
DS-PAW 手册

2023 鸿之微科技（上海）股份有限公司

2024-12-31

1	软件简介	3
1.1	command 命令说明	3
1.1.1	list 命令列表	3
1.1.2	detail 命令详细描述	3
1.2	run 程序运行	4
1.2.1	submit 命令提交运行	4
1.2.2	script 脚本提交运行	4
2	快速入门	5
2.1	relax 结构弛豫计算	5
2.1.1	<i>Si</i> 原子结构弛豫输入文件	5
2.1.2	run 程序运行	7
2.1.3	analysis 计算结果分析	7
2.2	scf 自洽计算	8
2.2.1	<i>Si</i> 原子自洽计算之准备输入文件	9
2.2.2	run 程序运行	9
2.2.3	analysis 计算结果分析	10
2.3	band 能带计算	10
2.3.1	<i>Si</i> 体系能带计算输入文件	10
2.3.1.1	task = band 两步算	10
2.3.1.2	task = scf 一步算	11
2.3.2	run 程序运行	12
2.3.3	analysis 计算结果分析	12
2.4	pband 投影能带计算	13
2.4.1	<i>Si</i> 投影能带计算输入文件	13
2.4.2	run 程序运行	14
2.4.3	analysis 计算结果分析	14
2.5	dos 态密度计算	15
2.5.1	<i>Si</i> 体系态密度计算输入文件	15
2.5.1.1	task = dos 两步算	15
2.5.1.2	task = scf 一步算	16
2.5.2	run 程序运行	17
2.5.3	analysis 计算结果分析	17
2.6	pdos 投影态密度计算	18
2.6.1	<i>Si</i> 投影态密度计算输入文件	19
2.6.2	run 程序运行	19

2.6.3	analysis 计算结果分析	19
2.7	potential 势函数计算	20
2.7.1	<i>Si</i> 势函数计算输入文件	21
2.7.1.1	task = potential 两步算	21
2.7.1.2	task = scf 一步算	21
2.7.2	run 程序运行	22
2.7.3	analysis 计算结果分析	22
2.8	elf 电子局域密度计算	23
2.8.1	<i>Si</i> 电子局域密度计算输入文件	23
2.8.1.1	task = elf 两步算	23
2.8.1.2	task = scf 一步算	24
2.8.2	run 程序运行	25
2.8.3	analysis 计算结果分析	25
2.9	pcharge 部分电荷密度计算	25
2.9.1	graphene 石墨烯部分电荷密度计算输入文件	26
2.9.2	run 程序运行	27
2.9.3	analysis 计算结果分析	27
2.10	hse 杂化泛函计算	28
2.10.1	<i>Si</i> 杂化泛函计算输入文件	28
2.10.2	run 程序运行	29
2.10.3	analysis 计算结果分析	29
2.10.4	修改杂化泛函 Alpha 系数	30
2.11	vdw 范德瓦尔斯修正计算	30
2.11.1	graphite 石墨结构弛豫输入文件	31
2.11.1.1	半经验修正	31
2.11.1.2	泛函修正	32
2.11.2	run 程序运行	32
2.11.3	analysis 计算结果分析	32
2.12	optical 光学性质计算	33
2.12.1	<i>Si</i> 光学性质计算输入文件	33
2.12.1.1	task = optical 两步算	33
2.12.1.2	task = scf 一步算	34
2.12.2	run 程序运行	34
2.12.3	analysis 计算结果分析	34
2.13	frequency 频率计算	35
2.13.1	CO 频率计算输入文件	35
2.13.2	run 程序运行	36
2.13.3	analysis 计算结果分析	36
2.14	elastic 弹性常数计算	37
2.14.1	<i>Si</i> 弹性常数计算输入文件	37
2.14.2	run 程序运行	38
2.14.3	analysis 计算结果分析	38
2.15	neb 过渡态计算	39
2.15.1	<i>Pt</i> 过渡态计算输入文件	39
2.15.2	run 程序运行	41
2.15.3	analysis 计算结果分析	41
2.16	phonon 声子谱计算	44
2.16.1	<i>MgO</i> 声子谱能带计算输入文件	44
2.16.2	run 程序运行	46
2.16.3	analysis 计算结果分析	46
2.16.4	nac 计算结果分析	47
2.16.5	fdphonon 有限位移法计算声子	48
2.17	soc 自旋轨道耦合计算	49
2.17.1	<i>Bi₂Se₃</i> 自旋轨道耦合计算输入文件	49

2.17.2	run 程序运行	51
2.17.3	analysis 计算结果分析	51
2.18	aimd 分子动力学模拟	53
2.18.1	H_2O 分子动力学模拟输入文件	54
2.18.2	run 程序运行	55
2.18.3	analysis 计算结果分析	55
2.19	efield 外加电场计算	57
2.19.1	硅烯真空方向外加电场计算输入文件	57
2.19.2	run 程序运行	58
2.19.3	analysis 计算结果分析	58
2.20	polarization 铁电计算	60
2.20.1	HfO_2 铁电计算输入文件	61
2.20.2	run 程序运行	63
2.20.3	analysis 计算结果分析	63
2.21	bader 电荷计算	64
2.21.1	$NaCl$ 晶体 Bader 电荷计算输入文件	65
2.21.2	run 程序运行	66
2.21.3	analysis 计算结果分析	66
2.22	bandunfolding 能带反折叠计算	66
2.22.1	Cu_3Au 能带反折叠计算输入文件	66
2.22.2	run 程序运行	68
2.22.3	analysis 计算结果分析	68
2.23	epsilon 介电常数计算	69
2.23.1	Si 介电常数计算输入文件	69
2.23.2	run 程序运行	70
2.23.3	analysis 计算结果分析	70
2.24	piezo 压电张量计算	70
2.24.1	AlN 压电张量计算输入文件	70
2.24.2	run 程序运行	71
2.24.3	analysis 计算结果分析	71
2.25	fixcell 固定基矢弛豫计算	72
2.25.1	MoS_2 固定基矢弛豫计算输入文件	72
2.25.2	run 程序运行	73
2.25.3	analysis 计算结果分析	73
2.26	thermal 声子热力学性质计算	74
2.26.1	Si 声子热力学性质计算输入文件	74
2.26.2	run 程序运行	74
2.26.3	analysis 计算结果分析	75
2.27	solid state NEB 计算	75
2.27.1	$HfZrO$ Solid state NEB 计算输入文件	76
2.27.2	run 程序运行	78
2.27.3	analysis 计算结果分析	78
2.28	solvation 溶剂化能计算	79
2.28.1	H_2O 溶剂化能计算输入文件	79
2.28.2	run 程序运行	80
2.28.3	analysis 计算结果分析	80
2.29	fixedpotential 固定电势计算	81
2.29.1	$Cu - slab$ 固定电势计算输入文件	81
2.29.2	run 程序运行	83
2.29.3	analysis 计算结果分析	83
2.30	wannier 插值能带计算	84
2.30.1	Si 插值能带计算输入文件	84
2.30.2	run 程序运行	85
2.30.3	analysis 计算结果分析	86

2.31	ref 参考文献	87
3	应用案例	89
3.1	<i>O</i> 原子的磁矩计算	89
3.1.1	<i>O</i> 原子自洽计算之文件准备	89
3.1.2	run 程序运行	90
3.1.3	analysis 计算结果分析	90
3.2	<i>NiO</i> 体系的反铁磁计算	90
3.2.1	<i>NiO</i> 体系自洽计算	90
3.2.2	run 程序运行	92
3.2.3	analysis 自洽计算结果分析	92
3.2.4	<i>NiO</i> 体系态密度计算	92
3.2.5	run 程序运行	92
3.2.6	dos 态密度计算结果分析	92
3.2.7	<i>NiO</i> 体系 DFT+U 的态密度计算	93
3.3	<i>AuAl</i> slab 模型功函数计算	94
3.3.1	<i>AuAl</i> slab 模型自洽计算之文件准备	94
3.3.2	run 程序运行	96
3.3.3	workfunction 功函数数据分析	96
3.4	<i>Ru - N4</i> 电催化氮还原反应计算	97
3.4.1	Flow 计算流程及输入文件	97
3.4.1.1	Build 搭建模型	97
3.4.1.2	Relaxation 结构弛豫	97
3.4.1.3	Energy 能量计算	98
3.4.1.4	ReactionEnergy 反应能计算	99
3.4.2	Run 程序运行	100
3.4.3	ReactionEnergy 反应能数据分析	100
3.5	ref 参考文献	101
4	赝势说明	103
4.1	hzw 内部 paw 赝势	103
4.2	VASP 赝势	105
4.3	gbrv 赝势	105
4.4	compare 赝势对比	105
4.4.1	<i>Si</i> 体系能带计算	105
4.4.2	multi 多体系带隙计算	106
5	参数说明	109
5.1	parameter 参数列表	109
5.2	detail 参数详细描述	114
6	输出文件格式说明	145
6.1	relax.h5	145
6.2	scf.h5	147
6.3	rho.h5	151
6.4	rhoBound.h5	151
6.5	band.h5	152
6.6	dos.h5	153
6.7	potential.h5	153
6.8	elf.h5	154
6.9	pcharge.h5	154
6.10	optical.h5	155
6.11	frequency.h5	156
6.12	elastic.h5	156
6.13	neb.h5	157

6.14	neb01.h5	159
6.15	phonon.h5	159
6.16	phonon001.h5	162
6.17	aimd.h5	163
6.18	epsilon.h5	164
6.19	wannier.h5	164
7	续算说明	167
7.1	relax 弛豫计算续算说明	167
7.2	neb 过渡态计算续算说明	167
7.3	aimd 分子动力学模拟续算说明	169
7.4	fixedPotential 恒电势计算续算说明	169
7.5	读取 rho 和 wave 续算说明	170
8	辅助工具使用教程	171
8.1	安装与更新	172
8.1.1	更新 dspawpy	173
8.2	structure 结构转化	174
8.3	volumetricData 数据处理	180
8.3.1	volumetricData 可视化	180
8.3.2	差分 volumetricData 可视化	181
8.3.3	volumetricData 面平均	181
8.4	band 能带数据处理	186
8.4.1	普通能带处理	187
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	188
8.4.3	能带投影到不同元素的不同轨道	190
8.4.4	将能带投影到不同原子的不同轨道	191
8.4.5	能带反折叠处理	193
8.4.6	band-compare 能带对比图处理	195
8.5	dos 态密度数据处理	197
8.5.1	总的态密度	197
8.5.2	将态密度投影到不同的轨道上	199
8.5.3	将态密度投影到不同的元素上	200
8.5.4	将态密度投影到不同原子的不同轨道上	202
8.5.5	将态密度投影到不同原子的分裂 d 轨道 (t2g, eg) 上	203
8.5.6	d-带中心分析	205
8.6	bandDos 能带和态密度共同显示	206
8.6.1	将能带和态密度显示在一张图上	206
8.6.2	将能带和投影态密度显示在一张图上	208
8.7	optical 光学性质数据处理	211
8.8	neb 过渡态计算数据处理	213
8.8.1	输入文件之生成中间构型	213
8.8.2	绘制能垒图	214
8.8.2.1	neb.iniFin = true/false	214
8.8.2.2	neb.iniFin = true	217
8.8.3	过渡态计算数据处理	218
8.8.4	观察 NEB 链	220
8.8.5	计算构型间距	220
8.8.6	neb 续算	221
8.8.7	neb 计算过程中能量和最大原子受力的变化趋势图	221
8.9	phonon 声子计算数据处理	228
8.9.1	声子能带数据处理	228
8.9.2	声子态密度数据处理	230
8.9.3	声子热力学数据处理	231

8.10	aimd 分子动力学模拟数据处理	234
8.10.1	轨迹文件转换格式为.xyz 或.dump	234
8.10.2	动力学过程中能量、温度等变化曲线	235
8.10.3	均方位移 (MSD) 分析	237
8.10.4	均方根偏差 (RMSD) 分析	238
8.10.5	原子对分布函数或径向分布函数 (RDFs) 分析	240
8.11	Polarization 铁电极化数据处理	247
8.12	ZPE 零点振动能数据处理	251
8.13	TS 的热校正能	252
8.13.1	吸附质的熵变对能量的贡献	252
8.13.2	理想气体的熵变对能量的贡献	252
8.14	附录	255
9	常见问题	257
9.1	License 常见错误信息	257
9.2	Inputcheck 检查输入文件常见错误信息	258
9.3	Error 计算过程中常见错误信息	259
9.4	Version 版本更新常见问题	261
9.5	Dspawpy 相关问题	261
9.6	手册相关问题	262
10	发布说明	263
10.1	2023A 版本更新说明	263
10.1.1	新增功能	263
10.1.2	赝势更新	263
10.1.3	IO 调整	263
10.1.4	功能优化	264
10.1.5	2023A 2024/04/03 更新	264
10.1.5.1	IO 调整	264
10.1.5.2	功能优化	264
10.1.6	2023A 2024/03/15 更新	264
10.1.6.1	功能优化	264
10.1.7	2023A 2024/01/12 更新	264
10.1.7.1	IO 调整	264
10.1.7.2	功能优化	264
10.1.8	2023A 2023/10/07 更新	265
10.1.8.1	IO 调整	265
10.1.8.2	功能优化	265
10.1.9	2023A 2023/6/21 更新	265
10.1.9.1	IO 调整	265
10.1.9.2	功能优化	265
10.1.10	2023A 2023/5/9 更新	265
10.1.10.1	IO 调整	265
10.1.10.2	功能优化	265
10.2	functions 功能概要	266
10.3	history 历史版本更新说明	266
10.3.1	2022A	266
10.3.1.1	新增功能	266
10.3.1.2	功能优化	267
10.3.2	2021B	267
10.3.2.1	新增功能	267
10.3.3	2021beta	268
10.3.3.1	新增功能	268



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 膺势头数据信息, 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。

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：

```
1 # task type
2 task = relax
3 #system related
4 sys.structure = structure.as
5 sys.symmetry = true
6 sys.functional = PBE
7 sys.spin = none
8 #scf related
9 cal.methods = 2
10 cal.smearing = 1
11 cal.ksampling = G
```

(续下页)

```

12 cal.kpoints = [10, 10, 10]
13 cal.cutoffFactor = 1.5
14
15 #relax related
16 relax.max = 60
17 relax.freedom = atom
18 relax.convergenceType = force
19 relax.convergence = 0.05
20 relax.methods = CG
21
22 io.wave = false
23 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.ksampling**：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：

```

1 Total number of atoms
2 2
3 Lattice

```

(接上页)

```

4 0.00 2.75 2.75
5 2.75 0.00 2.75
6 2.75 2.75 0.00
7 Direct
8 Si -0.115000000 -0.125000000 -0.125000000
9 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。

```

1 Direct Fix_x Fix_y Fix_z
2 Si -0.115000000 -0.125000000 -0.125000000 F F F
3 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；

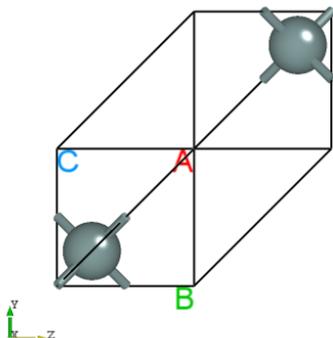
在无图形界面的 linux 环境下运行软件的方式与在 windows 的环境下运行程序会有很大的区别，在 linux 下需要通过命令行的方式来运行程序；一般情况下需要先加载一下环境变量，通常会将需要用的环境变量写入文本或者 *~/.bashrc*，通过 **source** 的命令来加载环境；环境加载完成之后，运行 *DS-PAW relax.in*，此时使用单机版的 DS-PAW 进行计算；如需并行计算则需运行 *DS-PAW -mpi mpirun -mpiargs "-n 2" relax.in*，**-mpi** 指定 mpirun 的名称，**-mpiargs** 指定 mpirun 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：

```

1 Total number of atoms
2 2
3 Lattice
4 0.0000000000000000 2.7500000000000000 2.7500000000000000
5 2.7500000000000000 0.0000000000000000 2.7500000000000000
6 2.7500000000000000 2.7500000000000000 0.0000000000000000
7 Direct
8 Si 0.8801735223171917 0.8748246492235915 0.8748246492235915
9 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。特别需要注意的是：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:

```

1 # task type
2 task = scf
3 #system related
4 sys.structure = structure.as
5 sys.symmetry = true
6 sys.functional = PBE
7 sys.spin = none
8 #scf related
9 cal.methods = 2
10 cal.smearing = 1
11 cal.ksampling = G
12 cal.kpoints = [10, 10, 10]
13 cal.cutoffFactor = 1.5
14 #outputs
15 io.charge = true
16 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:

```

1 Total number of atoms
2 2
3 Lattice
4 0.00 2.75 2.75
5 2.75 0.00 2.75
6 2.75 2.75 0.00
7 Direct
8 Si -0.125000000 -0.125000000 -0.125000000
9 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

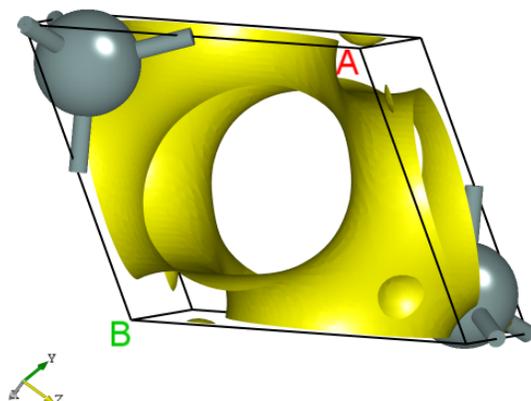
准备好输入文件 *scf.in* 和 *structure.as* 后, 将文件上传到服务器上运行, 按照结构弛豫中介绍的方法执行 DS-PAW *scf.in* 。

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：

```

1 # task type
2 task = band
3 #system related
4 sys.structure = structure.as
5 sys.symmetry = true
6 sys.functional = PBE

```

(续下页)

(接上页)

```

7 sys.spin = none
8
9 cal.iniCharge = ./rho.bin
10 cal.methods = 2
11 cal.smearing = 1
12 cal.cutoffFactor = 1.5
13 cal.totalBands = 12
14
15 #band related
16 band.kpointsLabel= [G,X,W,K,G,L]
17 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]
18 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：

```

1 # task type
2 task = scf
3 #system related
4 sys.structure = structure.as
5 sys.symmetry = true

```

(续下页)

```

6  sys.functional = PBE
7  sys.spin = none
8  #scf related
9  cal.methods = 2
10 cal.totalBands = 12
11 cal.smearing = 1
12 cal.ksampling = G
13 cal.kpoints = [10, 10, 10]
14 cal.cutoffFactor = 1.5
15 #outputs
16 io.charge = true
17 io.wave = true
18 #band related
19 io.band=true
20 band.kpointsLabel= [G,X,W,K,G,L]
21 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]
22 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

