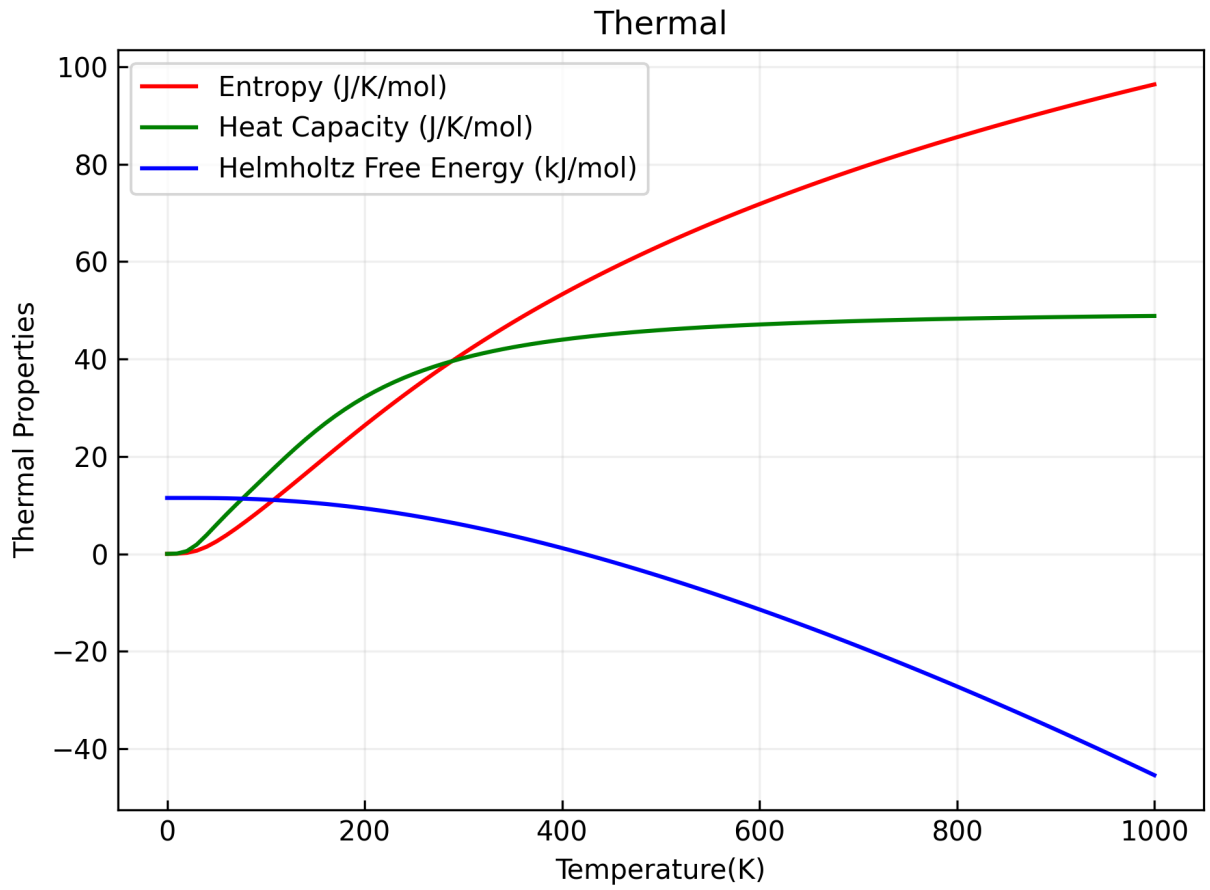
```
import matplotlib
matplotlib.use('agg')
```

8.9.3 声子热力学数据处理

可以参考 9phonon_thermal.py：

```
# coding:utf-8
from dspawpy.plot import plot_phonon_thermal

plot_phonon_thermal(
    datafile="tests/2.26/phonon.h5",  # phonon.h5 data file path
    figname="tests/outputs/us/9phonon.png",  # Output phonon thermodynamics figure filename
    show=False,  # Whether to display the image
)
```

执行代码可以得到类似以下声子热力学曲线：



单晶硅声子热力学性质示意图

API: get_phonon_band_data(), get_phonon_dos_data(), plot_phonon_thermal()

- get_phonon_band_data 函数负责读取声子能带：

dspawpy.io.read.**get_phonon_band_data**(*phonon_band_dir: str*, *verbose: bool = False*)

Reads phonon band data from an h5 or json file and constructs a PhononBandStructureSymmLine object

参数

phonon_band_dir –Path to the band structure file, phonon.h5 / phonon.json, or a folder containing these files

返回类型

PhononBandStructureSymmLine

示例

```
>>> from dspawpy.io.read import get_phonon_band_data
>>> band_data = get_phonon_band_data("tests/2.16/phonon.h5") # Read phonon␣
↪band data
>>> band_data = get_phonon_band_data("tests/2.16/phonon.json") # Read phonon␣
↪band data
```

- get_phonon_dos_data 函数负责读取声子态密度：

dspawpy.io.read.**get_phonon_dos_data**(*phonon_dos_dir: str*, *verbose: bool = False*)

Reads phonon density of states data from an h5 or json file, constructs a PhononDos object

参数

phonon_dos_dir –Path to the phonon DOS file, phonon_dos.h5 / phonon_dos.json, or a folder containing these files

返回类型

PhononDos

示例

```
>>> from dspawpy.io.read import get_phonon_dos_data
>>> phdos = get_phonon_dos_data(phonon_dos_dir='tests/2.16.1/phonon.json')
>>> phdos = get_phonon_dos_data(phonon_dos_dir='tests/2.16.1/phonon.h5')
>>> phdos.frequencies
array([ 0. ,  0.1,  0.2,  0.3,  0.4,  0.5,  0.6,  0.7,  0.8,  0.9,  1. ,
        1.1,  1.2,  1.3,  1.4,  1.5,  1.6,  1.7,  1.8,  1.9,  2. ,  2.1,
        2.2,  2.3,  2.4,  2.5,  2.6,  2.7,  2.8,  2.9,  3. ,  3.1,  3.2,
        3.3,  3.4,  3.5,  3.6,  3.7,  3.8,  3.9,  4. ,  4.1,  4.2,  4.3,
        4.4,  4.5,  4.6,  4.7,  4.8,  4.9,  5. ,  5.1,  5.2,  5.3,  5.4,
        5.5,  5.6,  5.7,  5.8,  5.9,  6. ,  6.1,  6.2,  6.3,  6.4,  6.5,
        6.6,  6.7,  6.8,  6.9,  7. ,  7.1,  7.2,  7.3,  7.4,  7.5,  7.6,
        7.7,  7.8,  7.9,  8. ,  8.1,  8.2,  8.3,  8.4,  8.5,  8.6,  8.7,
        8.8,  8.9,  9. ,  9.1,  9.2,  9.3,  9.4,  9.5,  9.6,  9.7,  9.8,
        9.9, 10. , 10.1, 10.2, 10.3, 10.4, 10.5, 10.6, 10.7, 10.8, 10.9,
       11. , 11.1, 11.2, 11.3, 11.4, 11.5, 11.6, 11.7, 11.8, 11.9, 12. ,
       12.1, 12.2, 12.3, 12.4, 12.5, 12.6, 12.7, 12.8, 12.9, 13. , 13.1,
       13.2, 13.3, 13.4, 13.5, 13.6, 13.7, 13.8, 13.9, 14. , 14.1, 14.2,
       14.3, 14.4, 14.5, 14.6, 14.7, 14.8, 14.9, 15. , 15.1, 15.2, 15.3,
       15.4, 15.5, 15.6, 15.7, 15.8, 15.9, 16. , 16.1, 16.2, 16.3, 16.4,
```

(续下页)

(接上页)

```
       16.5, 16.6, 16.7, 16.8, 16.9, 17. , 17.1, 17.2, 17.3, 17.4, 17.5,
       17.6, 17.7, 17.8, 17.9, 18. , 18.1, 18.2, 18.3, 18.4, 18.5, 18.6,
       18.7, 18.8, 18.9, 19. , 19.1, 19.2, 19.3, 19.4, 19.5, 19.6, 19.7,
       19.8, 19.9, 20. ])
```

- plot_phonon_thermal 函数负责绘制声子热力学性质图：

dspawpy.plot.**plot_phonon_thermal**(*datafile: str = 'phonon.h5'*, *figname: str = 'phonon.png'*, *show: bool = True*, *raw: bool = False*, *verbose: bool = False*)

Task completed for phonon thermodynamic calculations, plot curves of relevant physical quantities versus temperature

phonon.h5/phonon.json -> phonon.png

参数

- **datafile** –Path to an h5 or json file or a folder containing any of these files, default 'phonon.h5'
- **figname** –Filename to save the image
- **show** –Whether to pop up an interactive interface
- **raw** –Whether to save the plotting data to rawphonon.csv file

返回

Image path, default 'phonon.png'

返回类型

figname

示例

```
>>> from dspawpy.plot import plot_phonon_thermal
>>> plot_phonon_thermal('tests/2.26/phonon.h5', figname='tests/outputs/
↪doctest/phonon_thermal_h5.png', show=False)
>>> plot_phonon_thermal('tests/2.26/phonon.json', figname='tests/outputs/
↪doctest/phonon_thermal_json.png', show=False, raw=True)
```

警告

如果通过 SSH 连接到远程服务器执行上述脚本，出现 QT 相关的报错信息，可能是使用的程序（比如 MobaXterm 等）和 QT 库不兼容，要么更换程序（例如 VSCode 或者系统自带的终端命令行），要么请在 python 脚本第二行开始添加以下代码：

```
import matplotlib
matplotlib.use('agg')
```

8.10 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.h5* 为例：

8.10.1 轨迹文件转换格式为.xyz 或.dump

从 aimd 输出的 hdf5 文件中读取数据，并生成轨迹文件

生成的.xyz 或.dump 格式文件，可拖入 OVITO 观察，通过 Device Studio –> Simulator –> OVITO 打开 OVITO 可视化界面，将 xyz 文件或 dump 文件拖入 OVITO 即可。