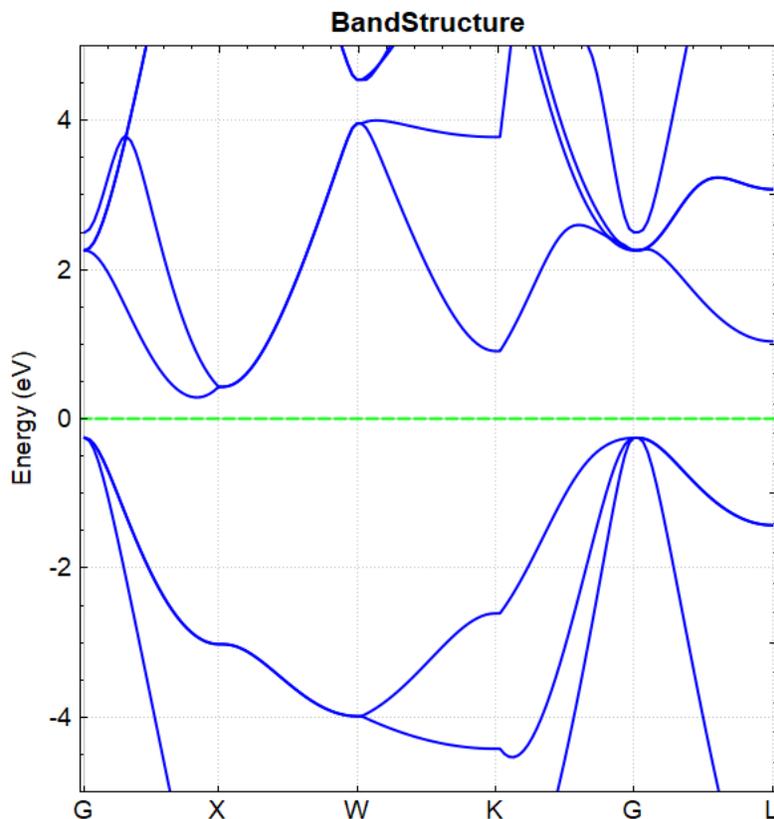
以两步算为例，将参数控制文件 `scf.in`、`band.in` 和结构文件 `structure.as` 上传到服务器，按照结构弛豫中介绍的方法依次执行 `DS-PAW scf.in`、`DS-PAW band.in`。

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:

```
1 # task type
2 task = band
```

(续下页)

```
3 #system related
4 sys.structure = structure.as
5 sys.symmetry = true
6 sys.functional = PBE
7 sys.spin = none
8
9 cal.iniCharge = ./rho.bin
10 cal.methods = 2
11 cal.smearing = 1
12 cal.cutoffFactor = 1.5
13 cal.totalBands = 12
14
15 #band related
16 band.kpointsLabel= [G,X,W,K,G,L]
17 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]
18 band.kpointsNumber= [30, 30, 30, 30, 30]
19 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

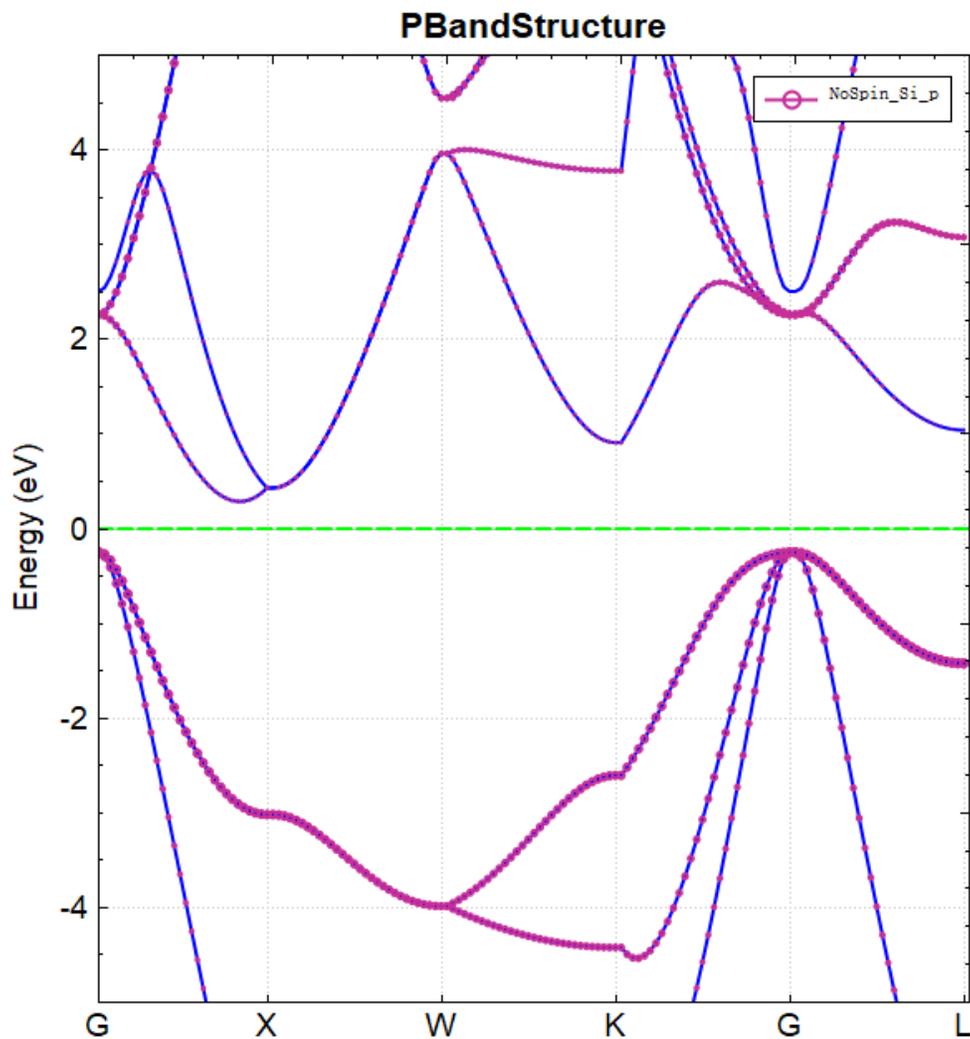
准备好输入文件 *pw_band.in* 和 *structure.as* 以及 *rho.bin* 之后, 将文件上传到服务器上运行, 按照结构弛豫中介绍的方法执行 *DS-PAW pw_band.in*。

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：

```
1 # task type
2 task = dos
```

(续下页)

(接上页)

```

3 #system related
4 sys.structure = structure.as
5 sys.symmetry = true
6 sys.functional = PBE
7 sys.spin = none
8
9 cal.iniCharge = ./rho.bin
10 cal.methods = 2
11 cal.smearing = 4
12 cal.ksampling = G
13 cal.kpoints = [20, 20, 20]
14 cal.cutoffFactor = 1.5
15
16 #dos related
17 dos.range=[-10, 10]
18 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:

```

1 # task type
2 task = scf
3 #system related
4 sys.structure = structure.as
5 sys.symmetry = true
6 sys.functional = PBE
7 sys.spin = none
8 #scf related
9 cal.methods = 2
10 cal.smearing = 4
11 cal.ksampling = G

```

(续下页)

(接上页)

```
12 cal.kpoints = [10, 10, 10]
13 cal.cutoffFactor = 1.5
14 #outputs
15 io.charge = true
16 io.wave = true
17 #dos related
18 io.dos=true
19 dos.range=[-10, 10]
20 dos.resolution=0.05
```

i 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

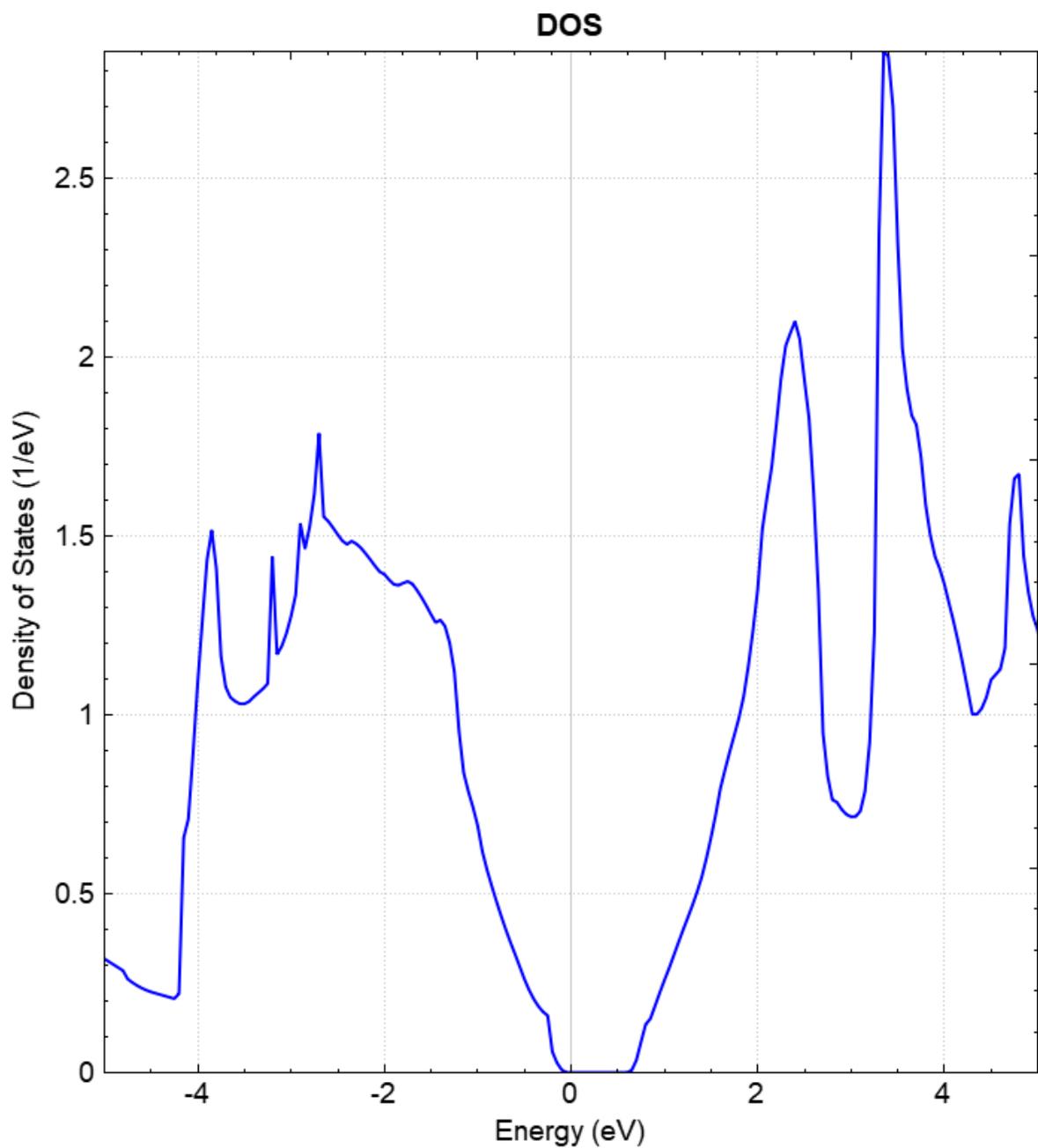
以两步算为例，将参数控制文件 `scf.in`、`dos.in` 和结构文件 `structure.as` 上传到服务器，按照[结构弛豫中介绍的方法](#)依次执行 `DS-PAW scf.in`、`DS-PAW dos.in`。

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：



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 S_i 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:

```

1 # task type
2 task = dos
3 #system related
4 sys.structure = structure.as
5 sys.symmetry = true
6 sys.functional = PBE
7 sys.spin = none
8
9 cal.iniCharge = ./rho.bin
10 cal.methods = 2
11 cal.smearing = 4
12 cal.ksampling = G
13 cal.kpoints = [20, 20, 20]
14 cal.cutoffFactor = 1.5
15
16 #dos related
17 dos.range=[-10, 10]
18 dos.resolution=0.05
19 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

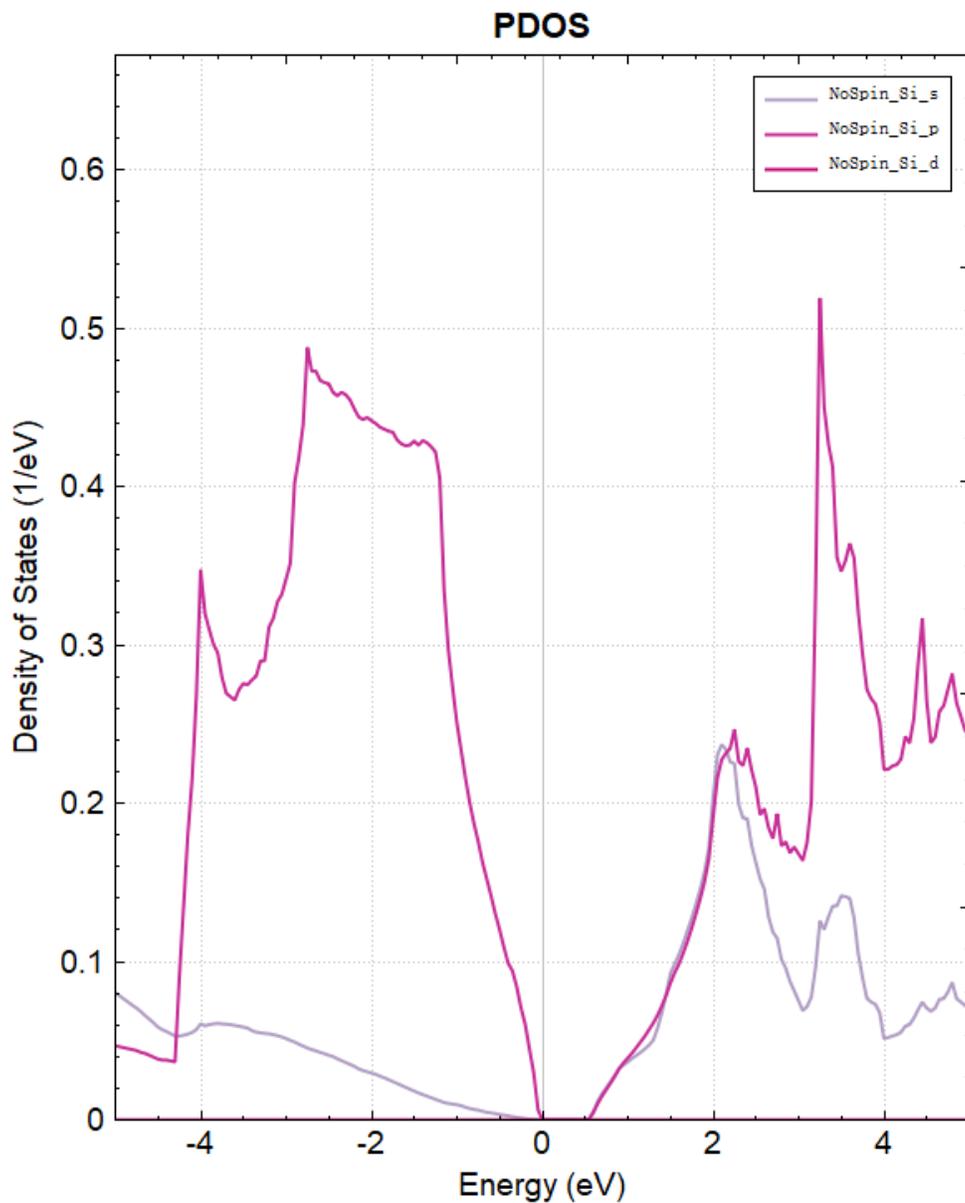
准备好输入文件 *pdos.in* 和 *structure.as* 以及 *rho.bin* 之后, 将文件上传到服务器上运行, 按照结构弛豫中介绍的方法执行 *DS-PAW pdos.in* 。

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:



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 S_i 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：

```

1 # task type
2 task = potential
3 #system related
4 sys.structure = structure.as
5 sys.symmetry = true
6 sys.functional = PBE
7 sys.spin = none
8
9 cal.iniCharge = ./rho.bin
10 cal.methods = 2
11 cal.smearing = 1
12 cal.ksampling = G
13 cal.kpoints = [10, 10, 10]
14 cal.cutoffFactor = 1.5
15
16 #potential related
17 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：

```

1 # task type
2 task = scf
3 #system related

```

(续下页)

```
4 sys.structure = structure.as
5 sys.symmetry = true
6 sys.functional = PBE
7 sys.spin = none
8 #scf related
9 cal.methods = 2
10 cal.smearing = 1
11 cal.ksampling = G
12 cal.kpoints = [10, 10, 10]
13 cal.cutoffFactor = 1.5
14 #outputs
15 io.charge = true
16 io.wave = true
17 #potential related
18 io.potential = true
```

i 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

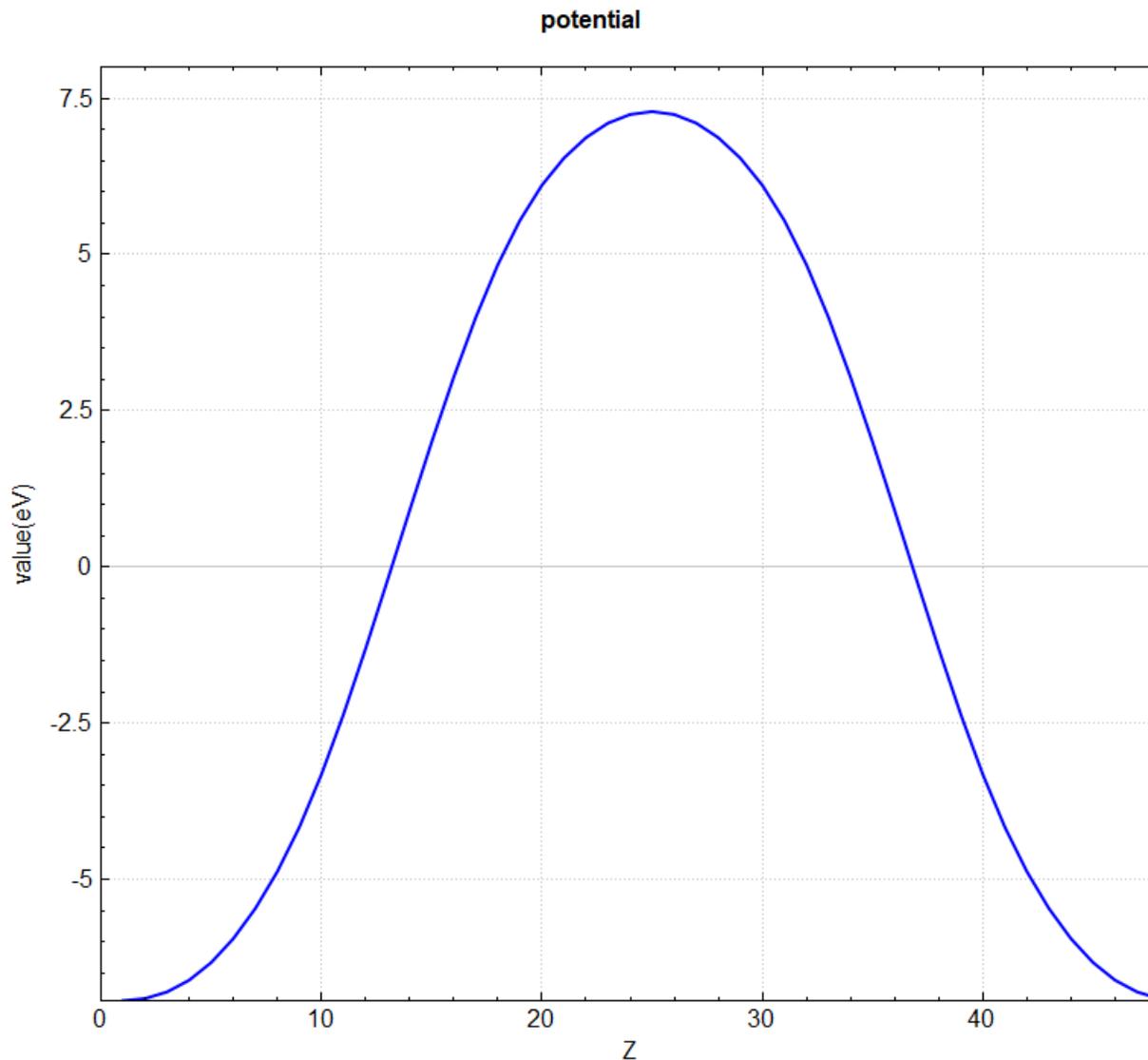
以两步算为例，将参数控制文件 `scf.in`、`potential.in` 和结构文件 `structure.as` 上传到服务器，按照结构弛豫中介绍的方法依次执行 `DS-PAW scf.in`、`DS-PAW potential.in`。

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：

```

1 # task type
2 task = elf
3 #system related
4 sys.structure = structure.as
5 sys.symmetry = true
6 sys.functional = PBE
7 sys.spin = none
8
9 cal.iniCharge = ./rho.bin
10 cal.methods = 2
11
12 cal.smearing = 1
13 cal.ksampling = G
14 cal.kpoints = [10, 10, 10]
15 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：

```

1 # task type
2 task = scf
3 #system related
4 sys.structure = structure.as
5 sys.symmetry = true
6 sys.functional = PBE
7 sys.spin = none
8 #scf related
9 cal.methods = 2
10 cal.smearing = 1
11 cal.ksampling = G
12 cal.kpoints = [10, 10, 10]
13 cal.cutoffFactor = 1.5
14 #outputs
15 io.charge = true
16 io.wave = true
17 #elf related
18 io.elf = true

```

i 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

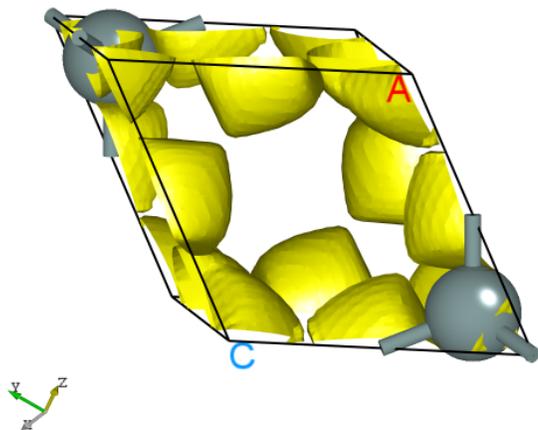
以两步算为例，将参数控制文件 `scf.in`、`ELF.in` 和结构文件 `structure.as` 上传到服务器，按照[结构弛豫中介绍的方法](#)依次执行 `DS-PAW scf.in`、`DS-PAW ELF.in`。

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：

```

1 # task type
2 task = pcharge
3 #system related
4 sys.structure = structure.as
5 sys.symmetry = true
6 sys.functional = PBE
7 sys.spin = none
8
9 cal.methods = 2
10 cal.smearing = 1
11 cal.ksampling = G
12 cal.kpoints = [9, 9, 1]
13 cal.cutoffFactor = 1.5
14 cal.iniCharge = ./rho.bin
15 cal.iniWave = ./wave.bin
16
17 #pcharge related
18 pcharge.bandIndex = [4,5]
19 pcharge.kpointsIndex = [12]
20 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：

```

1 Total number of atoms
2 2
3 Lattice
4 2.46120000 0.00000000 0.00000000
5 -1.23060000 2.13146172 0.00000000
6 0.00000000 0.00000000 6.70900000
7 Cartesian
8 C 0.61530000 0.35524362 3.35450000
9 C 0.61530000 1.77621810 3.35450000

```

i 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

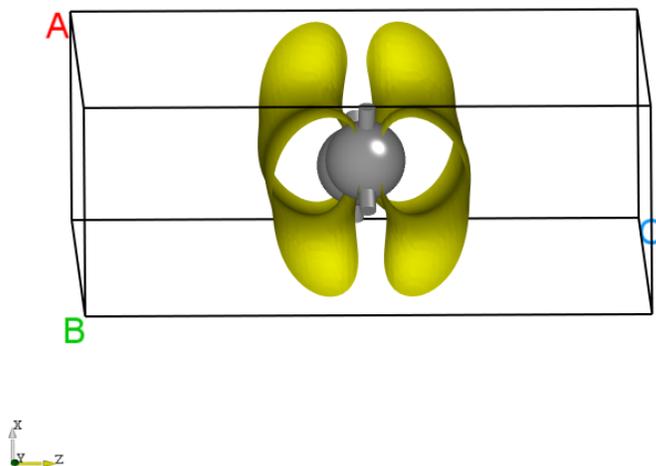
准备好输入文件 `pcharge.in`、`structure.as` 以及自洽计算得到的 `rho.bin`、`wave.bin` 文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 `DS-PAW pcharge.in`。

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：

```

1 # task type
2 task = scf
3 #system related
4 sys.structure = structure.as
5 sys.symmetry = true
6 sys.spin = none
7 #scf related
8 cal.methods = 1
9 cal.totalBands = 12
10 cal.smearing = 1
11 cal.ksampling = G
12 cal.kpoints = [5, 5, 5]
13 cal.cutoffFactor = 1.5
14 #band related
15 io.band = true
16 band.kpointsCoord=[0.62500000,0.25000000,0.62500000,0.50000000,0.00000000,0.50000000,0.00000000,
↳0.00000000,0.00000000,0.50000000,0.00000000,0.50000000,0.50000000,0.25000000,0.75000000,0.
↳37500000,0.37500000,0.75000000,0.00000000,0.00000000,0.00000000]
17 band.kpointsLabel = [U,X,G,X,W,K,G]
18 band.kpointsNumber = [20,20,20,20,20,20]
19 band.project = false
20 #HSE related
21 sys.hybrid=true
22 sys.hybridType=HSE06
23 #outputs
24 io.charge = true
25 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*

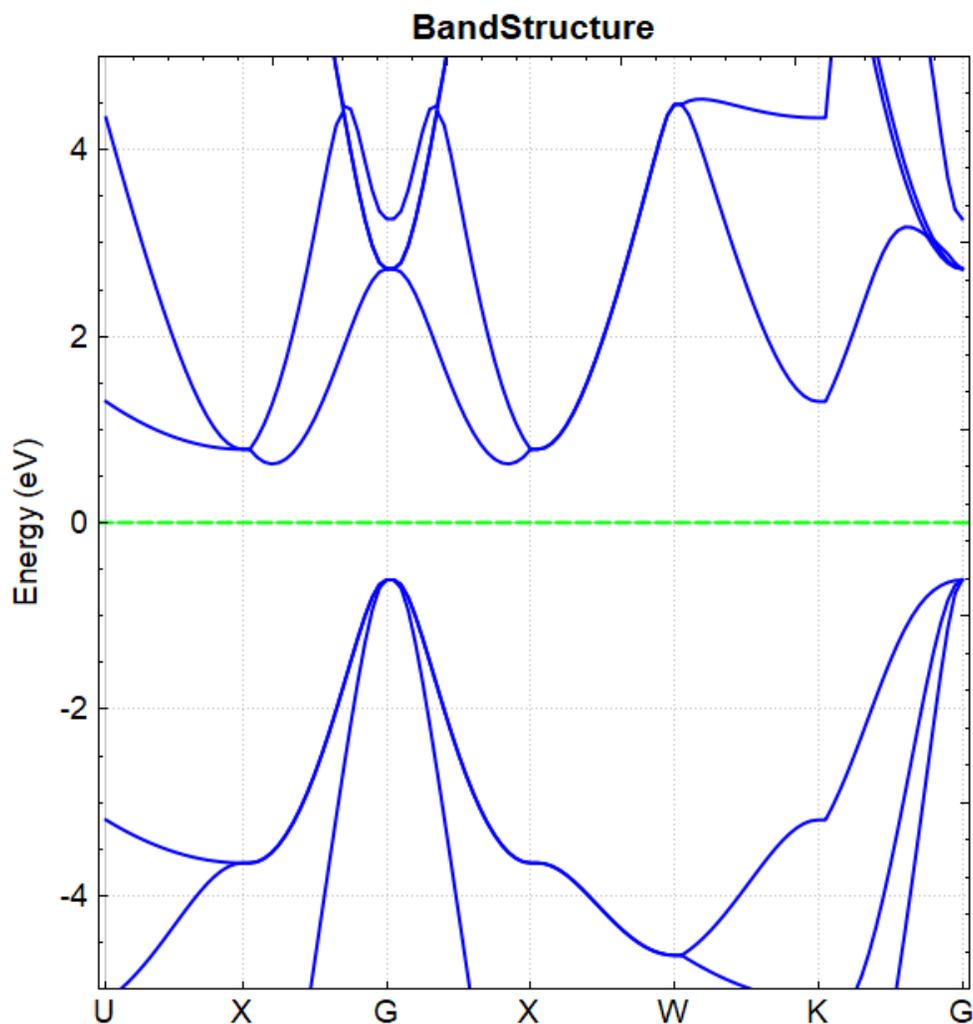
4. 杂化泛函计算也可使用 **block Davidson** 方法进行电子自洽计算，即此例所设 `cal.methods = 1`，此时 `scf.mixType` 参数会默认为 `Kerker`

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：



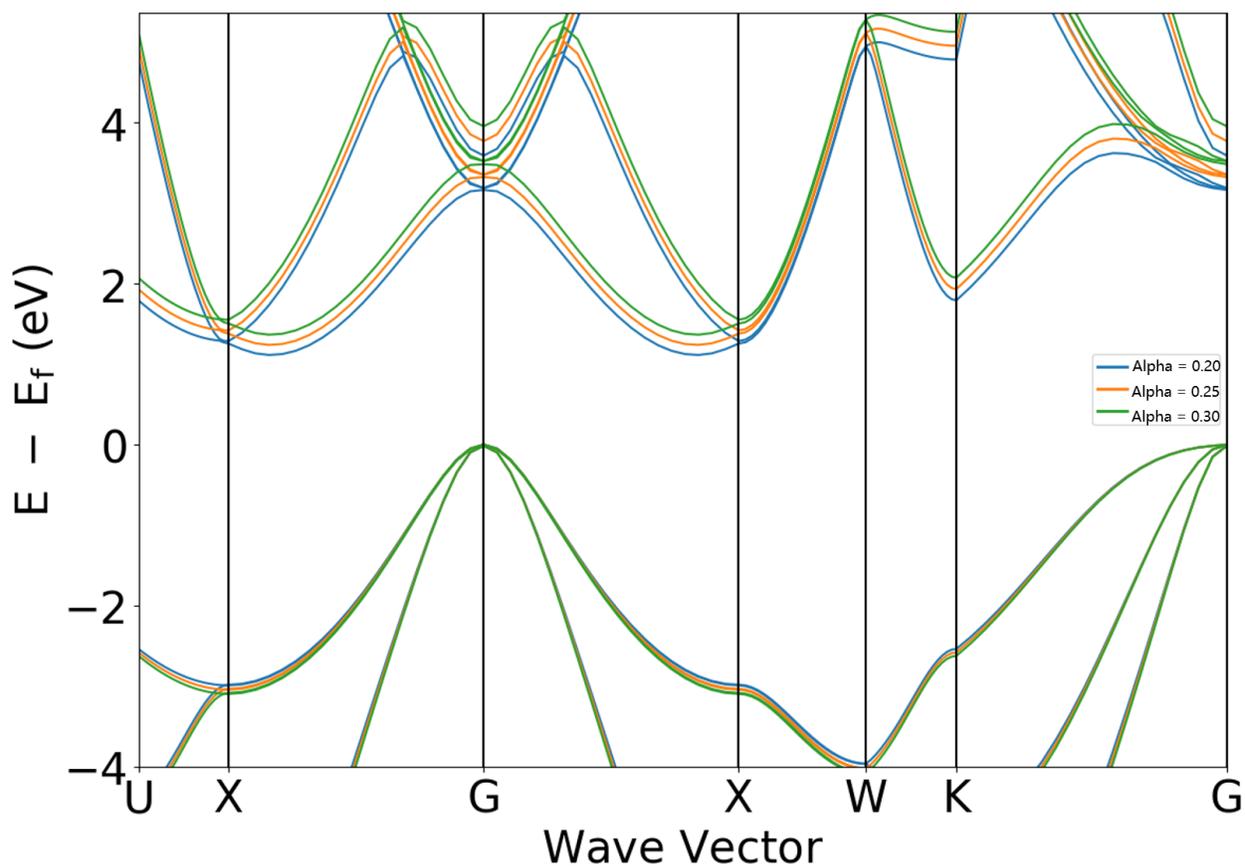
能带图表明打开杂化泛函计算后价带与导带之间的带隙变大，约为 **1.2394 eV**，不进行杂化泛函计算得到的能带带隙约为 **0.6433 eV**。

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：

```

1 # task type
2 task = relax
3 #system related
4 sys.structure = structure.as
5 sys.symmetry = true
6 sys.functional = PBE
7 sys.spin = none
8 #scf related
9 cal.methods = 1
10 cal.smearing = 1
11 cal.ksampling = G
12 cal.kpoints = [21, 21, 7]
13 cal.cutoff = 600
14 scf.convergence = 1.0e-05
15 #relax related
16 relax.max = 60
17 relax.freedom = all
18 relax.convergence = 0.01
19 relax.methods = CG
20 #vdw related
21 corr.VDW = true
22 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：

```

1 Total number of atoms
2 4
3 Lattice
4 2.46729136 0.00000000 0.00000000
5 -1.23364568 2.13673699 0.00000000
6 0.00000000 0.00000000 7.80307245
7 Cartesian
8 C 0.00000000 0.00000000 1.95076811
9 C 0.00000000 0.00000000 5.85230434
10 C 0.00000000 1.42449201 1.95076811
11 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` 的可选项有 **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：

```

1 # task type
2 task = relax
3 #system related
4 sys.structure = structure.as
5 sys.symmetry = true
6 sys.spin = none
7 #scf related
8 cal.methods = 1
9 cal.smearing = 1
10 cal.ksampling = G
11 cal.kpoints = [21, 21, 7]
12 cal.cutoff = 600
13 scf.convergence = 1.0e-05
14 #relax related
15 relax.max = 60
16 relax.freedom = all
17 relax.convergence = 0.01
18 relax.methods = CG
19 #vdw related
20 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 \mathbf{c} 的值与实验报道结果¹ 更接近。

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：

```

1 # task type
2 task = optical
3 #system related
4 sys.structure = structure.as
5 sys.symmetry = true
6 sys.functional = PBE
7 sys.spin = none
8 #scf related
9 cal.methods = 1
10 cal.smearing = 1
11 cal.ksampling = G
12 cal.kpoints = [12, 12, 12]
13 cal.cutoffFactor = 1.5
14 cal.iniCharge = ./rho.bin
15
16 #optical related
17 optical.grid = 2000
18 optical.sigma = 0.05
19 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；

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.

- `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:

```

1 # task type
2 task = scf
3 #system related
4 sys.structure = structure.as
5 sys.symmetry = true
6 sys.functional = PBE
7 sys.spin = none
8 #scf related
9 cal.methods = 1
10 cal.smearing = 1
11 cal.ksampling = G
12 cal.kpoints = [12, 12, 12]
13 cal.cutoffFactor = 1.5
14 #optical related
15 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

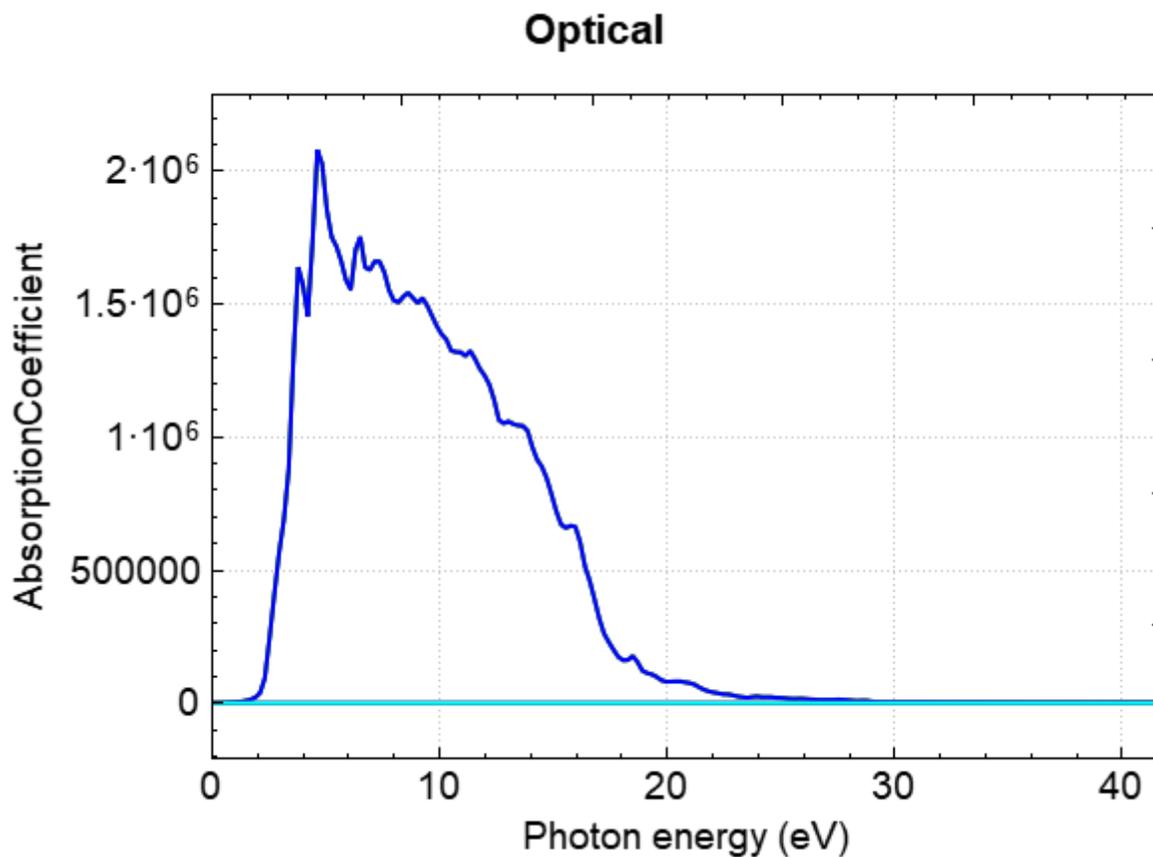
以两步算为例, 准备好输入文件之后, 将 `scf.in`、`optical.in` 和 `structure.as` 文件上传到服务器上运行, 按照结构弛豫中介绍的方法执行 `DS-PAW scf.in`、`optical.in`。

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：

```

1 # task type
2 task = frequency
3 #system related
4 sys.structure = structure.as
5 sys.symmetry = true
6 sys.functional = PBE
7 sys.spin = none
8 #scf related
9 cal.methods = 2
10 cal.smearing = 1
11 cal.ksampling = MP
12 cal.kpoints = [9, 9, 9]
13 cal.cutoffFactor = 1.5
14 scf.convergence = 1.0e-6
15 #frequency related
16 frequency.dispOrder = 1
17 frequency.dispRange = 0.02
18 #outputs

```

(续下页)

```

19 io.charge = false
20 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 $\pm\text{frequency.dispRange}$; 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 $\pm\text{frequency.dispRange}$ 和 $\pm 2 * \text{frequency.dispRange}$;
- *frequency.dispRange*: 设置频率计算时的原子位移大小;

structure.as 文件参考如下:

```

1 Total number of atoms
2 2
3 Lattice
4 8.0 0.0 0.0
5 0.0 8.0 0.0
6 0.0 0.0 8.0
7 Cartesian  Fix_x Fix_y Fix_z
8 O 0 0 0      T T F
9 C 0 0 1.143  T T F

```

i 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：