参考 `10write_aimd_traj.py`：

```
# coding:utf-8
from dspawpy.io.structure import convert

convert(
    infile="tests/2.18/aimd.h5",  # Structure to be converted, if in the current path, only the filename can be written
    si=None,  # Filter the configuration number, if not specified, read all by default
    ele=None,  # Filter element symbol, default reads atomic information for all elements
    ai=None,  # Filter atomic indices (starting from 1), default to read all atomic information
    outfile="tests/outputs/us/10aimdTraj.xyz",  # Can also generate .dump files (lower precision), currently only supports orthogonal unit cells
)
```

执行代码将生成.xyz 和.dump 格式的轨迹文件，该文件可通过 OVITO 打开。关于结构文件转化的更多细节，请参考*structure 结构转化*

知识点：

OVITO 与 dspawpy 都不支持将非正交晶胞的体系保存为 dump 文件

8.10.2 动力学过程中能量、温度等变化曲线

- 参考 `10check_aimd_conv.py`：

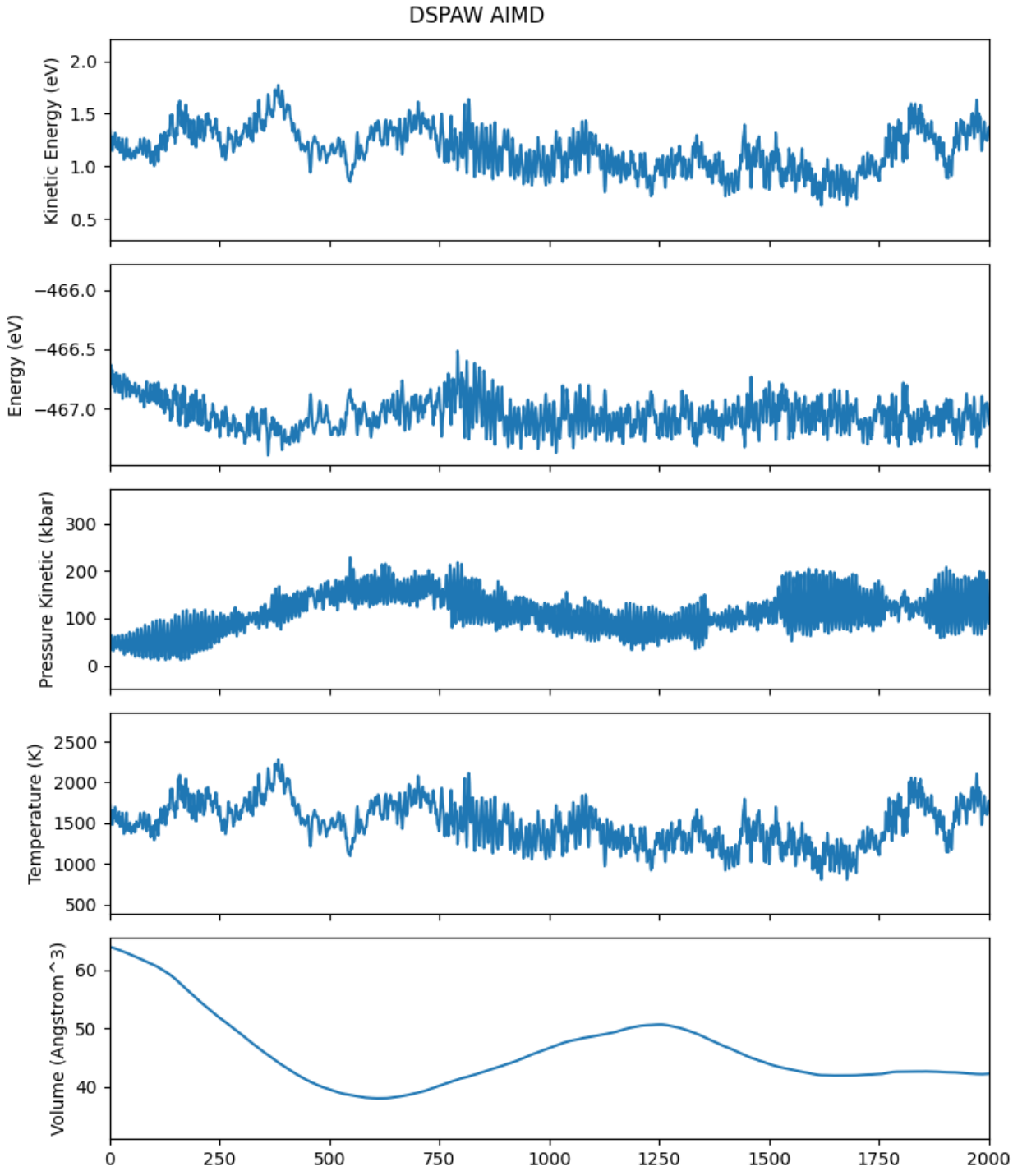
```
# coding:utf-8
from dspawpy.plot import plot_aimd

plot_aimd(
    datafile="tests/2.18/aimd.h5",  # Data file path
    show=False,  # Whether to pop up the image window
    figname="tests/outputs/us/10aimd.png",  # Output image file name
    flags_str="1 2 3 4 5",  # Select data types
)
# The meaning of flags_str is as follows
# 1. Kinetic energy
# 2. Total Energy
# 3. Pressure
```

(续下页)

(接上页)

```
# 4. Temperature
# 5. Volume
```

执行代码将生成如下组合图：



警告

如果通过 SSH 连接到远程服务器执行上述脚本，出现 QT 相关的报错信息，可能是使用的程序（比如 MobaXterm 等）和 QT 库不兼容，要么更换程序（例如 VSCode 或者系统自带的终端命令行），要么请在 python 脚本第二行开始添加以下代码：

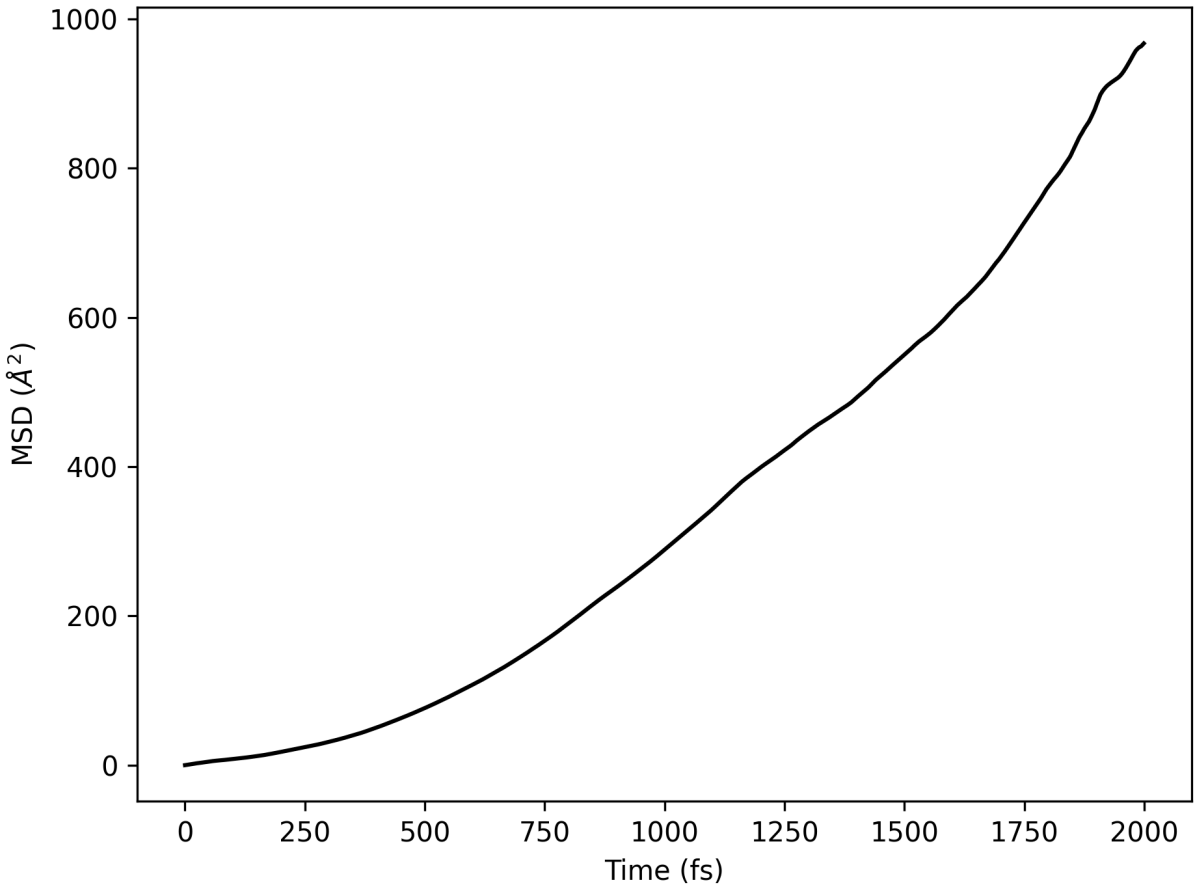
```
import matplotlib
matplotlib.use('agg')
```

8.10.3 均方位移（MSD）分析

- 参考 `10aimd_msd.py`：

```
# coding:utf-8
from dspawpy.analysis.aimdtools import get_lagtime_msd, plot_msd

# If AIMD is not completed in one go, you can assign multiple h5 file paths to the datafile parameter in a list form
lagtime, msd = get_lagtime_msd(
    datafile="tests/2.18/aimd.h5",  # Data file path
    select="all",  # Default selects all atoms
    msd_type="xyz",  # Default to calculate MSD in the xyz directions
    timestep=None,  # Default reads the timestep from the datafile
)
# Plot the graph using the obtained data and save it
plot_msd(
    lagtime,  # X-axis coordinate
    msd,  # vertical axis
    xlim=None,  # Set the display range of the x-axis
    ylim=None,  # Set the display range of the y-axis
    figname="tests/outputs/us/10MSD.png",  # Output image filename
    show=False,  # Whether to pop up the image window
    ax=None,  # Optional subplot specification
)
```