```

1 # task type
2 task = elastic
3 #system related
4 sys.structure = structure.as
5 sys.symmetry = true
6 sys.functional = PBE
7 sys.spin = none
8 #scf related
9 cal.methods = 1
10 cal.smearing = 1
11 cal.ksampling = G
12 cal.kpoints = [5, 5, 5]
13 cal.cutoffFactor = 1.5
14 scf.convergence = 1.0e-6
15 #frequency related
16 elastic.dispOrder = 1
17 elastic.dispRange = 0.01
18 #outputs
19 io.charge = false
20 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：

```

1 Total number of atoms
2 8
3 Lattice
4 5.43070000 0.00000000 0.00000000
5 0.00000000 5.43070000 0.00000000
6 0.00000000 0.00000000 5.43070000
7 Cartesian
8 Si 0.67883750 0.67883750 0.67883750
9 Si 3.39418750 3.39418750 0.67883750

```

(续下页)

(接上页)

```

10 Si 3.39418750 0.67883750 3.39418750
11 Si 0.67883750 3.39418750 3.39418750
12 Si 2.03651250 2.03651250 2.03651250
13 Si 4.75186250 4.75186250 2.03651250
14 Si 4.75186250 2.03651250 4.75186250
15 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 (CI-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:

```

1 task = neb
2
3 sys.structure = structure.as
4 sys.functional = PBE
5 sys.spin = none
6 sys.symmetry = true
7
8 cal.ksampling = G
9 cal.kpoints = [3,3,1]
10 cal.cutoffFactor = 1.0
11 cal.smearing = 1
12 cal.sigma = 0.05
13
14 neb.freedom = atom
15 neb.springK = 5
16 neb.images = 3
17 neb.iniFin = true
18 neb.method = LBFGS
19 neb.convergence = 0.03
20 neb.stepRange = 0.1
21 neb.max = 60
22
23 io.wave = false
24 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

```

1 Total number of atoms
2 13
3 Lattice
4 5.60580000 0.00000000 0.00000000
5 0.00000000 5.60580000 0.00000000
6 0.00000000 0.00000000 16.81740000
7 Cartesian Fix_x Fix_y Fix_z
8 H 2.80881670 4.20393628 6.94088012 F F F
9 Pt 1.40145000 1.40145000 1.98192999 T T T
10 Pt 4.20434996 1.40145000 1.98192999 T T T
11 Pt 1.40145000 4.20434996 1.98192999 T T T
12 Pt 4.20434996 4.20434996 1.98192999 T T T
13 Pt 0.00272621 0.00056545 3.91746017 F F F
14 Pt 0.00271751 2.80233938 3.91708172 F F F
15 Pt 2.80568712 -0.00141176 3.91894328 F F F
16 Pt 2.80548220 2.80426217 3.91792247 F F F
17 Pt 1.39865124 1.40124680 5.84694340 F F F
18 Pt 4.21951864 1.40156999 5.84719575 F F F
19 Pt 1.38647954 4.20437926 5.89984296 F F F
20 Pt 4.23154392 4.20414605 5.89983612 F F F

```

末态结构 `structure04.as` 参考如下

```

1 Total number of atoms
2 13
3 Lattice
4 5.60580000 0.00000000 0.00000000
5 0.00000000 5.60580000 0.00000000
6 0.00000000 0.00000000 16.81740000
7 Cartesian Fix_x Fix_y Fix_z
8 H 1.52157824 2.80289997 6.91583941 F F F
9 Pt 1.40145000 1.40145000 1.98192999 T T T
10 Pt 4.20434997 1.40145000 1.98192999 T T T
11 Pt 1.40145000 4.20434997 1.98192999 T T T
12 Pt 4.20434997 4.20434997 1.98192999 T T T
13 Pt 0.02556963 0.00000000 3.90765450 F F F
14 Pt 0.02708862 2.80290000 3.91082177 F F F
15 Pt 2.83159105 0.00000000 3.91547525 F F F
16 Pt 2.82981856 2.80290000 3.90913282 F F F
17 Pt 1.45998966 1.38039927 5.88134827 F F F
18 Pt 4.25691060 1.38811299 5.84551487 F F F
19 Pt 1.45998966 4.22540069 5.88134827 F F F
20 Pt 4.25691060 4.21768697 5.84551487 F F F

```

i 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。

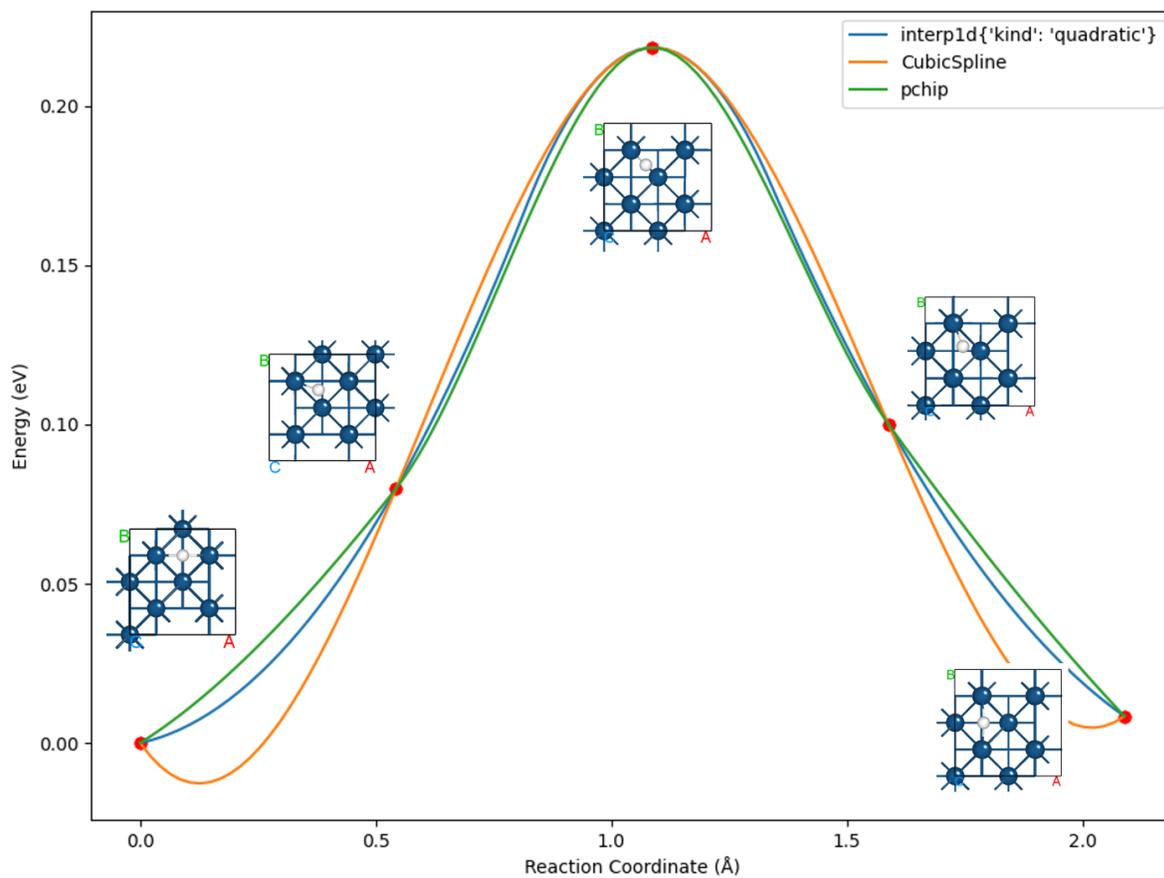
- `DS-PAW.log`：DS-PAW 过渡态计算之后得到的日志文件；
- `neb.h5`：过渡态计算完成之后的 **h5** 数据文件；此时反应坐标及能量变化等数据被保存在 `neb.h5` 中，具体的数据结构详见输出文件格式说明部分；

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。

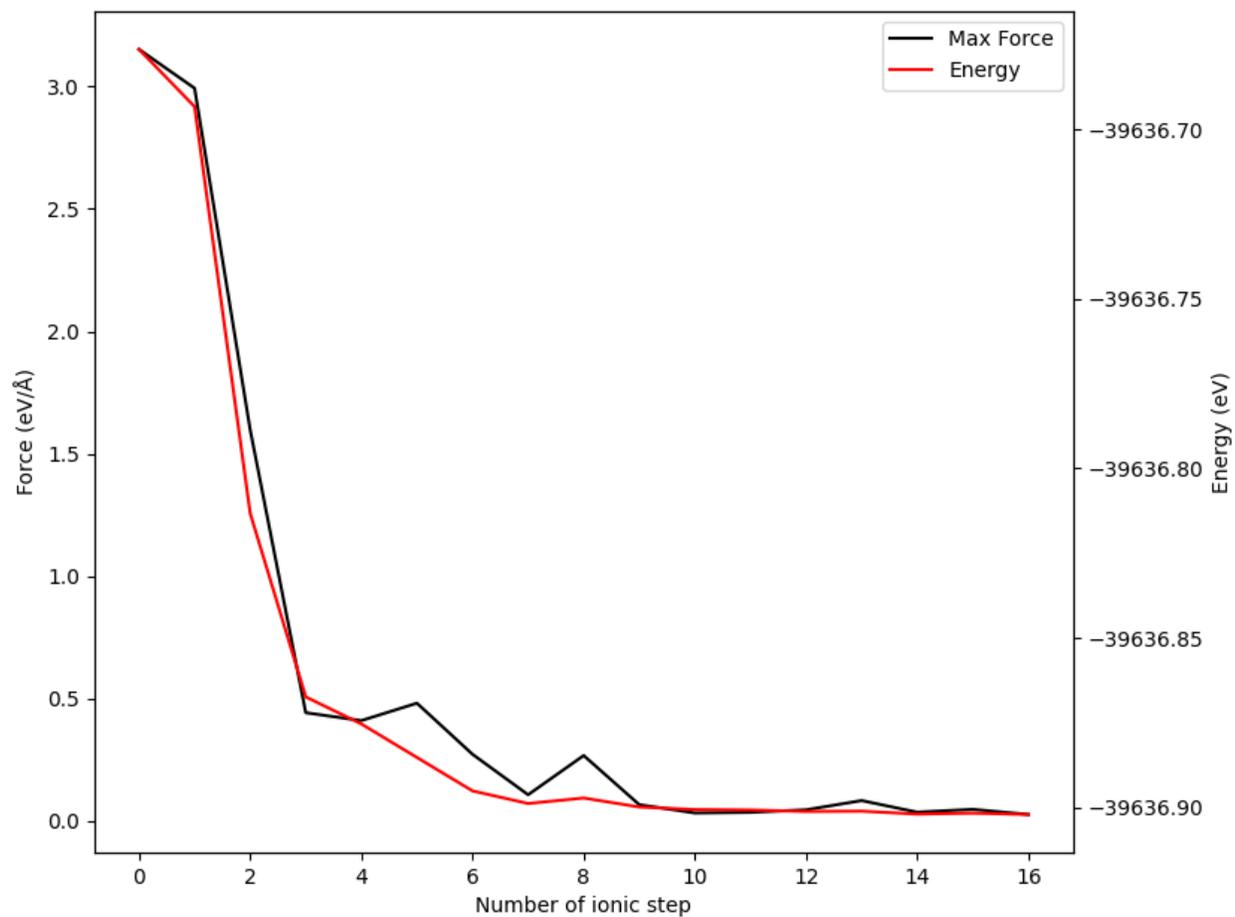
处理可得到 NEB 计算各构型的能量和受力表格：

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

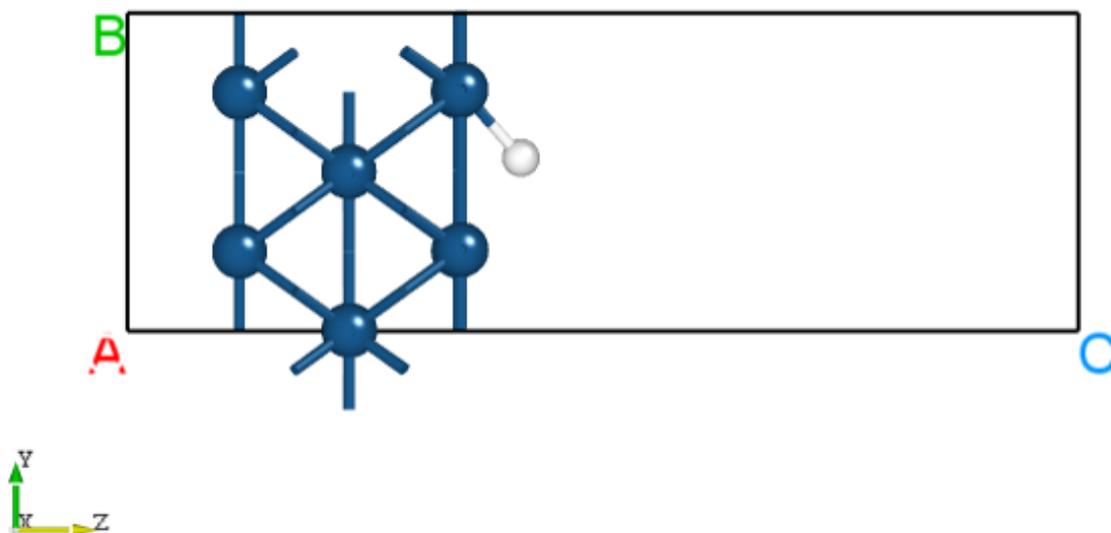
处理得到的势垒曲线效果应如下所示：



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：

```

1 task = phonon
2
3 sys.structure = structure.as
4 sys.functional = PBE
5 sys.spin = none
6
7 cal.methods = 1
8 cal.smearing = 1
9 sys.symmetry = true
10 scf.convergence = 1.0e-07
11 cal.ksampling = G
12 cal.kpoints = [3,3,3]
13 cal.sigma = 0.25
14
15 phonon.type = bandDos
16 phonon.structureSize = [2,2,2]
17 phonon.primitiveUVW = [0.0, 0.5, 0.5, 0.5, 0.0, 0.5, 0.5, 0.5, 0.0]
18 phonon.method = dfpt
19 phonon.qpoints = [41,41,41]

```

(续下页)

(接上页)

```

20 phonon.dosRange = [0,20]
21 phonon.qpointsLabel = [G,X,W,G,M]
22 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]
23 phonon.qpointsNumber = 51
24
25 io.charge = false
26 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:

```

1 Total number of atoms
2 8
3 Lattice
4 4.2555564654942897 0.0000000000000000 0.0000000000000000
5 0.0000000000000000 4.2555564654942888 0.0000000000000000
6 0.0000000000000000 0.0000000000000000 4.2555564654942897
7 Direct
8 Mg 0.0000000000000000 0.0000000000000000 0.0000000000000000
9 Mg 0.0000000000000000 0.5000000000000000 0.5000000000000000
10 Mg 0.5000000000000000 0.0000000000000000 0.5000000000000000
11 Mg 0.5000000000000000 0.5000000000000000 0.0000000000000000
12 O 0.5000000000000000 0.5000000000000000 0.5000000000000000
13 O 0.5000000000000000 0.0000000000000000 0.0000000000000000
14 O 0.0000000000000000 0.5000000000000000 0.0000000000000000
15 O 0.0000000000000000 0.0000000000000000 0.5000000000000000

```

i 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

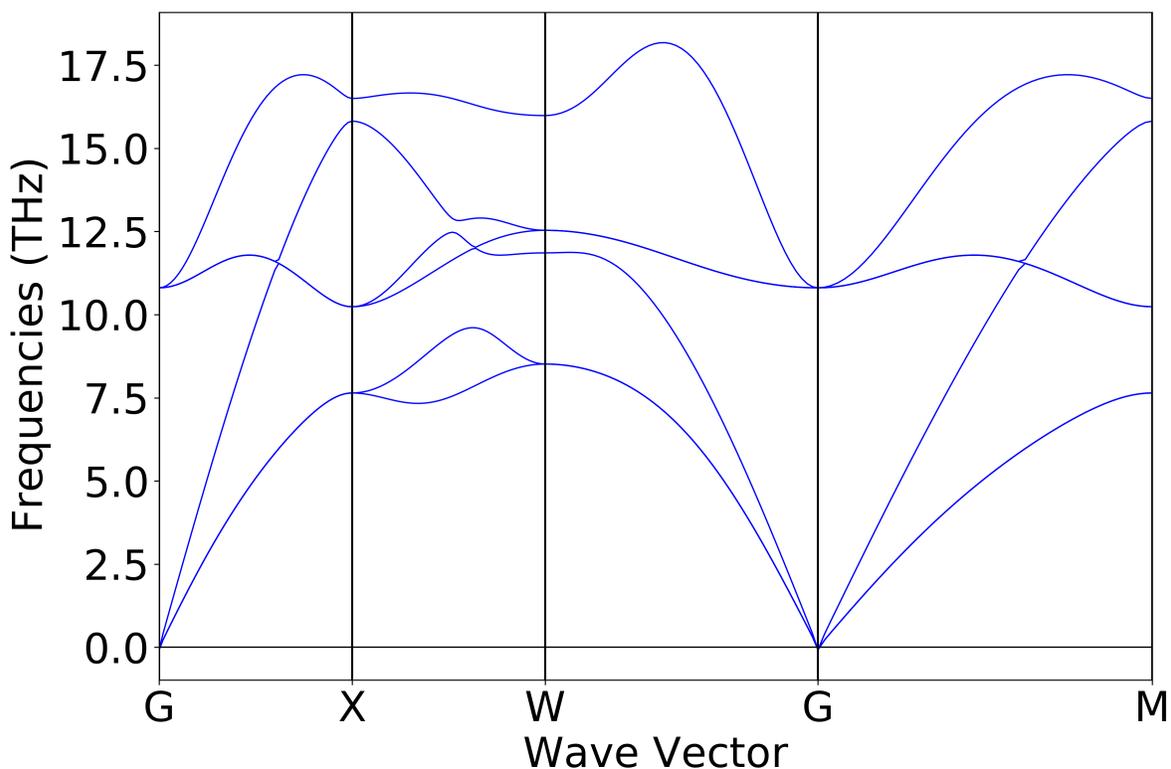
准备好输入文件之后，将 *phonon.in* 和 *structure.as* 文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 DS-PAW *phonon.in*。

2.16.3 analysis 计算结果分析

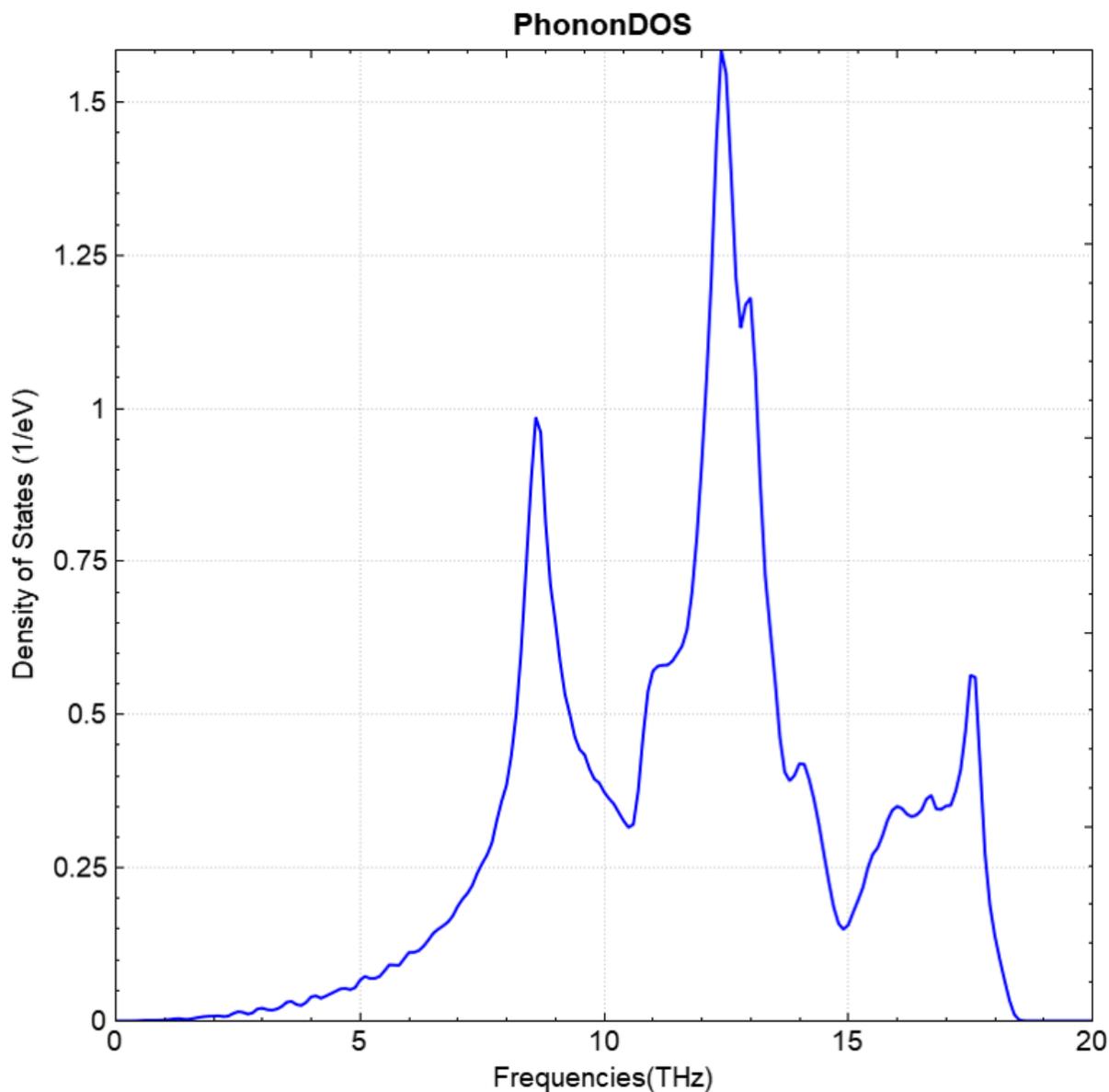
根据上述的输入文件，计算完成之后将会得到 *DS-PAW.log*、*phonon.h5*、*dfpt.json* 和 *dfpt.as* 等输出文件。

- *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)



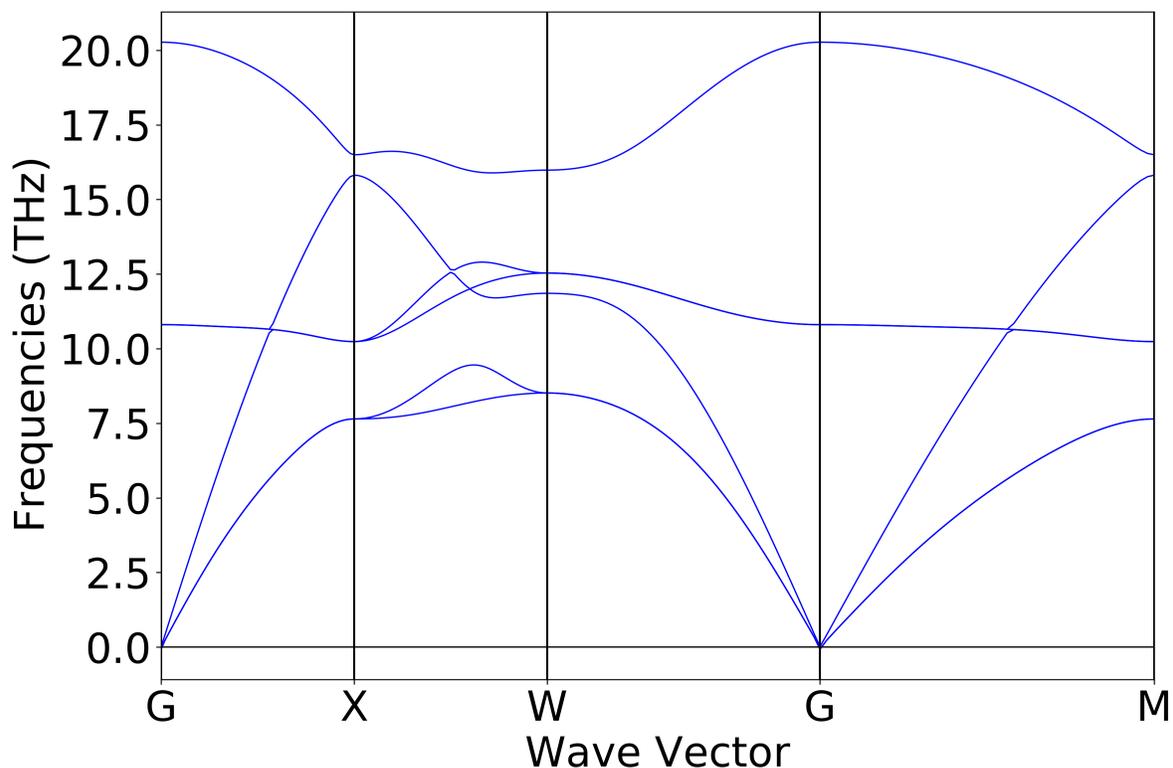
(b)

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：