执行代码将生成类似如下图片：



均方位移（MSD）示意图

警告

如果通过 SSH 连接到远程服务器执行上述脚本，出现 QT 相关的报错信息，可能是使用的程序（比如 MobaXterm 等）和 QT 库不兼容，要么更换程序（例如 VSCode 或者系统自带的终端命令行），要么请在 python 脚本第二行开始添加以下代码：

```
import matplotlib
matplotlib.use('agg')
```

8.10.4 均方根偏差（RMSD）分析

- 参考 `10aimd_rmsd.py`：

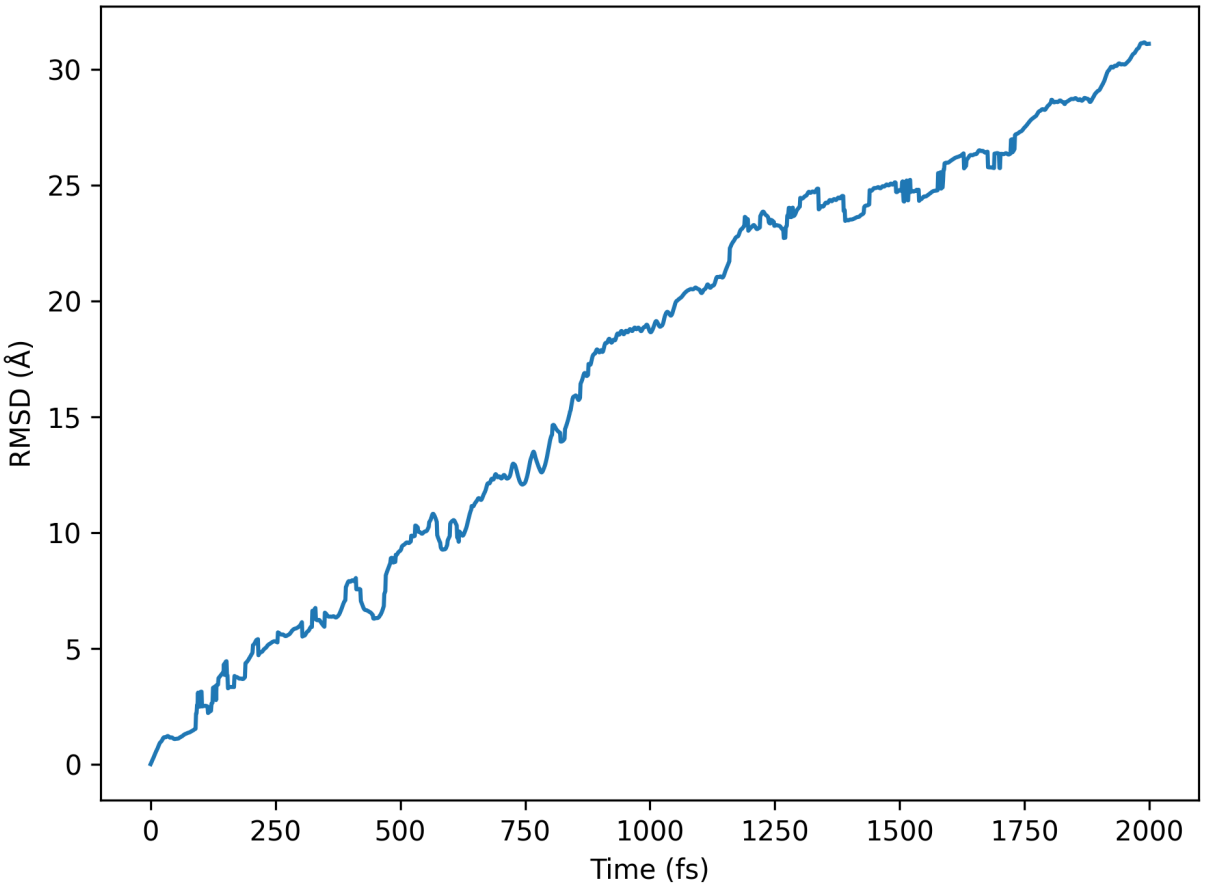
```
# coding:utf-8
from dspawpy.analysis.aimdtools import get_lagtime_rmsd, plot_rmsd

# If AIMD is not completed in a single run, you can assign multiple paths of h5␣
↪files in list form to the datafile parameter
lagtime, rmsd = get_lagtime_rmsd(
```

(续下页)

(接上页)

```
    datafile="tests/2.18/aimd.h5",
    timestep=None,  # Data file path  # Default reads the time step from the␣
↪datafile
)
plot_rmsd(
    lagtime,  # Horizontal coordinate
    rmsd,  # vertical coordinate
    xlim=None,  # Set the display range of the x-axis
    ylim=None,  # Set the display range of the y-axis
    figname="tests/outputs/us/10RMSD.png",  # Output image filename
    show=False,  # Whether to pop up the image window
    ax=None,  # Optional subplot specification
)
```

执行代码将生成类似如下图片：



均方根偏差（RMSD）示意图

警告

如果通过 SSH 连接到远程服务器执行上述脚本，出现 QT 相关的报错信息，可能是使用的程序（比如 MobaXterm 等）和 QT 库不兼容，要么更换程序（例如 VSCode 或者系统自带的终端命令行），要么请

在 python 脚本第二行开始添加以下代码：

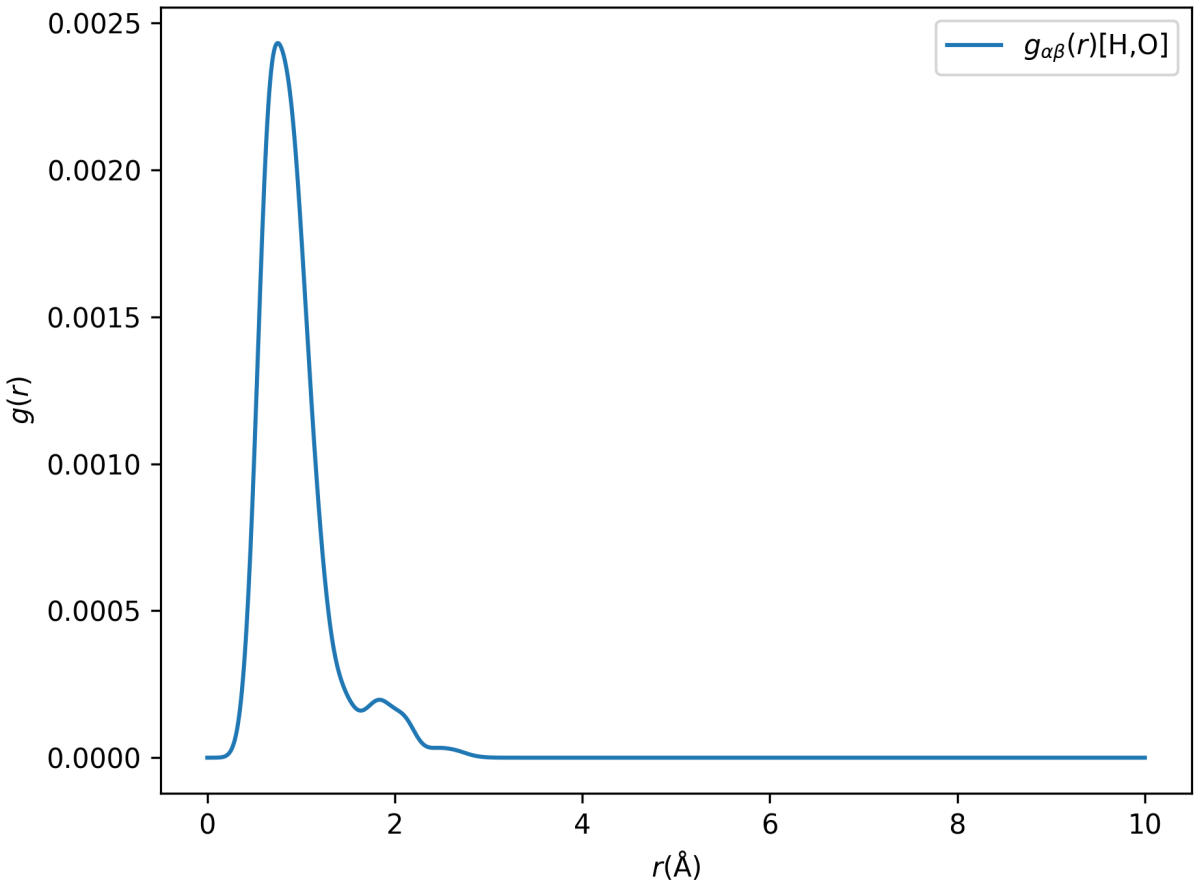
```
import matplotlib
matplotlib.use('agg')
```

8.10.5 原子对分布函数或径向分布函数（RDFs）分析

- 参考 10aimd_rdf.py：

```
# coding:utf-8
from dspawpy.analysis.aimdtools import get_rs_rdfs, plot_rdf

# If AIMD is not completed in one go, you can assign multiple h5 file paths to␣
↪the datafile parameter in the form of a list.
rs, rdfs = get_rs_rdfs(
    datafile="tests/2.18/aimd.h5",  # Data file path
    ele1="H",  # Central element
    ele2="O",  # Target element
    rmin=0.0,  # Minimum radius
    rmax=10.0,  # Maximum radius
    ngrid=1000,  # Number of grid points
    sigma=0.1,  # sigma value
)
plot_rdf(
    rs,  # x-axis values
    rdfs,  # Vertical coordinate
    "H",  # Central element
    "O",  # Object element
    figname="tests/outputs/us/10RDF.png",  # Image save path
    show=False,  # Whether to pop up the image window
    ax=None,  # Subplot can be specified
)
```

执行代码将生成类似如下图片：



径向分布函数（RDFs）示意图

- 这部分涉及的统计学计算较复杂，更多细节请参考函数 API

API: plot_aimd(), get_lagtime_msd(), plot_msd(), get_rs_rdfs(), plot_rdf(), get_lagtime_rmsd(), plot_rmsd()

- `plot_aimd` 函数可用于协助检查 AIMD 计算过程中关键物理量的收敛过程：

`dspawpy.plot.`**`plot_aimd`**(*datafile: str = 'aimd.h5'*, *show: bool = True*, *figname: str = 'aimd.png'*, *flags_str: str = '12345'*, *raw: bool = False*)

Plot the convergence process of key physical quantities after the AIMD task completion

aimd.h5 -> aimd.png

参数

- **`datafile`** –Location of the h5 file. For example, ‘aimd.h5’ or [‘aimd.h5’ , ‘aimd2.h5’]
- **`show`** –Whether to display the interactive interface. Default is False
- **`figname`** –Path to the saved image. Default ‘aimd.h5’
- **`flags_str`** –Subplot number. 1. Kinetic Energy 2. Total Energy 3. Pressure 4. Temperature 5. Volume

- **raw** –Whether to output plot data to a CSV file

返回

Image path, default ‘aimd.png’

返回类型

figname

示例

```
>>> from dspawpy.plot import plot_aimd
```

Read the contents of the aimd.h5 file, plot the convergence process graphs of kinetic energy, total energy, temperature, and volume, and save the corresponding data to rawaimd_*.csv.

```
>>> plot_aimd(datafile='tests/2.18/aimd.h5', flags_str='1 2 3 4 5', raw=True,␣
↪show=False, figname="tests/outputs/doctest/aimdconv.png")
>>> plot_aimd(datafile='tests/2.18/aimd.json', flags_str='1 2 3 4 5',␣
↪show=False, figname="tests/outputs/doctest/aimdconv_json.png")
```

- `get_*` 和 `plot_*` 函数负责读取 AIMD 计算过程中的关键物理量：

dspawpy.analysis.aimdtools.**get_lagtime_msd**(*datafile: str | List[str]*, *select: str | List[int] = 'all'*, *msd_type: str = 'xyz'*, *timestep: float | None = None*)

Calculate the mean squared displacement at different time steps

参数

- **datafile** –
  - Path to *aimd.h5* or *aimd.json* files, or a directory containing these files (prioritizes searching for *aimd.h5*)
  - Written as a list, the data will be read sequentially and merged together
  - For example [‘aimd1.h5’ , ‘aimd2.h5’ , ‘/data/home/my_aimd_task’]
- **select** –Select atomic number or element; atomic numbers start from 0; default is ‘all’ , which calculates all atoms
- **msd_type** –Calculate the type of MSD, options: xyz, xy, xz, yz, x, y, z, default is ‘xyz’ , which calculates all components
- **timestep** –Time interval between adjacent structures, in units of fs, default None, will be read from datafile, if failed, set to 1.0fs; If not None, this value will be used to calculate the time series

返回

- **lagtime** (*np.ndarray*) –Time series
- **result** (*np.ndarray*) –Mean square displacement sequence

示例

```
>>> from dspawpy.analysis.aimdtools import get_lagtime_msd
>>> lagtime, msd = get_lagtime_msd(datafile='tests/2.18/aimd.json',␣
↪timestep=0.1)
Calculating MSD...
>>> lagtime, msd = get_lagtime_msd(datafile='tests/2.18/aimd.h5')
Calculating MSD...
>>> lagtime
array([0.000e+00, 1.000e+00, 2.000e+00, ..., 1.997e+03, 1.998e+03,
       1.999e+03])
>>> msd
array([0.00000000e+00, 3.75844096e-03, 1.45298732e-02, ...,
       7.98518472e+02, 7.99267490e+02, 7.99992702e+02])
>>> lagtime, msd = get_lagtime_msd(datafile='tests/2.18/aimd.h5', select='H')
Calculating MSD...
>>> lagtime, msd = get_lagtime_msd(datafile='tests/2.18/aimd.json', select=[0,
↪1])
Calculating MSD...
>>> lagtime, msd = get_lagtime_msd(datafile='tests/2.18/aimd.h5', select=['H',
↪'O'])
Calculating MSD...
>>> lagtime, msd = get_lagtime_msd(datafile='tests/2.18/aimd.json', select=0)
Calculating MSD...
```

dspawpy.analysis.aimdtools.**get_lagtime_rmsd**(*datafile: str | List[str]*, *timestep: float | None = None*)

参数

- **datafile** –
  - Path to *aimd.h5* or *aimd.json* files, or a directory containing these files (prioritizes searching for *aimd.h5*).
  - Written as a list, the data will be read sequentially and merged together
  - For example [‘aimd1.h5’ , ‘aimd2.h5’ , ‘/data/home/my_aimd_task’]
- **timestep** –Time interval between adjacent structures, in fs, default None, will be read from datafile, set to 1.0fs if failed; If not None, it will be used to calculate the time series

返回

- **lagtime** (*numpy.ndarray*) –Time series
- **rmsd** (*numpy.ndarray*) –Root mean square deviation sequence

示例

```
>>> from dspawpy.analysis.aimdtools import get_lagtime_rmsd
>>> lagtime, rmsd = get_lagtime_rmsd(datafile='tests/2.18/aimd.json')
Calculating RMSD...
>>> lagtime, rmsd = get_lagtime_rmsd(datafile='tests/2.18/aimd.h5',␣
↪timestep=0.1)
Calculating RMSD...
>>> lagtime
array([0.000e+00, 1.000e-01, 2.000e-01, ..., 1.997e+02, 1.998e+02,
```

(续下页)

(接上页)

```
       1.999e+02])
>>> rmsd
array([ 0.        ,  0.05321816,  0.09771622, ..., 28.27847679,
       28.28130893, 28.28414224])
```

dspawpy.analysis.aimdtools.**get_rs_rdfs**(*datafile: str | List[str]*, *ele1: str*, *ele2: str*, *rmin: float = 0*, *rmax: float = 10*, *ngrid: int = 101*, *sigma: float = 0*)

Compute the radial distribution function (RDF).

参数

- **datafile** –
  - Path to *aimd.h5* or *aimd.json* files, or a directory containing these files (prioritizes searching for *aimd.h5*)
  - Written as a list, the data will be read sequentially and merged together
  - For example [‘aimd1.h5’ , ‘aimd2.h5’ , ‘/data/home/my_aimd_task’]
- **ele1** –Central element
- **ele2** –Adjacent elements
- **rmin** –Radial distribution minimum value, default is 0
- **rmax** –Radial distribution maximum value, default is 10
- **ngrid** –Number of grid points in the radial distribution, default is 101
- **sigma** –Smoothing parameter

返回

- **r** (*numpy.ndarray*) –Grid points for the radial distribution
- **rdf** (*numpy.ndarray*) –Radial distribution function

示例

```
>>> from dspawpy.analysis.aimdtools import get_rs_rdfs
>>> rs, rdfs = get_rs_rdfs(datafile='tests/2.18/aimd.h5',ele1='H',ele2='O',␣
↪sigma=1e-6)
Calculating RDF...
>>> rs, rdfs = get_rs_rdfs(datafile='tests/2.18/aimd.h5',ele1='H',ele2='O')
Calculating RDF...
>>> rdfs
array([ 0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        , 14.61375247,
       27.14259351,  1.15450104,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
```

(续下页)

(接上页)

```
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ])
```

dspawpy.analysis.aimdtools.**plot_msd**(*lagtime*, *result*, *xlim: Sequence | None = None*, *ylim: Sequence | None = None*, *figname: str | None = None*, *show: bool = True*, *ax=None*, ***kwargs*)

Compute mean squared displacement (MSD) after the AIMD task is completed

参数

- **lagtime** (*np.ndarray*) –Time series
- **result** (*np.ndarray*) –Mean squared displacement sequence
- **xlim** –x-axis range, default None, set automatically
- **ylim** –y-axis range, default to None, automatically set
- **figname** –Image name, default to None, do not save the image
- **show** –Whether to display the image, default is True
- **ax** –Used to draw the image on a subplot in matplotlib
- ****kwargs** (*dict*) –Other parameters, such as line width, color, etc., are passed to plt.plot function

返回类型

Image after MSD analysis

示例

```
>>> from dspawpy.analysis.aimdtools import get_lagtime_msd, plot_msd
```

Specify the location of the h5 file, use the get_lagtime_msd function to obtain data, and the select parameter selects the nth atom (not by element)

```
>>> lagtime, msd = get_lagtime_msd('tests/2.18/aimd.h5', select=[0])
Calculating MSD...
```

Plot the data and save the figure

```
>>> plot_msd(lagtime, msd, xlim=[0,800], ylim=[0,1000], figname='tests/
↪outputs/doctest/MSD.png', show=False)
==> .../MSD...png
...
```

dspawpy.analysis.aimdtools.**plot_rdf**(*rs*, *rdfs*, *ele1: str*, *ele2: str*, *xlim: Sequence | None = None*, *ylim: Sequence | None = None*, *figname: str | None = None*, *show: bool = True*, *ax=None*, ***kwargs*)

Post-AIMD analysis of rdf and plotting.