```
1 phonon.dfptEpsilon=true
2 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。

```

1 task = phonon
2
3 sys.structure = structure.as
4 sys.functional = PBE
5 sys.spin = none
6
7 cal.methods = 1
8 cal.smearing = 1
9 sys.symmetry = true
10 scf.convergence = 1.0e-07
11 cal.ksampling = G
12 cal.kpoints = [3,3,3]
13 cal.sigma = 0.25
14
15 phonon.type = bandDos
16 phonon.structureSize = [2,2,2]
17 phonon.primitiveUVW = [0.0, 0.5, 0.5, 0.5, 0.0, 0.5, 0.5, 0.5, 0.0]
18 phonon.method = fd

```

(续下页)

(接上页)

```

19 phonon.qpoints = [41,41,41]
20 phonon.qpointsLabel = [G,X,W,G,M]
21 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]
22 phonon.qpointsNumber = 51
23
24 io.charge = false
25 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。

i 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：

```

1 # task type
2 task = scf
3 #system related
4 sys.structure = structure.as
5 sys.symmetry = false
6 sys.functional = PBE
7 #scf related
8 cal.methods = 2
9 cal.smearing = 1
10 cal.ksampling = G
11 cal.kpoints = [7, 7, 7]
12 cal.cutoffFactor = 1.5
13 #soi related
14 sys.spin= non-collinear
15 sys.soi = true
16 #outputs

```

(续下页)

```

17 io.charge = true
18 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:

```

1 Total number of atoms
2 5
3 Lattice
4 -2.069 -3.583614 0.000000
5 2.069 -3.583614 0.000000
6 0.000 2.389075 9.546667
7 Direct
8 Bi 0.3990 0.3990 0.6970
9 Bi 0.6010 0.6010 0.3030
10 Se 0.0000 0.0000 0.5000
11 Se 0.2060 0.2060 0.1180
12 Se 0.7940 0.7940 0.8820

```

能带计算的输入文件 *soiband.in*, 内容如下

```

1 # task type
2 task = band
3 #system related
4 sys.structure = structure.as
5 sys.symmetry = true
6 sys.functional = PBE
7 #scf related
8 cal.methods = 2
9 cal.smearing = 1
10 cal.ksampling = G
11 cal.kpoints = [7, 7, 7]
12 cal.cutoffFactor = 1.5
13 #band related
14 cal.iniCharge = ./rho.bin
15 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]
16 band.kpointsLabel = [G,Z,F,G,L]
17 band.kpointsNumber = [20,20,20,20]
18 band.project = true
19 #soi related
20 sys.spin= non-collinear
21 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定参数即可。

i 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。

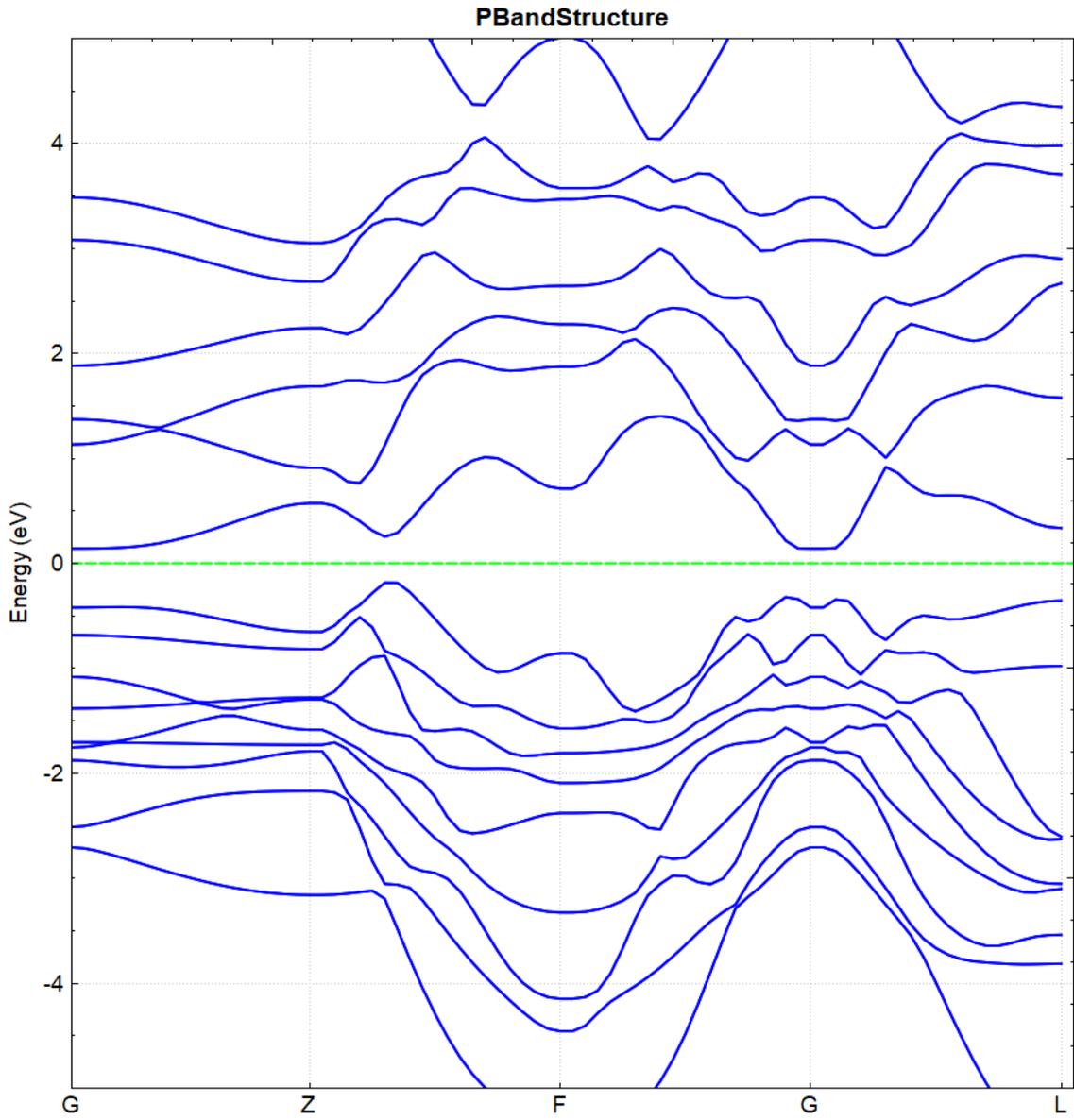
2.17.2 run 程序运行

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`。

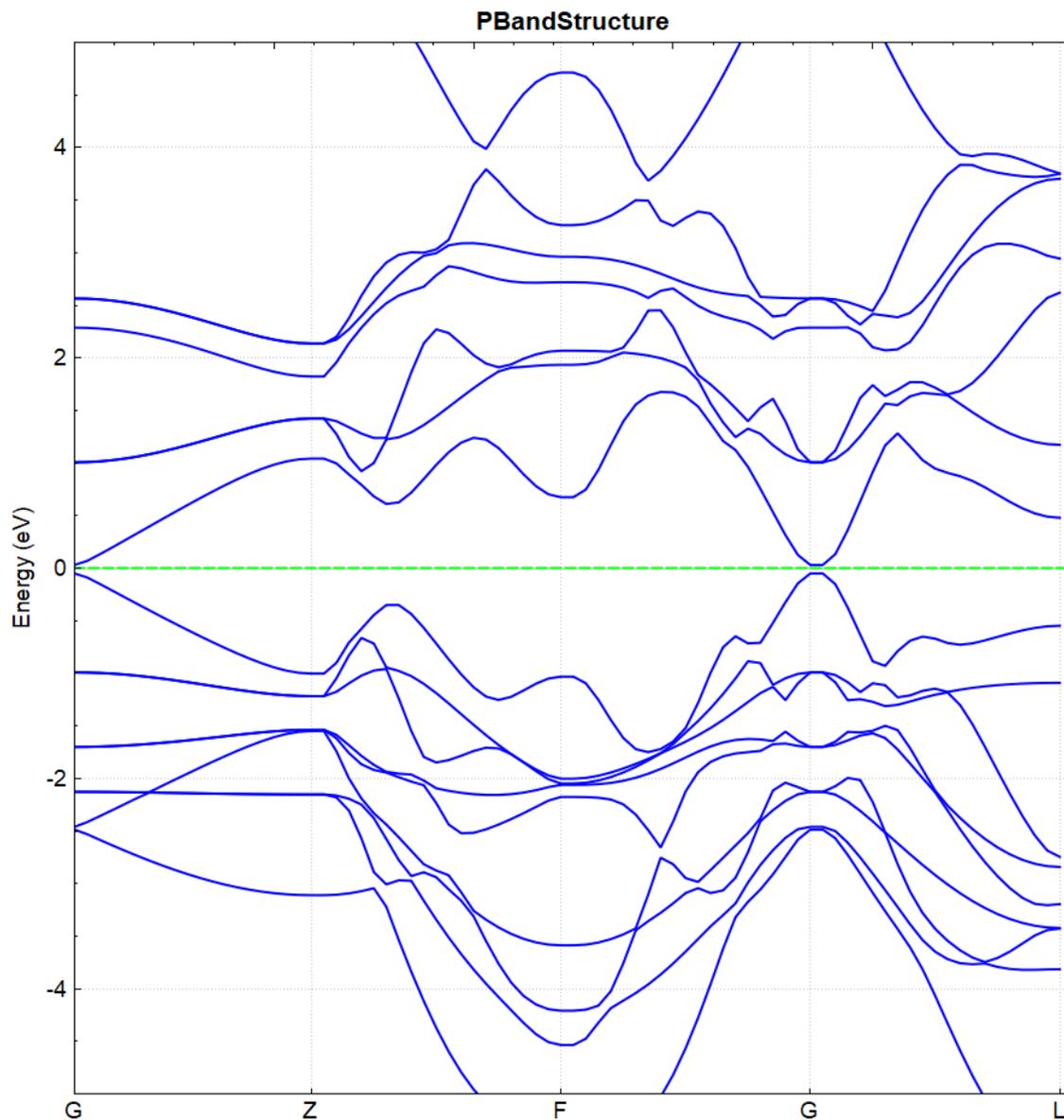
2.17.3 analysis 计算结果分析

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:

```
1 #task type
2 task = aimd
3
4 #system related
5 sys.structure = structure.as
6 sys.symmetry = false
7 sys.functional = PBE
8 sys.spin = none
9
10 #scf related
11 cal.methods = 1
12 cal.smearing = 1
13 cal.ksampling = G
14 cal.kpoints = [1, 1, 1]
15 cal.sigma = 0.1
16
17 #aimd related
18 aimd.ensemble = NPT
19 aimd.thermostat = langevin
20 aimd.atomFCoeffElements = [H_1]
21 aimd.atomFCoeffs = [1]
22 aimd.latticeFCoeff = 1
23 aimd.pressure = 100
24 aimd.timeStep = 1
25 aimd.totalSteps = 2000
26 aimd.iniTemp = 2000
27
28 #outputs
29 io.charge = false
30 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:

```

1 Total number of atoms
2 3
3 Lattice
4 4.00000000 0.00000000 0.00000000
5 0.00000000 4.00000000 0.00000000
6 0.00000000 0.00000000 4.00000000
7 Cartesian
8 H 2.63934013 1.89542007 1.58223984
9 H_1 1.36065987 2.11498988 2.45934006
10 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`。

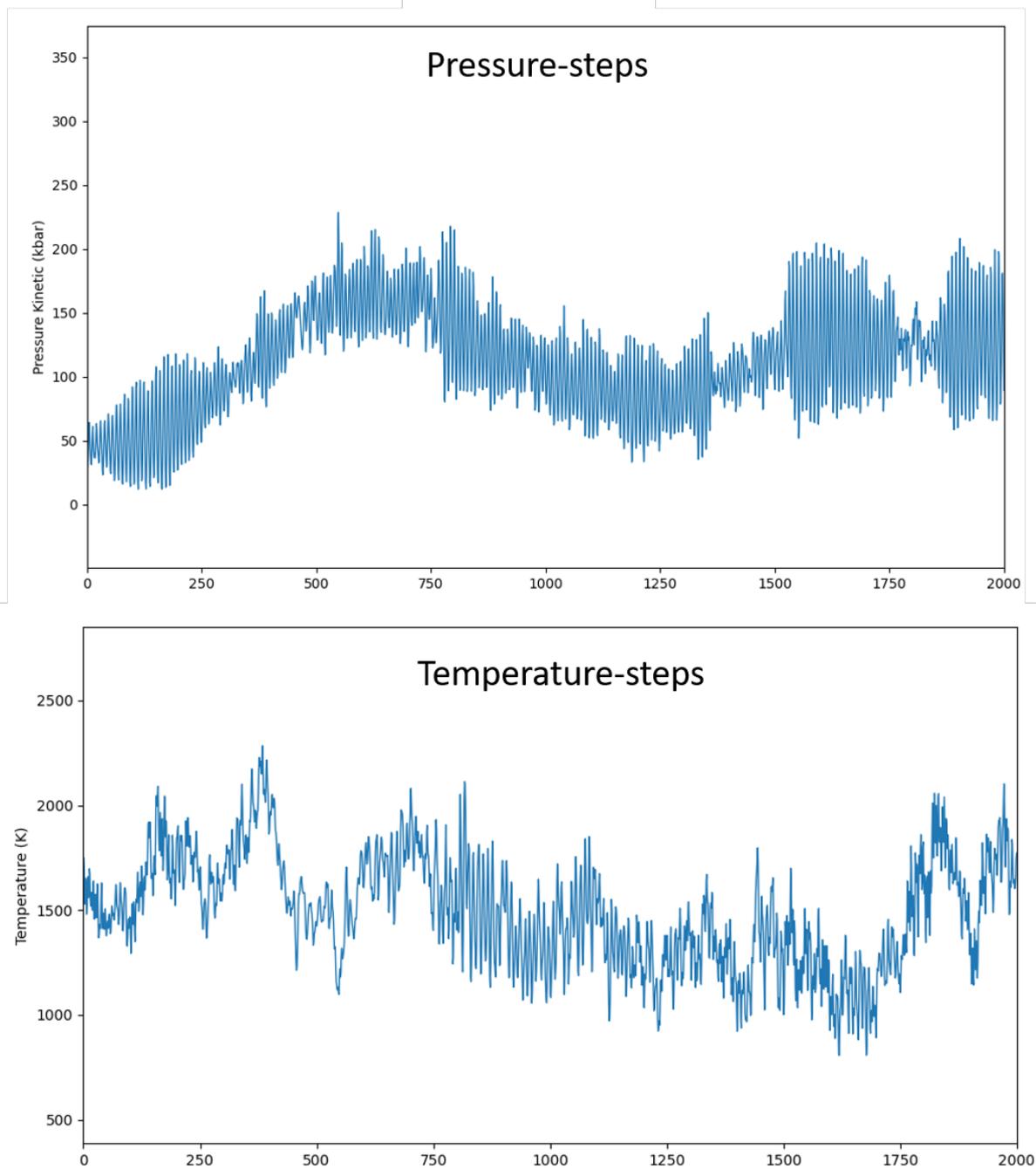
2.18.3 analysis 计算结果分析

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：

DSPA W AIMD

**i** 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：