参数

- **rs** (*numpy.ndarray*) –Radial distribution grid points
- **rdfs** (*numpy.ndarray*) –Radial distribution function
- **ele1** –Center element
- **ele2** –Adjacent elements
- **xlim** –x-axis range, default to None, i.e., set automatically
- **ylim** –y-axis range, default to None, i.e., automatically set
- **figname** –Image name, default to None, meaning no image is saved
- **show** –Whether to display the image, default to True
- **ax** (*matplotlib.axes.Axes*) –Axis for plotting, default is None, which means creating a new axis
- ****kwargs** (*dict*) –Other parameters, such as line width, color, etc., are passed to plt.plot function

返回类型

Image after RDF analysis

示例

```
>>> from dspawpy.analysis.aimdtools import get_rs_rdfs, plot_rdf
```

First obtain the rs and rdfs data as the x and y axis data

```
>>> rs, rdfs = get_rs_rdfs('tests/2.18/aimd.h5', 'H', 'O', rmax=6)
Calculating RDF...
```

Passing x and y data to the plot_rdf function to plot

```
>>> plot_rdf(rs, rdfs, 'H','O', xlim=[0, 6], ylim=[0, 0.015],figname='tests/
↪outputs/doctest/RDF.png', show=False)
==> .../RDF...png
```

dspawpy.analysis.aimdtools.**plot_rmsd**(*lagtime*, *result*, *xlim: Sequence | None = None*, *ylim: Sequence | None = None*, *figname: str | None = None*, *show: bool = True*, *ax=None*, ***kwargs*)

Post-AIMD analysis of RMSD and plotting

参数

- **lagtime** –Time series
- **result** –Root mean square deviation sequence
- **xlim** –x-axis range
- **ylim** –y-axis range
- **figname** –Image save path
- **show** –Whether to display the image

- **ax** (*matplotlib.axes._subplots.AxesSubplot*) –If plotting subplots, pass the subplot object
- ****kwargs** (*dict*) –Parameters passed to plt.plot

返回类型
Image of RMSD analysis of structures

示例

```
>>> from dspawpy.analysis.aimdtools import get_lagtime_rmsd, plot_rmsd
```

timestep represents the time step length

```
>>> lagtime, rmsd = get_lagtime_rmsd(datafile='tests/2.18/aimd.h5',␣
↪timestep=0.1)
Calculating RMSD...
>>> lagtime, rmsd = get_lagtime_rmsd(datafile='tests/2.18/aimd.json',␣
↪timestep=0.1)
Calculating RMSD...
```

Saving directly as RMSD.png image

```
>>> plot_rmsd(lagtime, rmsd, xlim=[0,200], ylim=[0, 30],figname='tests/
↪outputs/doctest/RMSD.png', show=False)
==> .../RMSD...png
...
```

- 提取数据自行处理请参考：

```
from dspawpy.io.read import get_sinfo
from dspawpy.io.structure import read

aimd_h5_files = ['aimd1.h5','aimd2.h5','aimd3.h5'] # 可以从计算完成的多个 aimd.h5␣
↪中依次抽取数据并合并

# 一次性读取多个aimd.h5文件中的数据并创建pymatgen的Structures列表
pymatgen_structures = read(datafile=aimd_h5_files)

# 或者将数组提取出来
for i, df in enumerate(aimd_h5_files): # 依次获取每个aimd.h5文件的数据
    Nstep, elements, positions, lattices, D_mag_fix = get_sinfo(df)
    # Nstep 表示总离子步数 (int)
    # elements 表示元素列表, (Natom x 1)
    # positions 表示离子坐标, (Nstep x Natom x 3)
    # lattices 表示晶胞矩阵, (Nstep x 3 x 3
    # D_mag_fix 磁矩、自由度相关信息字典
```

警告

如果通过 SSH 连接到远程服务器执行上述脚本，出现 QT 相关的报错信息，可能是使用的程序（比如 MobaXterm 等）和 QT 库不兼容，要么更换程序（例如 VSCode 或者系统自带的终端命令行），要么请在 python 脚本第二行开始添加以下代码：

```
import matplotlib
matplotlib.use('agg')
```

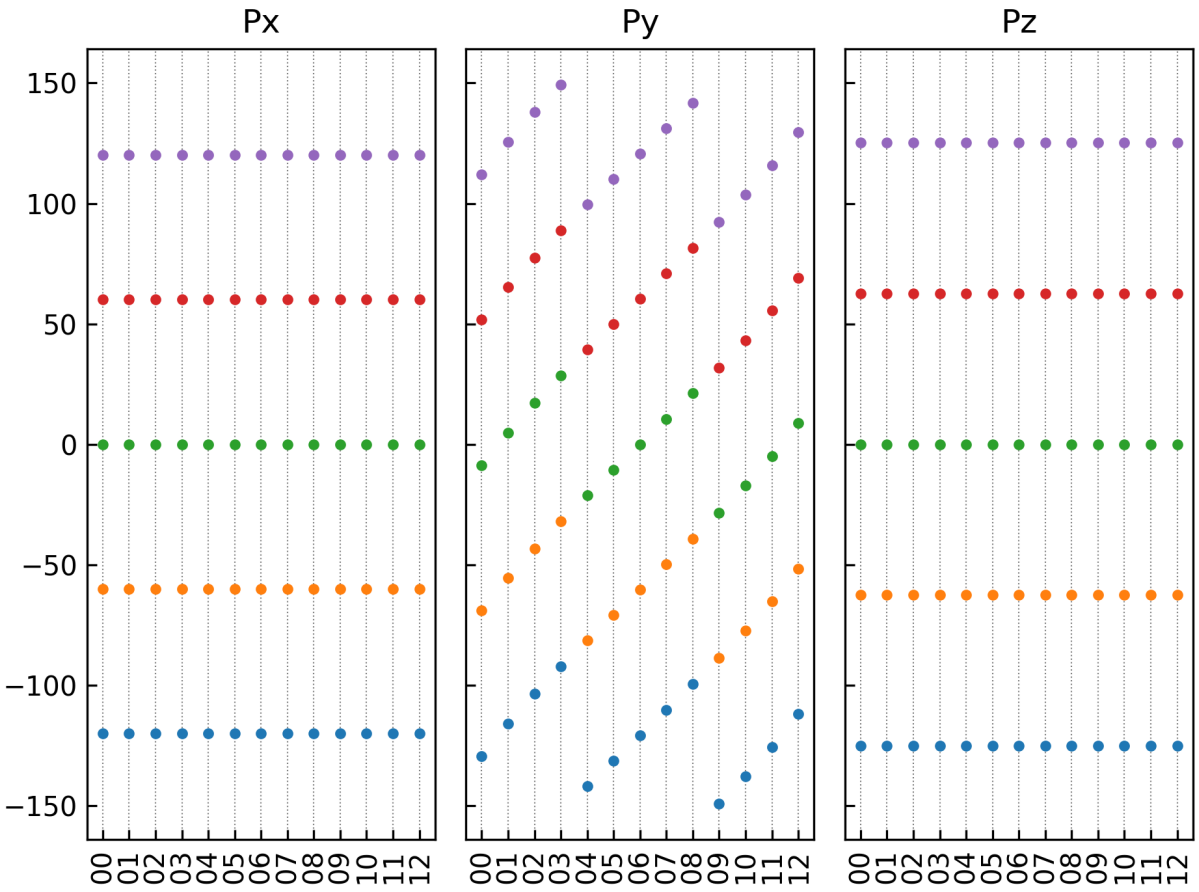
8.11 Polarization 铁电极化数据处理

以快速入门 HfO_2 体系铁电计算得到的系列 *scf.h5* 为例：

- 参考 `11Ferri.py`：

```
# coding:utf-8
from dspawpy.plot import plot_polarization_figure

plot_polarization_figure(
    directory="tests/2.20",  # Path for iron polarization calculation
    repetition=2,  # Number of times to repeat the data points when plotting
    figname="tests/outputs/us/11pol.png",  # Output polarization figure filename
    show=False,  # Whether to display the polarization figure
)  # --> pol.png
```

执行代码将生成如下组合图：



12 组结构对应极化数值

查看首尾构型的铁电极化数值，可以参考如下：

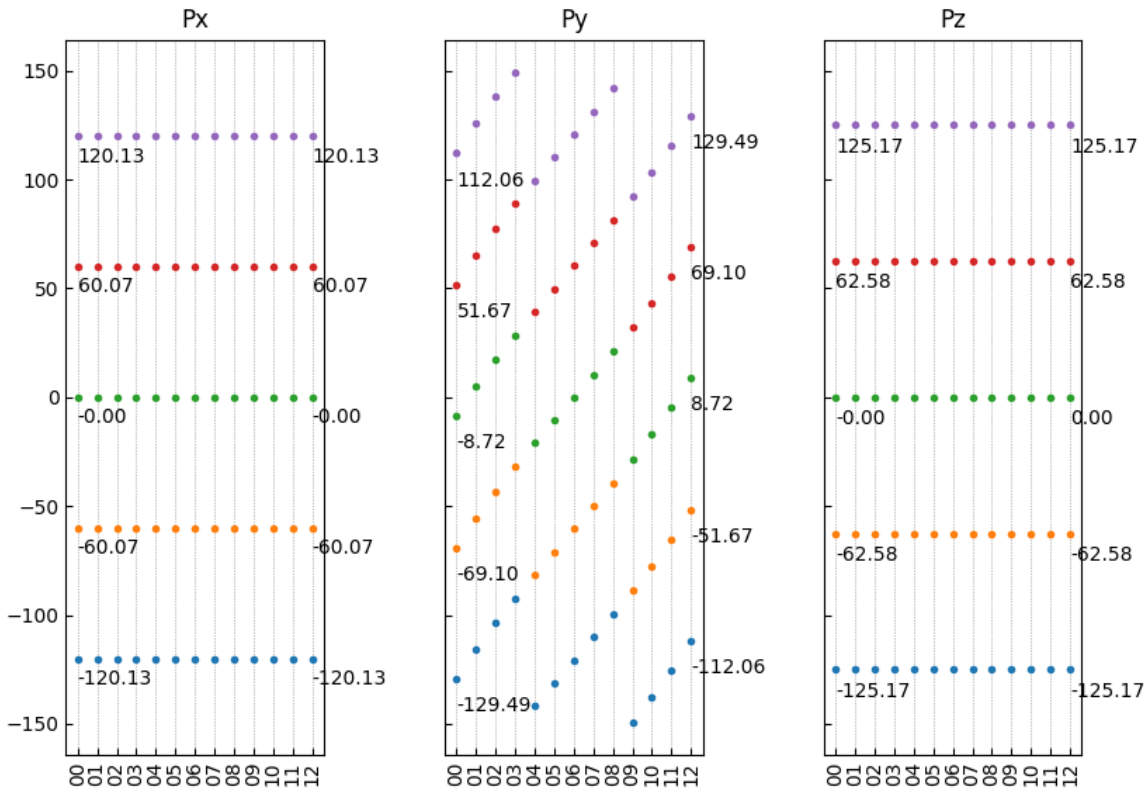
```
from dspawpy.plot import plot_polarization_figure

```

(续下页)

(接上页)

```
plot_polarization_figure(directory='.', annotation=True, annotation_style=1) #␣
↪显示首尾构型的铁电极化数值
```

执行代码将生成如下组合图：

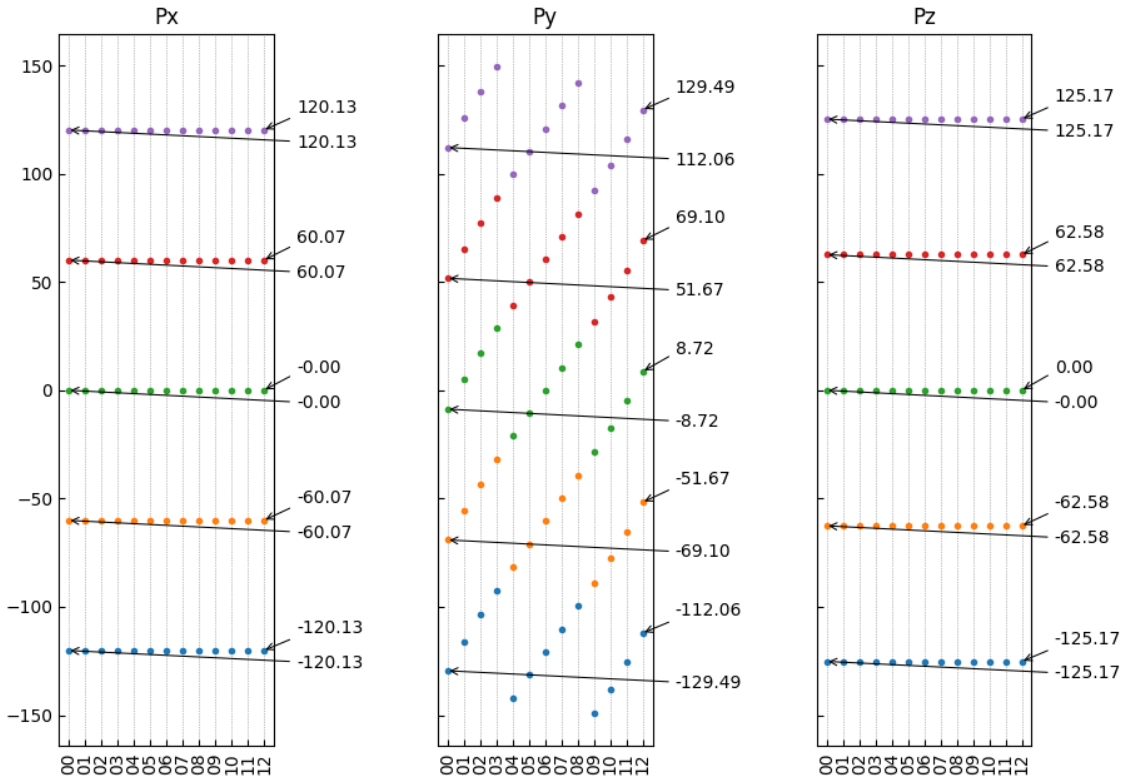


12 组结构对应极化数值（带首尾构型数值）

也可以使用第二种批注样式：

```
from dspawpy.plot import plot_polarization_figure

plot_polarization_figure(directory='.', annotation=True, annotation_style=2) #␣
↪显示首尾构型的铁电极化数值
```

执行代码将生成如下组合图：



12 组结构对应极化数值（带首尾构型数值）

API: plot_polarization_figure()

- `plot_polarization_figure` 函数负责绘制铁电极化图：

`dspawpy.plot.`**`plot_polarization_figure`**(*directory: str*, *repetition: int = 2*, *annotation: bool = False*, *annotation_style: int = 1*, *show: bool = True*, *figname: str = 'pol.png'*, *raw: bool = False*)

Plot the polarization results of the iron electrode

参数

- **`directory`** –Main directory for the iron polarization calculation task
- **`repetition`** –Number of times to repeat drawing along the upper (or lower) direction, default 2
- **`annotation`** –Whether to display the polarization values of the iron electrodes at the beginning and end configurations, displayed by default
- **`show`** –Interactive display of the image, default True
- **`figname`** –Image save path, default ‘pol.png’
- **`raw`** –Whether to save the raw data to a CSV file

返回

axes –Can be passed to other functions for further processing

返回类型
matplotlib.axes._subplots.AxesSubplot

示例

```
>>> from dspawpy.plot import plot_polarization_figure
>>> result = plot_polarization_figure(directory='tests/2.20', figname='tests/
↪outputs/doctest/pol1.png', show=False, annotation=True, annotation_style=1)
>>> result = plot_polarization_figure(directory='tests/2.20', figname='tests/
↪outputs/doctest/pol2.png', show=False, annotation=True, annotation_style=2)
```

警告

如果通过 SSH 连接到远程服务器执行上述脚本，出现 QT 相关的报错信息，可能是使用的程序（比如 MobaXterm 等）和 QT 库不兼容，要么更换程序（例如 VSCode 或者系统自带的终端命令行），要么请在 python 脚本第二行开始添加以下代码：

```
import matplotlib
matplotlib.use('agg')
```

8.12 ZPE 零点振动能数据处理

以快速入门 CO 体系频率计算得到的 *frequency.txt* 文件为例，计算零点振动能，基于以下公式：

$$ZPE = \sum_{i=1}^{3N} \frac{h\nu_i}{2}$$

其中，ν_i 是简正频率，h 是普朗克常数（$6.626 \times 10^{-34} J \cdot s$），$N$ 是原子数。

- 参考 `12getZPE.py`：

```
# coding:utf-8
from dspawpy.io.utils import getZPE

# Import the frequency.txt file obtained from frequency calculation
getZPE(
    fretxt="tests/2.13/frequency.txt",
)
```

执行代码结果文件将保存到 *ZPE.dat* 文件中，文件内容如下：

```
Data read from D:\quickstart\2.13\frequency.txt
Frequency (meV)
284.840038

--> Zero-point energy,  ZPE (eV): 0.142420019
```

API: getZPE()

- `getZPE` 函数负责计算零点振动能：

  Some functions are extracted from [ase](https://wiki.fysik.dtu.dk/ase/index.html).

  `dspawpy.io.utils.getZPE`(*fretxt: str = 'frequency.txt'*)

  Read data from fretxt, calculate ZPE

  The results will also be saved to ZPE_TS.dat.

  参数

  fretxt –Path to the file recording frequency information, default to 'frequency.txt' in the current path

  返回

  Zero-point energy

  返回类型

  ZPE

  示例

```
>>> from dspawpy.io.utils import getZPE
>>> ZPE=getZPE(fretxt='tests/2.13/frequency.txt')
--> Zero-point energy,  ZPE (eV): 0.1424200165
```

8.13 TS 的热校正能

8.13.1 吸附质的熵变对能量的贡献

计算基于以下公式：

$$\Delta S_{ads}\left(0 \rightarrow T, P^{0}\right)=S_{vib}=\sum_{i=1}^{3N}\left[\frac{N_{\mathrm{A}} h \nu_{i}}{T\left(e^{h \nu_{i} / k_{\mathrm{B}} T}-1\right)}-R \ln\left(1-e^{-h \nu_{i} / k_{\mathrm{B}} T}\right)\right]$$

其中，ΔS_{ads} 表示吸附质的熵变，根据简谐近似计算。S_{vib} 表示振动熵。ν_i 是简正频率，N_A 是阿伏伽德罗常数（$6.022 \times 10^{23} mol^{-1}$），$h$ 是普朗克常数（$6.626 \times 10^{-34} J \cdot s$），$k_B$ 是玻尔兹曼常数（$1.38 \times 10^{-23} J \cdot K^{-1}$），$R$ 是理想气体常数（$8.314 J \cdot mol^{-1} \cdot K^{-1}$），$T$ 是体系温度，单位 K。

- 参考 `13getTSads.py`：

```
# coding:utf-8
from dspawpy.io.utils import getTSads

# Import the frequency.txt file calculated from frequency, temperature can be modified arbitrarily
getTSads(
    fretxt="tests/2.13/frequency.txt",
    T=298.15,
)
```

执行代码结果文件将保存到 *TS.dat* 文件中，文件内容如下：

```
Data read from D:\quickstart\2.13\frequency.txt
Frequency (THz)
68.873994

--> Entropy contribution,  T*S (eV): 4.7566990201851275e-06
```

8.13.2 理想气体的熵变对能量的贡献

计算基于如下公式：

$$\begin{aligned} S(T,P) &= S\left(T,P^{\circ}\right) - k_{\mathrm{B}} \ln \frac{P}{P^{\circ}} \\ &= S_{\mathrm{trans}} + S_{\mathrm{rot}} + S_{\mathrm{elec}} + S_{\mathrm{vib}} - k_{\mathrm{B}} \ln \frac{P}{P^{\circ}} \end{aligned}$$