```

1 # task type
2 task = scf
3 #system related
4 sys.structure = structure.as
5 sys.symmetry = true
6 sys.functional = PBE
7 sys.spin = none
8
9 #scf related
10 cal.sigma = 0.1
11 cal.cutoff = 520
12 cal.ksampling = G
13 cal.kpoints = [9, 9, 1]
14
15 scf.convergence = 1e-5
16
17 #outputs
18 io.charge = false
19 io.wave = false
20 io.band = true
21
22 corr.dipol=true
23 corr.dipolDirection = c
24 corr.dipolEfield = 0.2
25
26 band.kpointsLabel = [G,M,K,G]
27 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]
28 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:

```
1 Total number of atoms
2 2
3 Lattice
4 3.860000 0.000000 0.000000
5 -1.930000 3.342860 0.000000
6 0.000000 0.000000 26.460000
7 Direct
8 Si 0.333333 0.166667 0.396825
9 Si 0.666758 0.833380 0.379216
```

2.19.2 run 程序运行

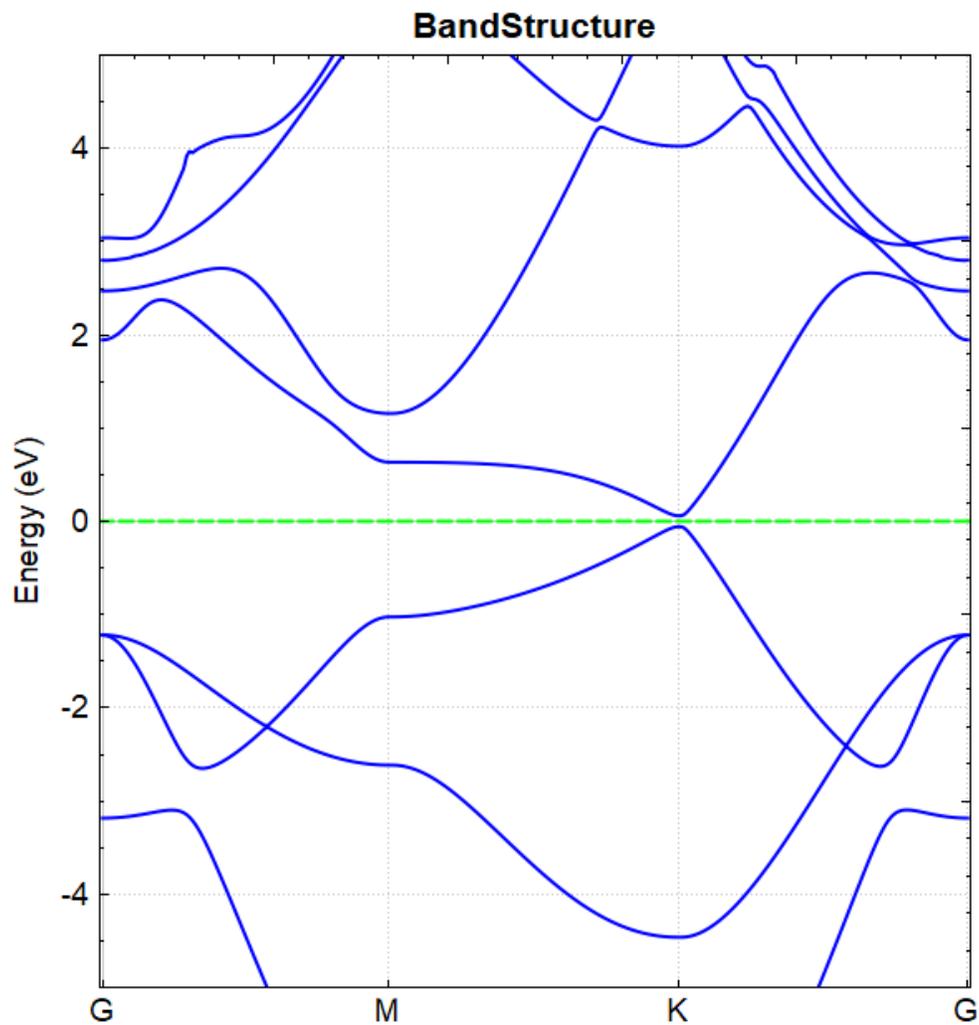
准备好输入文件之后, 将 `Efield.in` 和 `structure.as` 文件上传到服务器上运行, 按照结构弛豫中介绍的方法执行 `DS-PAW Efield.in`。

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。

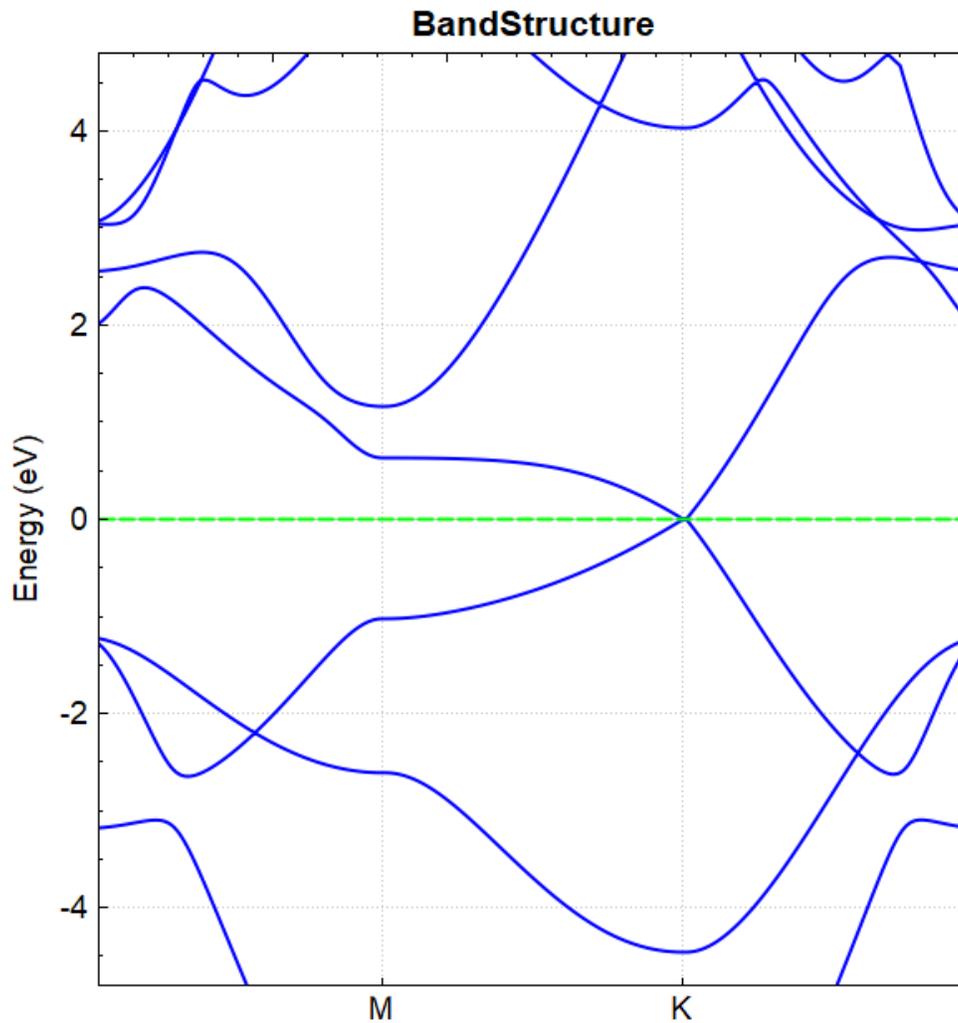
`scf.h5`: 自洽计算对应的 **h5** 输出文件, 当 `io.band = true` 时, `scf.h5` 文件会写入能带数据;

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.dipoleEfield = 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/\AA 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:

```

1 # task type
2 task = scf
3 #system related
4 sys.structure = structure.as
5 sys.symmetry = true
6 sys.functional = PBE
7 sys.spin = none
8
9 #scf related
10 cal.methods = 3
11 cal.smearing = 4
12 cal.cutoff = 520
13 cal.ksampling = MP
14 cal.kpoints = [4, 4, 4]
15
16 scf.convergence = 1e-5
17
18 #outputs
19 io.charge = false
20 io.wave = false
21 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:

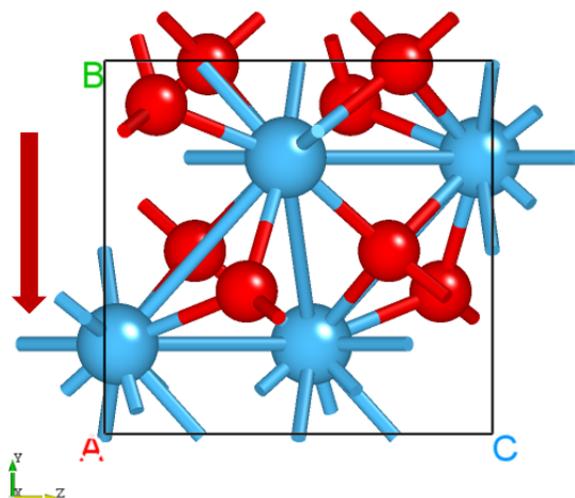
- io.polarization: 控制自洽计算中铁电计算的开关;

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

```

1 Total number of atoms
2 12
3 Lattice
4 5.04621935 0.00000000 0.00000000
5 0.00000000 5.07315250 0.00000000
6 0.00000000 0.00000000 5.25768906
7 Cartesian
8 Hf 1.34815269 1.22145222 0.17639072
9 Hf 1.34815269 3.75802848 2.45245381
10 Hf 3.69806665 1.22145222 2.80523525
11 Hf 3.69806665 3.75802848 5.08129834
12 O 0.35195212 1.93667284 1.92589951
13 O 0.35195212 4.47324910 0.70294502
14 O 2.32678304 2.48829365 3.85528783
15 O 2.32678304 5.02486989 4.03124575
16 O 2.71943629 5.02486989 1.40240122
17 O 2.71943629 2.48829365 1.22644331
18 O 4.69426723 1.93667284 4.55474404
19 O 4.69426723 4.47324910 3.33178954

```



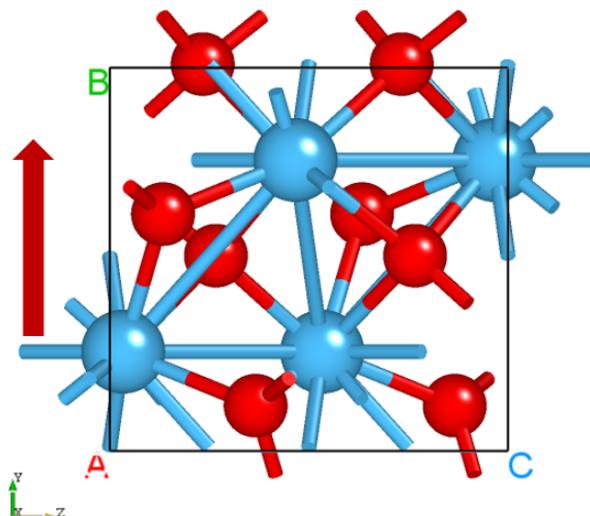
铁电相极化方向向下

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

```

1 Total number of atoms
2 12
3 Lattice
4 5.04621935 0.00000000 0.00000000
5 0.00000000 5.07315250 0.00000000
6 0.00000000 0.00000000 5.25768906
7 Cartesian
8 Hf 1.34815269 1.31512402 0.17639072
9 Hf 1.34815269 3.85170026 2.45245381
10 Hf 3.69806665 1.31512402 2.80523525
11 Hf 3.69806665 3.85170026 5.08129834
12 O 0.35195212 0.59990340 1.92589951
13 O 0.35195212 3.13647965 0.70294502
14 O 2.32678304 2.58485884 4.03124575
15 O 2.32678304 5.12143510 3.85528783
16 O 2.71943630 5.12143510 1.22644331
17 O 2.71943630 2.58485884 1.40240122
18 O 4.69426723 0.59990340 4.55474404
19 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。

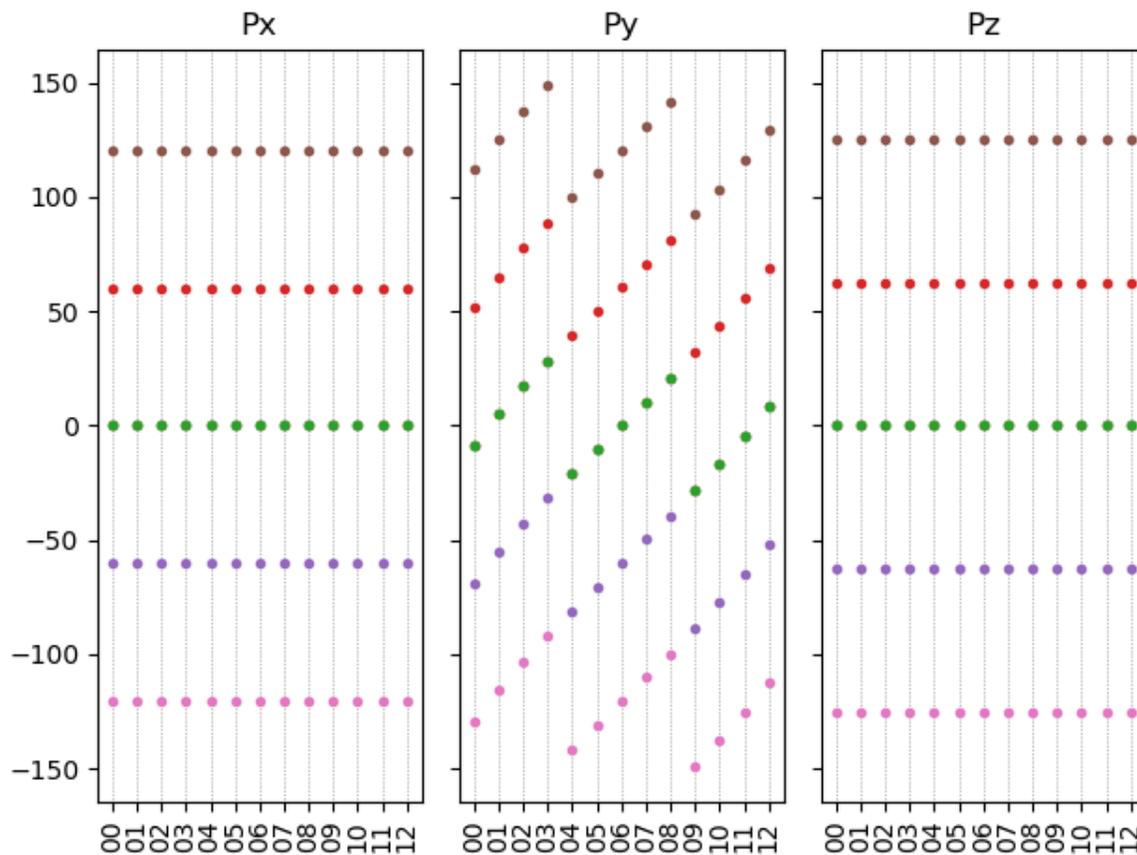
以极化向下（00）铁电相体系为例，从 `polarization.txt` 文件可得 HfO_2 的铁电极化数据如下所示：

Total(x y z) (($\mu C/cm^2$))		
-0.000043	-8.715604	-0.000002
Quantum(x y z) ($\mu C/cm^2$)		
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) (($\mu C/cm^2$))		
-0.000049	8.715446	0.000001
Quantum(x y z) ($\mu C/cm^2$)		
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 P_x, P_y, P_z 。因 HfO_2 极化方向为 y 方向， P_x, P_z 数值不随原子位移而改变。

取 P_y 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 $-69.103 \mu C/cm^2$

12 与 06 构型的极化差值为 $69.103 \mu C/cm^2$

遂， HfO_2 的极化强度值为 $69.103 \mu C/cm^2$

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:

```

1 # task type
2 task = scf
3 #system related
4 sys.structure = structure.as
5 sys.symmetry = true
6 sys.functional = PBE
7 sys.spin = none
8
9 #scf related
10 cal.methods = 1
11 cal.smearing = 1
12 cal.ksampling = G
13 cal.kpoints = [10, 10, 10]
14 cal.cutoff = 650
15 #outputs
16 io.charge = true
17 io.wave = false
18 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:

```

1 Total number of atoms
2 8
3 Lattice
4 5.68452692 0.00000000 0.00000000
5 0.00000000 5.68452692 0.00000000
6 0.00000000 0.00000000 5.68452692
7 Cartesian
8 Na 4.26339519 1.42113173 1.42113173
9 Na 1.42113173 4.26339519 1.42113173
10 Na 1.42113173 1.42113173 4.26339519
11 Na 4.26339519 4.26339519 4.26339519
12 Cl 1.42113173 1.42113173 1.42113173
13 Cl 4.26339519 4.26339519 1.42113173
14 Cl 4.26339519 1.42113173 4.26339519
15 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：

```

1 task = scf
2
3 sys.structure = structure.as
4 sys.symmetry = true
5 sys.functional = PBE
6 sys.spin = none

```

(续下页)

(接上页)

```

7
8 cal.methods = 1
9 cal.smearing = 1
10 cal.ksampling = MP
11 cal.kpoints = [3, 3, 3]
12 cal.cutoff = 650
13
14 scf.convergence = 1.0e-05
15
16 io.charge = true
17 io.wave = false

```

bandunfolding.in 如下:

```

1 task = band
2 cal.iniCharge = ./rho.bin
3
4 sys.structure = structure.as
5 sys.symmetry = true
6 sys.functional = PBE
7 sys.spin = none
8
9 cal.methods = 1
10 cal.smearing = 1
11 cal.ksampling = MP
12 cal.kpoints = [3, 3, 3]
13 cal.cutoff = 500
14
15 scf.convergence = 1.0e-05
16
17 band.unfolding = true
18 band.primitiveUVW=[0.0, 0.5, 0.5, 0.5, 0.0, 0.5, 0.5, 0.5, 0.0]
19 band.kpointsLabel= [R,G,X]
20 band.kpointsCoord= [0.5, 0.5, 0.5, 0.0, 0.0, 0.0, 0.5, 0.0, 0.5]
21 band.kpointsNumber= [101, 101]
22
23 io.charge = false
24 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，新增参数如下:

- `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:

```

1 Total number of atoms
2 4
3 Lattice
4 3.7530000210      0.0000000000      0.0000000000
5 0.0000000000      3.7530000210      0.0000000000
6 0.0000000000      0.0000000000      3.7530000210
7 Direct

```

(续下页)

(接上页)

8	Au	0.000000000	0.000000000	0.000000000
9	Cu	0.000000000	0.500000000	0.500000000
10	Cu	0.500000000	0.000000000	0.500000000
11	Cu	0.500000000	0.500000000	0.000000000

2.22.2 run 程序运行

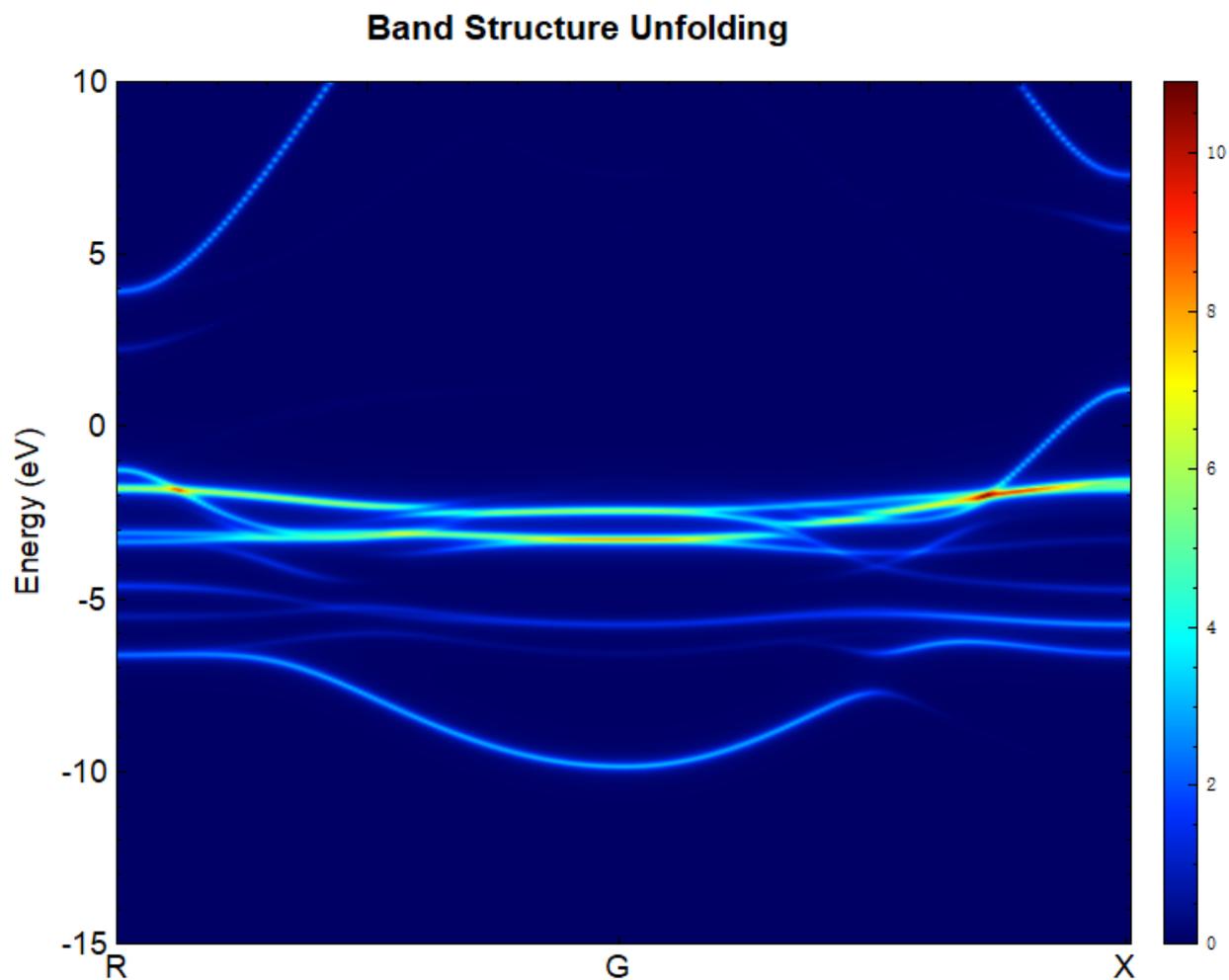
准备好输入文件之后，将 *scf.in*、*bandunfolding.in* 和 *structure.as* 文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 *DS-PAW scf.in*，自洽计算完成后执行 *DS-PAW bandunfolding.in*。

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.

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：

```

1 # task type
2 task = epsilon
3 #system related
4 sys.structure = structure.as
5 sys.symmetry = true
6 sys.functional = PBE
7 sys.spin = none
8
9 #scf related
10 cal.methods = 1
11 cal.smearing = 1
12 cal.ksampling = G
13 cal.kpoints = [5, 5, 5]
14 cal.cutoff = 500
15 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：

```

1 Total number of atoms
2 8
3 Lattice
4 5.43070000 0.00000000 0.00000000
5 0.00000000 5.43070000 0.00000000
6 0.00000000 0.00000000 5.43070000
7 Cartesian
8 Si 0.67883750 0.67883750 0.67883750
9 Si 3.39418750 3.39418750 0.67883750
10 Si 3.39418750 0.67883750 3.39418750
11 Si 0.67883750 3.39418750 3.39418750

```

(续下页)

(接上页)

```

12 Si 2.03651250 2.03651250 2.03651250
13 Si 4.75186250 4.75186250 2.03651250
14 Si 4.75186250 2.03651250 4.75186250
15 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 AIN 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 AIN 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：

```

1 task = epsilon
2 #system related
3 sys.structure = structure.as
4 sys.symmetry = true
5 sys.functional = PBE
6 sys.spin = none
7

```

(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)

```

8 #scf related
9 cal.methods = 1
10 cal.smearing = 1
11 cal.ksampling = G
12 cal.kpoints = [10, 10, 10]
13 cal.cutoffFactor = 1.5
14 scf.convergence = 1.0e-7
15
16 #outputs
17 io.charge = false
18 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：

```

1 Total number of atoms
2 8
3 Lattice
4 3.11606630 0.00000000 0.00000000
5 0.00000000 5.39683518 0.00000000
6 0.00000000 0.00000000 5.00770902
7 Cartesian
8 Al 0.00000000 3.59735137 0.00946380
9 Al 0.00000000 1.79945276 2.51320124
10 Al 1.55803315 0.89899597 0.00945662
11 Al 1.55803315 4.49786165 2.51308138
12 N 0.00000000 3.59851112 1.91845914
13 N 0.00000000 1.79831356 4.42266820
14 N 1.55803315 0.90013952 1.91851680
15 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：

```

1 # task type
2 task = relax
3 #system related
4 sys.structure = structure.as
5 sys.symmetry = false
6 sys.functional = PBE
7 sys.spin = none
8
9 #scf related
10 cal.methods = 1
11 cal.smearing = 1
12 cal.ksampling = G
13 cal.cutoff = 650
14 cal.kpoints = [19, 19, 5]
15 #relax related
16 relax.freedom = all
17 relax.convergence = 0.05
18 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.

```

1 Total number of atoms
2 6
3 Lattice Fix_x Fix_y Fix_z
4 3.19031572 0.00000000 0.00000000 F T T
5 -1.59515786 2.76289446 0.00000000 F F T
6 0.00000000 0.00000000 14.87900448 T T T
7 Cartesian
8 S 0.00000000 1.84193052 12.72413785
9 S 1.59515943 0.92096386 5.28463561
10 S 0.00000000 1.84193052 9.59436887
11 S 1.59515943 0.92096386 2.15486663
12 Mo 1.59515943 0.92096386 11.15925336
13 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：

```

1 Total number of atoms
2 6
3 Lattice
4 3.19696732 0.00000000 0.00000000
5 -1.59848077 2.76865753 0.00000000
6 0.00000000 0.00000000 14.87900448
7 Direct
8 Mo 0.66666701 0.33333316 0.74999995
9 Mo 0.33333340 0.66666675 0.24999997
10 S 0.33333340 0.66666666 0.85535854
11 S 0.66666686 0.33333303 0.35535875
12 S 0.33333367 0.66666699 0.64464148
13 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：

```

1 # task type
2 task = phonon
3 #system related
4 sys.structure = structure.as
5 sys.symmetry = true
6 sys.functional = PBE
7 sys.spin = none
8
9 #scf related
10 cal.methods = 1
11 cal.smearing = 1
12 cal.ksampling = G
13 cal.kpoints = [5, 5, 5]
14 cal.cutoffFactor = 1.5
15 scf.convergence = 1.0e-7
16 #phonon related
17 phonon.structureSize = [2,2,2]
18 phonon.type =dos
19 phonon.qpoints = [31,31,31]
20 phonon.method = dfpt
21
22 phonon.thermal=true
23 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：

```

1 Total number of atoms
2 2
3 Lattice
4 0.00 2.75 2.75
5 2.75 0.00 2.75
6 2.75 2.75 0.00
7 Direct
8 Si -0.125000000 -0.125000000 -0.125000000
9 Si 0.125000000 0.125000000 0.125000000

```

2.26.2 run 程序运行

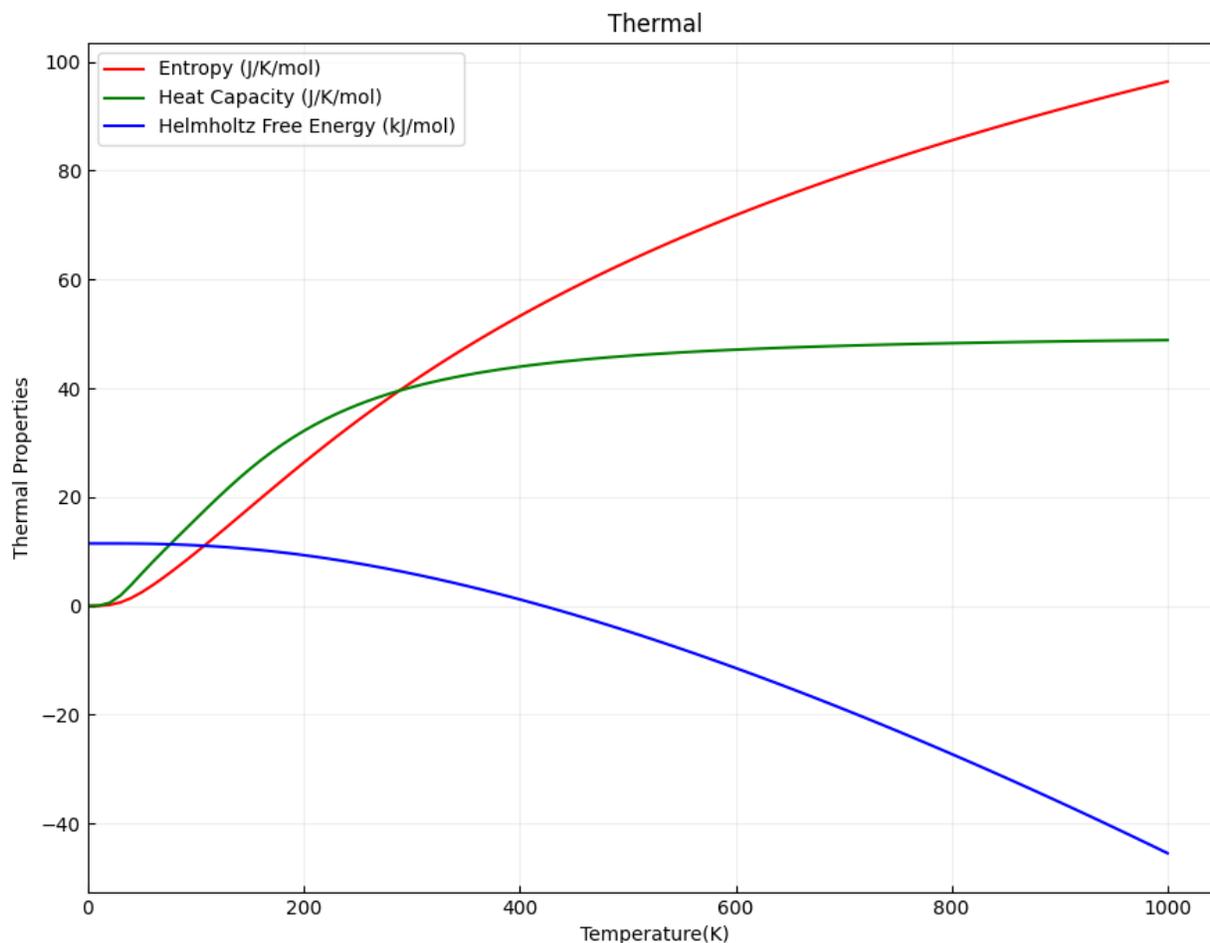
准备好输入文件之后，将 *phonon-thermal.in* 和 *structure.as* 文件上传到服务器上运行，按照结构弛豫中介绍的方法执行 DS-PAW *phonon-thermal.in*。

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:

```

1 task = neb
2
3 sys.structure = structure.as
4 sys.functional = LDA
5 sys.spin = none
6 sys.symmetry = false
7
8 cal.ksampling = G
9 cal.kpoints = [10,10,10]
10 cal.cutoff = 650
11 cal.methods = 1
12 cal.smearing = 1
13 cal.sigma = 0.05
14
15 scf.mixType = Broyden
16 scf.mixBeta = 0.4
17 scf.convergence = 1e-6
18 scf.max = 300
19
20 neb.springK = 5
21 neb.images = 6
22 neb.iniFin = true
23 neb.method = QM2
24 neb.convergence = 0.01
25 neb.max = 500
26 neb.freedom = all
27
28 io.wave = false
29 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

```

1 Total number of atoms
2 12
3 Lattice
4 5.00209138 0.00000009 0.00000004
5 0.00000009 5.00209143 -0.00000004
6 0.00000004 -0.00000004 5.07896990
7 Cartesian
8 Hf 2.50104558 2.50104575 0.00000000
9 Hf 0.00000000 0.00000000 0.00000000
10 O 3.75156841 1.25052303 1.47285183
11 O 3.75156857 3.75156869 1.04735062
12 O 1.25052293 1.25052297 3.60611823
13 O 1.25052286 3.75156867 4.03161932
14 O 1.25052287 3.75156860 1.47285187
15 O 1.25052275 1.25052294 1.04735054
16 O 3.75156850 1.25052287 4.03161945
17 O 3.75156850 3.75156869 3.60611821

```

(续下页)

(接上页)

```

18 Zr 2.50104577 0.00000000 2.53948497
19 Zr 0.00000000 2.50104594 2.53948491

```

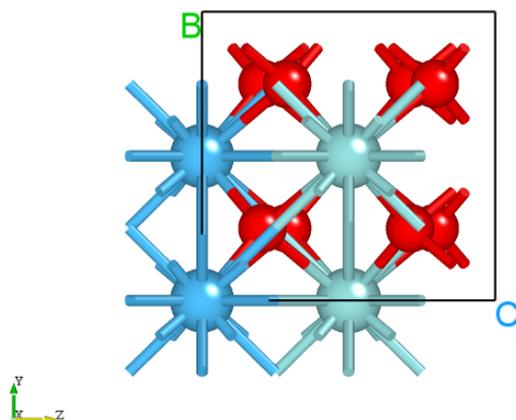
末态结构 *structure07.as* 参考如下

```

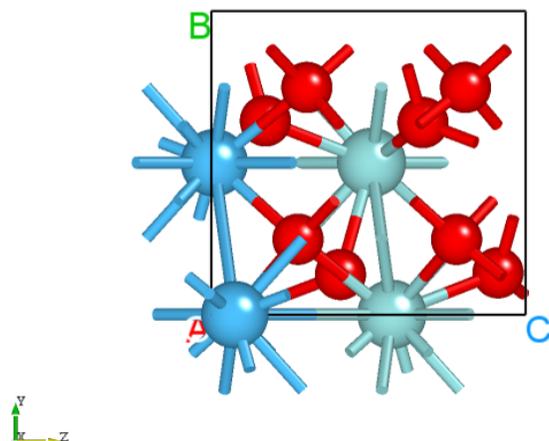
1 Total number of atoms
2 12
3 Lattice
4 4.98221520 -0.00002552 0.00036684
5 -0.00002562 4.99587652 0.00005905
6 0.00039053 0.00006126 5.18258321
7 Cartesian
8 Hf 2.30823006 2.49975412 0.04967381
9 Hf 0.00919001 0.00195723 0.38722458
10 O 4.03365086 0.66419181 2.12958714
11 O 4.00001549 3.18954023 0.89210846
12 O 0.95871628 1.24120307 4.04442128
13 O 0.94984693 3.74053908 4.19050825
14 O 1.35895285 3.73907584 1.57483409
15 O 1.36804279 1.24264997 1.42944278
16 O 3.29999107 0.69159253 4.72728663
17 O 3.26626721 3.16200890 3.48972595
18 Zr 2.31915914 0.00841995 2.97686955
19 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

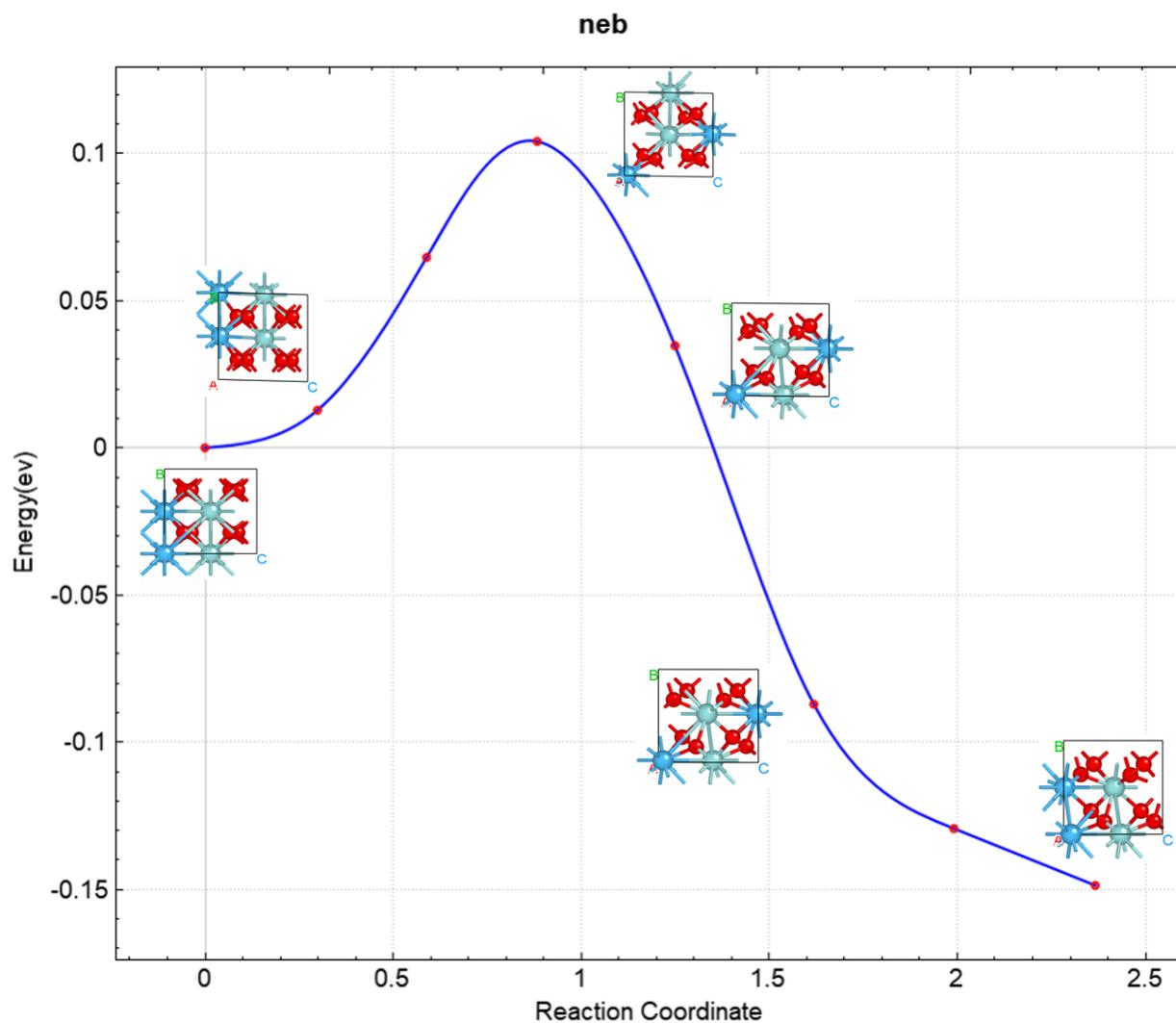
准备好输入文件之后, 将 `ssneb.in` 和包含 `structureNo.as` 文件的多个文件夹上传到服务器上运行, 按照结构弛豫中介绍的方法执行 `DS-PAW ssneb.in`。

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：

```

1 # task type
2 task = scf
3 #system related
4 sys.structure = structure.as
5 sys.symmetry = true

```

(续下页)

```

6 sys.functional = PBE
7 sys.spin = none
8
9 #scf related
10 cal.methods = 1
11 cal.smearing = 3
12 cal.sigma = 0.2
13 cal.ksampling = G
14 cal.kpoints = [1, 1, 1]
15 cal.supGrid = true
16 cal.cutoff = 800
17 scf.convergence = 1.0e-6
18
19 #implicit solvation model
20 sys.sol = true
21 sys.solEpsilon = 80
22 sys.solTAU = 5.25E-4
23
24 #outputs
25 io.charge = false
26 io.wave = false
27 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 $\text{eV}/\text{\AA}^2$, 默认值为 $5.25\text{E-}4$, 该参数建议设置为小于 $1\text{e-}3$ 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

```

1 Total number of atoms
2 3
3 Lattice
4 10.00000000 0.00000000 0.00000000
5 0.00000000 10.00000000 0.00000000
6 0.00000000 0.00000000 10.00000000
7 Cartesian
8 H 5.63934499 4.89541998 4.58224001
9 H 4.36065501 5.11499002 5.45934003
10 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* : 使用隐式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:

$$\text{Solvation Energy} = E(\text{sys.sol}=\text{true}) - E(\text{sys.sol}=\text{false})$$

根据该公式可知, 需另进行 $\text{sys.sol} = \text{false}$ 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。

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:

```

1 # task type
2 task = scf
3
4 sys.functional = PBE
5 sys.structure = structure.as

```

(续下页)

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.