其中：

$$S_{\mathrm{trans}} = k_{\mathrm{B}} \left\{ \ln \left[\left(\frac{2\pi M k_{\mathrm{B}} T}{h^2} \right)^{3/2} \frac{k_{\mathrm{B}} T}{P^{\circ}} \right] + \frac{5}{2} \right\}$$

$$S_{\mathrm{rot}} = \begin{cases} 0 & \text{, monatomic} \\ k_{\mathrm{B}} \left[\ln \left(\frac{8\pi^2 I k_{\mathrm{B}} T}{\sigma h^2} \right) + 1 \right] & \text{, linear} \\ k_{\mathrm{B}} \left\{ \ln \left[\frac{\sqrt{\pi I_A I_B I_C}}{\sigma} \left(\frac{8\pi^2 k_B T}{h^2} \right)^{3/2} \right] + \frac{3}{2} \right\} & \text{, nonlinear} \end{cases}$$

$$S_{\mathrm{elec}} = k_{\mathrm{B}} \ln[2 \times (\text{ total spin }) + 1]$$

$$S_{\mathrm{vib}} = k_{\mathrm{B}} \sum_{i}^{\text{vib DOF}} \left[\frac{\epsilon_i}{k_{\mathrm{B}} T \left(e^{\epsilon_i / k_{\mathrm{B}} T} - 1 \right)} - \ln \left(1 - e^{-\epsilon_i / k_{\mathrm{B}} T} \right) \right]$$

其中：I_A 到 I_C 是非线性分子的三个主惯性矩，I 是线性分子的简并惯性矩，σ 是分子的对称数。另外，monatomic 表示单原子分子，linear 表示线性分子，nonlinear 表示非线性分子。total spin 是总自旋数。vib DOF 表示振动自由度。

- 参考 `13getTSgas.py` 脚本处理：

```python
# coding:utf-8
from dspawpy.io.utils import getTSgas

# Read elements and coordinates from the calculation result file (json or h5)
TSgas = getTSgas(
    fretxt="tests/2.13/frequency.txt",
    datafile="tests/2.13/frequency.h5",
    potentialenergy=-0.0,
    geometry="linear",
    symmetrynumber=1,
    spin=1,
    temperature=298.15,
    pressure=101325.0,
)
print("Entropy contribution, T*S (eV): ", TSgas)

# If only the frequency.txt file is available, the calculation can be completed by manually specifying the elements and coordinates
```

(续下页)

(接上页)

```
# TSgas = getTSgas(fretxt='tests/2.13/frequency.txt', datafile=None,␣
↪potentialenergy=-0.0, elements=['H', 'H'], geometry='linear', positions=[[0.0,␣
↪0.0, 0.0], [0.0, 0.0, 1.0]], symmetrynumber=1, spin=1, temperature=298.15,␣
↪pressure=101325.0)
```

API: getTSads(), getTSgas()

- `getTSads` 函数负责计算吸附质熵变对能量的贡献：

  Some functions are extracted from [ase](https://wiki.fysik.dtu.dk/ase/index.html).

  `dspawpy.io.utils.`**`getTSads`**(*fretxt: str = 'frequency.txt'*, *T: float = 298.15*)

  Read data from fretxt, calculate ZPE and TS

  Will also save the results to TSads.dat

  参数

  - **`fretxt`** –Path to the file recording frequency information, default ‘frequency.txt’ in the current path
  - **`T`** –Temperature, unit K, default 298.15

  返回

  Entropy correction

  返回类型

  TS

  #### 示例

  ```
  >>> from dspawpy.io.utils import getTSads
  >>> TSads=getTSads(fretxt='tests/2.13/frequency.txt', T=298.15)
  --> T*S (eV): 4.7566997225177686e-06
  ```

- `getTSgas` 函数负责计算理想气体熵变对能量的贡献：

  Some functions are extracted from [ase](https://wiki.fysik.dtu.dk/ase/index.html).

  `dspawpy.io.utils.`**`getTSgas`**(*fretxt='frequency.txt'*, *datafile='.'*, *potentialenergy: float = 0.0*, *elements=None*, *geometry='linear'*, *positions=None*, *symmetrynumber=1*, *spin=1*, *temperature=298.15*, *pressure: float = 101325*, *verbose: bool = False*)

  Energy contribution to entropy under the ideal gas approximation”

  参数

  - **`fretxt`** –Path to the file recording frequency information, default is ‘frequency.txt’ in the current path
  - **`datafile`** –Path to the JSON or h5 file or folder containing them, default to the current path; If set to None, the elements and positions parameters must be provided
  - **`potentialenergy`** –Potential energy, unit eV
  - **`elements`** –List of elements, if
  - **`geometry`** –Molecular geometry, monatomic, linear, nonlinear

- **positions** –Atomic coordinates, unit Angstrom
- **symmetrynumber** –Symmetry number
- **spin** –Spin number
- **temperature** –Temperature, unit K
- **pressure** –Pressure, unit Pa

返回
Under the ideal gas approximation, calculates the energy contribution to entropy, in units of eV

返回类型
TSgas

示例

```
>>> from dspawpy.io.utils import getTSgas
>>> TSgas=getTSgas(fretxt='tests/2.13/frequency.txt', datafile='tests/2.13/
↪frequency.h5', potentialenergy=-0.0,  geometry='linear', symmetrynumber=1,␣
↪spin=1, temperature=298.15, pressure=101325.0)
--> T*S (eV): 0.8515317035550232
```

8.14 relaxation 结构优化日志分析

用于解析 DS-PAW 的结构优化日志（默认 *DS-PAW.log*），汇总每步最大受力与能量，并可输出轨迹与最小受力构型。

功能要点：

- 输出 *relax_summary.csv*，包含每步最大受力与能量
- 可生成 *relaxing.xyz* 轨迹文件
- 可生成 *min_force.as* 结构文件

API: read_relax_data()

- `read_relax_data` 函数负责读取结构优化日志并输出统计结果：

  Parse DS-PAW.log from relax task.

  `dspawpy.analysis.post_relax.`**read_relax_data**(*logfile: str = 'DS-PAW.log'*, *print_efs: bool = True*, *relative: bool = False*, *write_traj: bool = True*, *write_as: bool = True*)

  Read relax data from log file, not h5/json, because they lack data.

  Call this function will print info to screen directly.

  参数

  - **logfile** –location of “DS-PAW.log”, such as ‘../another/DS-PAW.log’.
  - **print_efs** –whether to print energies and forces table.
  - **relative** –whether to print energies and forces relative to the first step value.
  - **write_traj** –whether to write traj xyz file

- **write_as** –whether to write structure.as file corresponding to the minimal force.

示例

```
>>> from dspawpy.analysis.post_relax import read_relax_data
>>> read_relax_data(logfile='tests/2.1/DS-PAW.log')
==> ...relaxing...xyz
==> relaxing.xyz
==> min_force.as
==> ...min_force...xyz
shape: (3, 3)
┌──────┬──────────────────────────┬──────────────┐
│ step ┆ force (eV/Angstrom)      ┆ energy (eV)  │
╞══════╪══════════════════════════╪══════════════╡
│ 0    ┆ 0.489284                 ┆ -216.967842  │
│ 1    ┆ 0.1096                   ┆ -216.976886  │
│ 2    ┆ 0.024697                 ┆ -216.977327  │
└──────┴──────────────────────────┴──────────────┘
==> relax_summary.csv
```

```
>>> read_relax_data(logfile='tests/2.1/DS-PAW.log', relative=True, write_
↪traj=False, write_as=False)
shape: (3, 3)
┌──────┬──────────────────────────┬──────────────┐
│ step ┆ force (eV/Angstrom)      ┆ energy (eV)  │
╞══════╪══════════════════════════╪══════════════╡
│ 0    ┆ 0.0                      ┆ 0.0          │
│ 1    ┆ -0.379684                ┆ -0.009044    │
│ 2    ┆ -0.464587                ┆ -0.009485    │
└──────┴──────────────────────────┴──────────────┘
==> relax_summary.csv
```

8.15 hdf5 文件探索

用于查看 hdf5 文件内部数据结构与具体内容，对应 CLI 菜单 15。

功能要点：

- 数据结构：使用 `h5glance name.h5` 浏览层级结构
- 数据查看：可通过 `h5glance name.h5 -` 查看单个键的数据
- 全量数据：可通过 `ptdump -d name.h5` 查看全部键内容

8.16 附录

- 快速下载所有脚本，点击 `用户脚本.zip`
- dspawpy 更新日志

9

常见问题

9.1 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

9.2 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

9.3 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.ksamping= 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；若计算体系存在自定义元素名称，需从赝势库拷贝相应文件到计算目录并重命名

- 错误信息：`E4001`
- 错误详情：*wannier* 计算初始投影轨道与 *wannier* 函数数量不一致
- 解决方案：调整 *structure.as* 文件中初始投影轨道的数量，或者修改 *input.in* 文件中的参数 *wannier.functions*，使得两者数量一致。

- 错误信息：`E4024`
- 错误详情：*wannier* 计算冻结窗口设置错误
- 解决方案：冻结窗口内能带的数量不得多于 *wannier* 函数的数量，需缩小冻结窗口

- 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*

9.4 Version 版本更新常见问题

1. DS-PAW 与 Device Studio 兼容性相关问题：

- DS-PAW 2023A 的生成的能带和声子谱为什么在 DS 中打不开了？

  DS-PAW 2023A 接受用户建议将输出文件中的 Band 改为了 BandEnergies 使数据物理意义更加清晰。同步更新的 Device Studio 2022B-2.0.6 版本已经做了兼容性处理。另外，也可将输出文件中的 BandEnergies 改回为 Band，这样在 Device Studio 2022B-2.0.6 之前的版本中也能打开。

- DS-PAW 2023A 生成的 NEB 数据为什么不能在 DS 中打开？

  DS-PAW 2023A 接受用户建议对输出文件进行了调整。包括统一 neb0N.json/neb0N.h5 和 neb.json/neb.h5 中的标签，调整数据结构让数据物理含义更加清晰等。为了与当前版本的 Device Studio 兼容，我们提供了多个 neb 处理脚本以满足各种情景的需求，比如 neb_visualize.py 脚本可对 neb 优化过程中任意结构进行查看、将 neb 最后构型转成 xyz 轨迹文件，neb_check_results.py 脚本可以打印 NEB 计算各构型的能量和受力表格，绘制能垒图，绘制各 image 的能量与受力收敛图等。详细使用说明，请见辅助工具使用教程 过渡态数据处理部分。Device Studio 的 2023A 版本目前已做兼容性处理，若无法打开请更新后使用。

2. DS-PAW 2023A 中为什么 task=band 不再支持杂化泛函计算？

   由于杂化泛函的特殊性，task=band 和 io.band=true 计算能带对应的实际计算过程完全一致，为了避免用户疑惑两者区别，我们在 task=band（非自洽计算）中不再支持杂化泛函计算。

9.5 手册相关问题

1. EPUB、MOBI 电子书打开后排版错乱，图片前置，导航栏跳转错误

很可能是阅读器的渲染策略问题，windows 端请换用 calibre 或者 sigil 阅读；iOS 端请使用系统自带的图书 app

2. 网页阅读时公式未渲染

很可能是网络问题，请耐心等待渲染完成

10

发布说明

10.1 2025A

10.1.1 赝势更新

四至六周期部分元素（K Ca Sc Ti V Fe Co Ni Cu Zn Ga Ge As Se Br Sr Y Zr Nb Mo Tc Ru Rh Pd Ag Cd In Sn Sb Te Hf Ta W Re Os Ir Pt Au Hg Pb Bi Po）的 LDA 及 PBE 赝势推出 1.1 版本，提高了计算准确性，降低了截断能

10.1.2 功能优化

1. 优化 pcharge 计算时的内存消耗
2. 代码全面优化，提高了计算速度

10.2 2023A 版本更新说明

10.2.1 新增功能

1. SCF 计算时支持恒电势方法
2. 支持隐式溶剂化模型
3. AIMD 计算增加 Langevin 恒温（压）器，增加支持 NPT/NPH 系综；aimd.thermostat=none 更名为 SA（simulated annealing）
4. 支持最大局域化瓦尼尔函数 (Maximally Localized Wannier Function, MLWF) 拟合及插值能带计算

10.2.2 赝势更新

前三周期元素（H, He, Li, Be, B, C, N, O, F, Ne, Na, Mg, Al, Si, P, S, Cl, Ar）的 LDA 及 PBE 赝势推出 1.1 版本，提高了计算准确性，降低了截断能。

10.2.3 IO 调整

1. 新增 HDF5 格式文件作为 DS-PAW 的默认输出文件格式，后续不再维护 json 格式的输出文件
2. DS-PAW.log 参数输出部分修改
3. 去除 tmp 文件夹

10.2.4 功能优化

1. scf.mixType 新增 Pulay 可选值
2. relax.freedom 新增 atom&shape 可选值
3. 新增 relax.pressure 参数
4. 新增 FFT grid 相关参数：cal.FFTGrid, cal.supGrid
5. 在 io.optical=true 时，新增支持参数 cal.opticalGrid
6. band.kpointsNumber 新增支持书写方式
7. 新增 corr.dftuForm 参数，用以决定 DFT+U 方法类型
8. 新增半经验 VDW 修正兼容的交换关联泛函类型

10.2.5 2023A 2024/04/03 更新

10.2.5.1 IO 调整

1. optical 计算时补充输出波函数对 k 的导

10.2.5.2 功能优化

1. 新增 sys.spinDiff 参数，用以限制上下自旋电子数目之差
2. 新增 corr.coreEnergy 参数，用以控制是否计算芯电子能级
3. 修复部分情况 mag 参数读取出错的 bug

10.2.6 2023A 2024/03/15 更新

10.2.6.1 功能优化

1. 修复了单原子计算 H，使用 PWSP 赝势计算异常的问题
2. 优化 HSE 计算部分代码以避免出现 intel error 的问题

10.2.7 2023A 2024/01/12 更新

10.2.7.1 IO 调整

1. **在 rho.bin 中增加费米能级信息**
    a. task=dos/band 可直接读取 rho.bin 中的费米能级无需从 system.json 中读取；（此版本兼容旧版不含 EFermi 的 rho.bin, 但旧版 DS-PAW 不可读取此版本输出的 rho.bin）
    b. 增加参数 band.EfShift，用以控制 task=band 时是否从 rho.bin 中读取 EFermi
2. DS-PAW.log 中 ##PARAMETERS## 部分补充 io.band, io.dos 的输出
3. DS-PAW.log 中增加能带信息输出

10.2.7.2 功能优化

1. 修复了 task=pcharge 时，结果部分输出有误问题；
2. 增加参数 scf.timeStep，用以调整 cal.methods=4/5 时电子步的收敛性
3. **增加参数 task=optical**
    a. cal.opticalGrid 参数名修改为 optical.grid
    b. 增加参数 optical.KKEta, optical.smearing, optical.sigma, optical.Emax

10.2.8 2023A 2023/10/07 更新

10.2.8.1 IO 调整

1. 修复 sys.hybrid = true 时，不在输入文件中定义 sys.functional 会错误输出 warning 的问题
2. 修复 task=aimd/relax 时输出 FinalStep 与 step-XX 数目不匹配的问题

10.2.8.2 功能优化

1. 优化声学计算模块，强制满足声学支求和规则
2. 修复杂化密度泛函 task=relax 受力异常的问题

10.2.9 2023A 2023/6/21 更新

10.2.9.1 IO 调整

1. neb 计算过程中，在 neb0X.h5 中新增 force 相关输出
2. 将 E2095 报错改为 warning，并补充相关说明
3. 调整 task=wannier 相关输出格式
4. 增加 sys.fixedP 相关数据读写及参数传递时有效数字个数

10.2.9.2 功能优化

1. 优化 band.unfolding 相关算法，降低出现内存崩溃问题的几率
2. FixedPotential 迭代时，对输出.input.json 文件时作精度控制，防止部分案例重复计算
3. 修复了部分.txt 输出数据格式问题，防止数据输出堆叠

10.2.10 2023A 2023/5/9 更新

10.2.10.1 IO 调整

1. 调整了 task=neb，sys.hybrid=true，scf.mixType=Broyden 相关输出信息
2. 修复 task=phonon，phonon.thermal=true 时 HeatCapacity 输出 null 的错误

10.2.10.2 功能优化

1. latestStructure.as 补充固定晶格及固定原子位置相关信息；修复 mag 相关信息输出错误的问题
2. 调整 sys.spin = collinear/non-collinear 时 io.magProject 默认值为 true
3. 修复续算时读取 relax.json/relax.h5 中 mag 相关信息错误的问题

10.3 functions 功能概要

10.4 history 历史版本更新说明

10.4.1 2022A

10.4.1.1 新增功能

1. 支持 revPBE/PBEsol/RPBE 交换关联泛函
2. 支持 vdw-optPBE/vdw-optB88/vdw-optB86b/vdw-DF/vdw-DF2/vdw-revDF2 范德瓦尔斯泛函
3. 支持外加电场效应模拟

4. 支持体系晶格可变的 NEB 计算（solid state NEB, ssNEB）
5. 支持使用现代极化理论计算铁电极化值
6. 支持能带去折叠功能
7. 支持利用力常数矩阵计算亥姆霍兹自由能/固定体积热容/熵
8. 支持声子能带计算中考虑长程库仑相互作用
9. 支持使用线性响应的方法计算介电张量
10. 支持使用线性响应的方法计算压电张量
11. 支持使用线性响应的方法计算波恩有效电荷
12. 支持使用 Bader 电荷分析
13. 支持在结构优化时约束指定维度的晶格自由度

10.4.1.2 功能优化

1. 支持.paw（鸿之微 PAW 赝势格式）/.potcar（VASP POTCAR 赝势格式）/.pawpsp（GBRV PAW 赝势格式）
2. 在自洽迭代算法中新增预处理共轭梯度方法
3. 在 NEB 弛豫过程中新增快速惯性弛豫方法
4. 在结构弛豫和 NEB 计算过程中，新增能量收敛判据的收敛方式
5. 支持修改杂化泛函中 Alpha 和 Omega 系数，增加利用 Adaptively Compressed Exchange Operator 加速杂化泛函计算
6. 在输出文件中新增投影磁矩信息，结构弛豫最大受力，过渡态最大受力，能带结构的带隙等输出
7. 在计算目录中新增临时计算文件夹 paw_tmp，用于存放过程文件及报错信息

10.4.2 2021B

10.4.2.1 新增功能

1. 支持 CI-NEB 的方法进行过渡态搜索
2. 支持 PBE0、HSE03、HSE06 杂化泛函
3. 支持 DFT-D2、DFT-D3 范德瓦尔斯修正
4. 支持计算介电常数、折射率、反射率、吸收系数、消光系数等
5. 支持带电体系计算
6. 支持自旋轨道耦合
7. 支持使用有限位移的方法计算声子能带、声子态密度
8. 支持使用 DFPT 方法计算声子能带、声子态密度
9. 支持 DFT+U 处理强关联体系
10. 支持第一性原理分子动力学计算

10.4.3 2021beta

10.4.3.1 新增功能

1. 使用平面波的基组展开波函数
2. 使用投影缀加平面波赝势
3. 结构弛豫计算，支持原子位置弛豫、晶格弛豫、晶格和原子位置弛豫
4. 自洽场计算
5. 支持无自旋、共线自旋、非共线自旋及自旋轨道耦合体系
6. 系统总能量计算
7. 原子受力计算
8. 应力计算
9. 能带（投影能带）计算
10. 电子态密度（投影态密度）计算
11. 电子局域函数计算
12. 势函数计算，支持静电势函数、局域势函数计算