```

6
7 cal.ksampling = G
8 cal.cutoff = 650
9 cal.sigma = 0.2
10 cal.smearing = 3
11 cal.kpoints = [7,7,1]
12
13 scf.convergence = 1.0e-6
14 scf.max = 200
15
16 sys.sol = true
17 sys.solEpsilon = 78.4
18 sys.solLambdaD = 3.04
19 sys.solTAU = 0
20
21 # Potential fixed
22 sys.fixedP = true
23 sys.fixedPPotential = 2.155
24
25 io.charge = true
26 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 $\text{eV}/\text{\AA}^2$, 默认值为 $5.25\text{E-}4$, 该参数建议设置为小于 $1\text{e-}3$ 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 $\lambda_D = \sqrt{\frac{\epsilon\epsilon_0 k_B T}{2c^0 z^2 q^2}}$

1M 带 +/-1 电荷阴阳离子的水溶液的德拜长度为: 3.04 \AA

structure.as 文件参考如下:

```

1 Total number of atoms
2 8
3 Lattice
4 3.63404989 0.00000000 0.00000000
5 0.00000000 3.63404989 0.00000000
6 0.00000000 0.00000000 23.62132454
7 Cartesian
8 Cu 0.00000000 0.00000000 1.81702310

```

(接上页)

```

9 Cu 1.81702495 0.00000000 3.63404620
10 Cu 1.81702495 1.81702495 1.81702310
11 Cu 0.00000000 1.81702495 3.63404620
12 Cu 0.00000000 0.00000000 5.46390548
13 Cu 1.81702495 0.00000000 7.22885308
14 Cu 1.81702495 1.81702495 5.46390548
15 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：

```

1 ## FINISHED FIXEDPOTENTIAL LOOP 5 ##
2 Electron : 149.993000
3 ElectrodePotential_SHE : 2.157747 V
4 ElectrodePotential_PZC : 2.484286 V
5 ElectrodePotential_SHE(PZC) : -0.326539 V
6 Chemical Potential(electron) : -6.757747 eV
7 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：

```

1 # task type
2 task = wannier
3 sys.structure = structure.as
4 sys.symmetry = false
5 sys.functional = PBE
6 sys.spin = none
7 cal.methods = 1
8 cal.smearing = 1
9 cal.ksampling = G
10 cal.kpoints = [16,16,16]
11 cal.totalBands = 12
12
13 #wannier related
14 wannier.functions = 12
15 wannier.wannMaxIter = 20000
16 wannier.outStep = 50
17
18 #interpolated band related
19 wannier.interpolatedBand = true
20 wannier.kpointsLabel= [G,X,W,K,G,L]
21 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]
22 wannier.kpointsNumber = [100]
23
24 io.charge = true
25 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:

```

1 Total number of atoms
2 2
3 Lattice
4 0.00 2.75 2.75
5 2.75 0.00 2.75
6 2.75 2.75 0.00
7 Direct
8 Si -0.125000000 -0.125000000 -0.125000000
9 Si 0.125000000 0.125000000 0.125000000

```

i 备注

1. 初始投影的设置 `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:

```

1 Total number of atoms
2 2
3 Lattice
4 0.00 2.75 2.75
5 2.75 0.00 2.75
6 2.75 2.75 0.00
7 Direct WannProj
8 Si -0.125000000 -0.125000000 -0.125000000 [s,p,sp3-1,sp3-2]
9 Si 0.125000000 0.125000000 0.125000000 [s,p,sp3-3,sp3-4]

```

i 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, 在对应坐标后写入口即可。
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`。

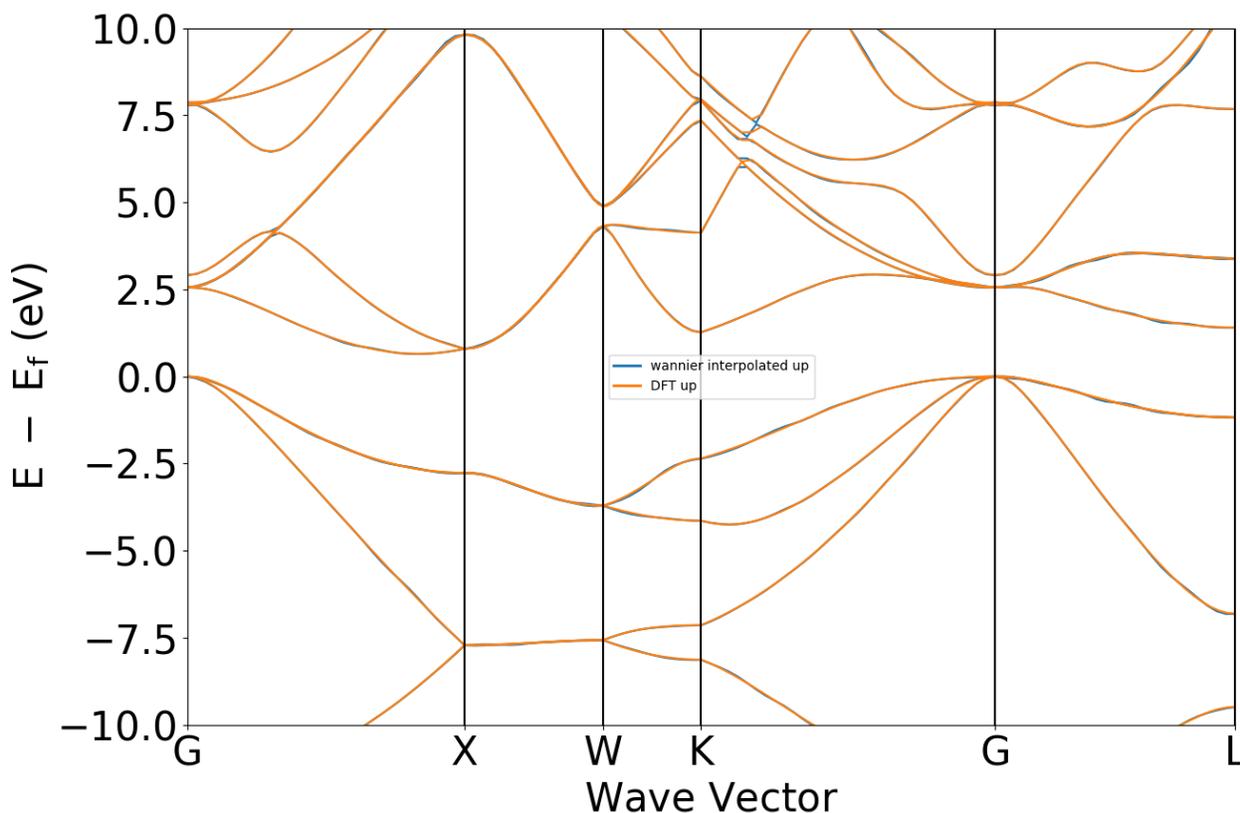
2.30.3 analysis 计算结果分析

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

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
```

(续下页)

(接上页)

```

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：

```

▼ object {6}
  ▶ AtomInfo {5}
  ▶ Eigenvalue {3}
  ▶ Energy {3}
  ▶ Force {1}
  ▼ MagInfo {1}
    ▼ TotalMag [1]
      0 : 2.000996884905
  ▶ Stress {2}

```

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 2 μ B，同时在 *DS-PAW.log* 中也可读取体系的总磁矩为 2 μ B。

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
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-8 μ B**，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：

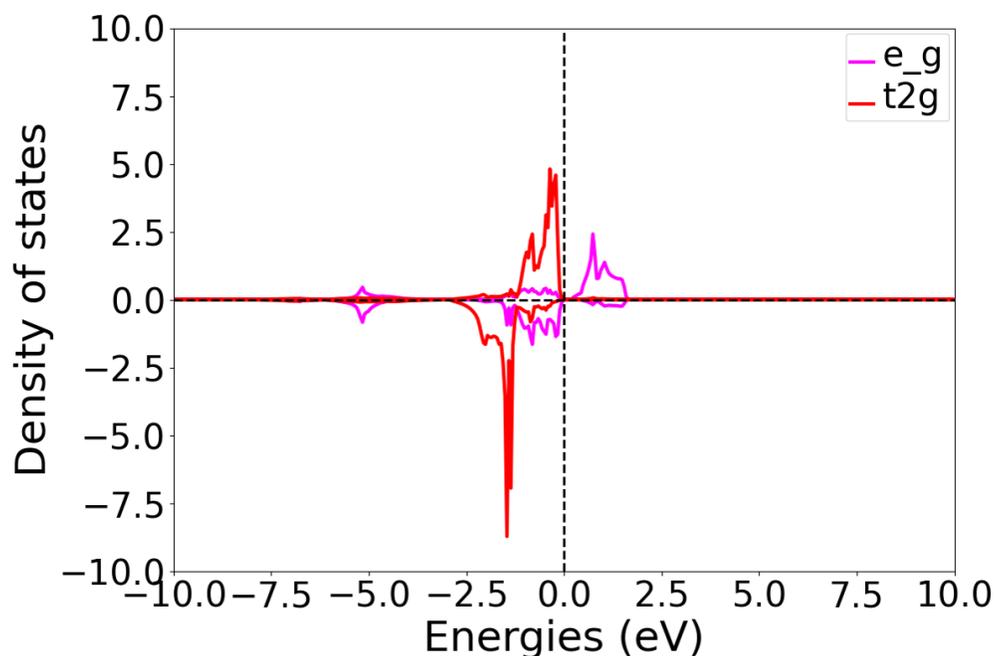
- *dos.range*：表示能量计算区间为 -20 到 20 eV；
- *dos.resolution*：表示在能量计算区间内打点的间隔精度；
- *dos.project*：控制态密度的投影计算，本次计算打开了态密度的投影。

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 t_{2g} 和 e_g 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

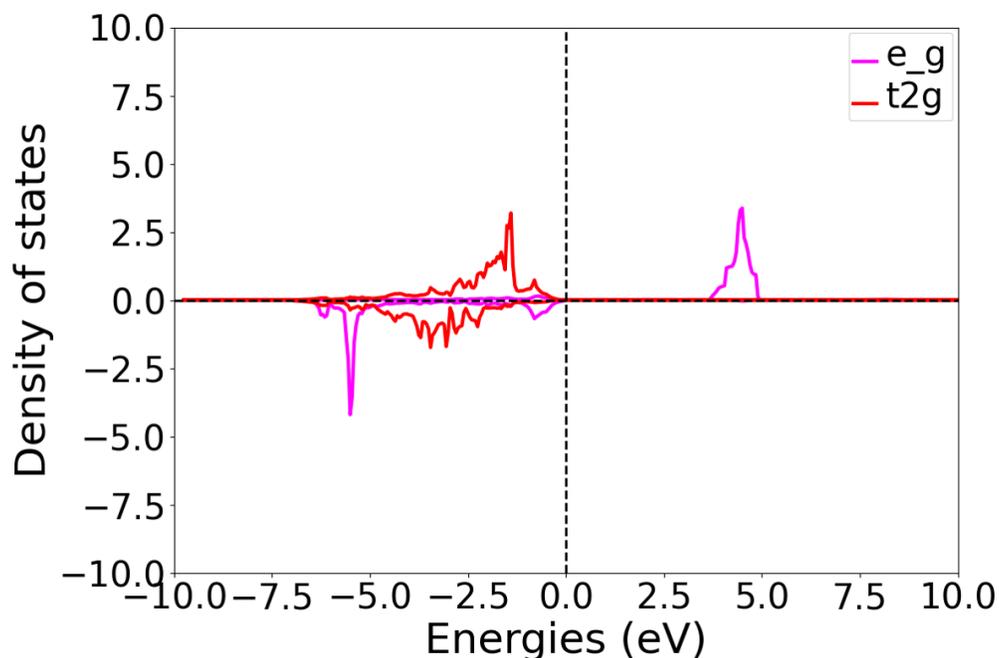
NiO 体系 DFT+U 的态密度计算的计算流程与上节 NiO 体系的态密度计算流程一致，不同之处在于需在自洽和态密度的计算中都加入 DFT+U 相关参数，需增加的输入参数如下所示：

```
#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
```

(续下页)

(接上页)

```

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。

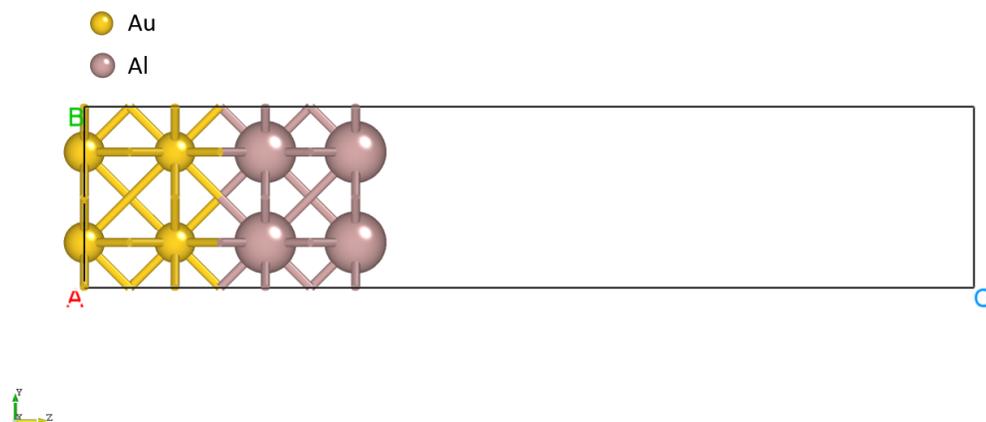
`structure.as` 文件参考如下：

```

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：



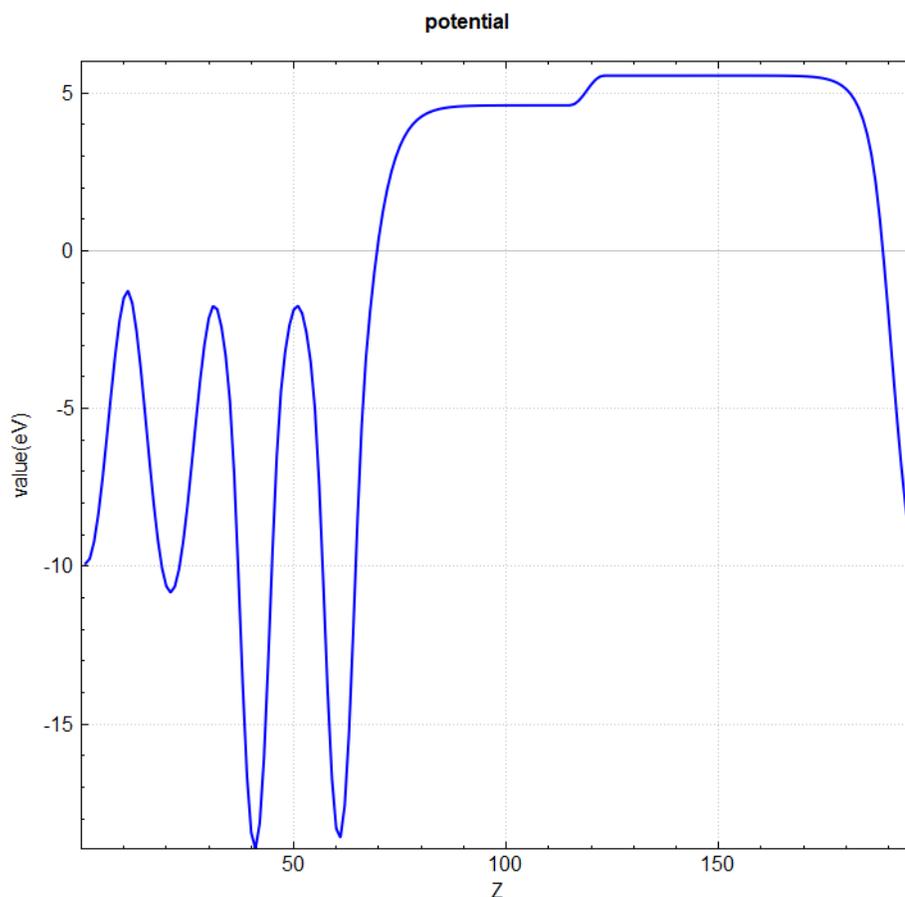
3.3.2 run 程序运行

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。

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

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)。

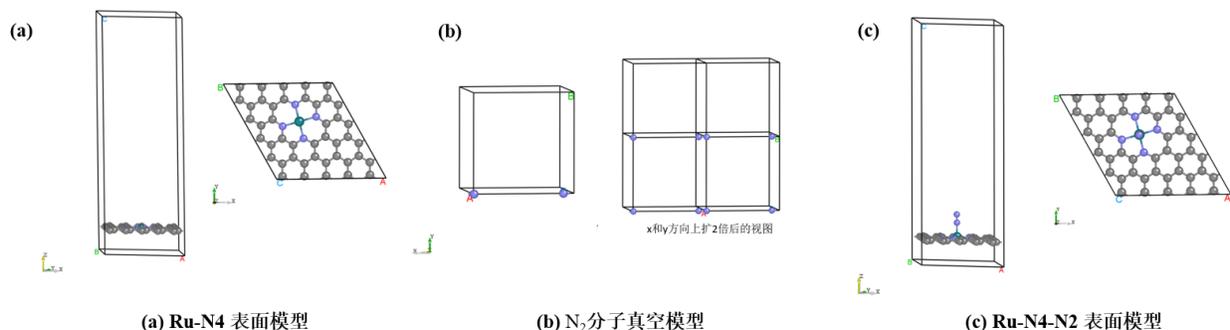
因恒电荷模型并不适于处理电化学界面问题，我们可采用在电子巨正则系综下展开第一性原理计算，这种计算方法又被称为固定电势方法 (fixed potential method/constant potential method)。在此例中我们将利用固定电势计算的模型称为恒电势模型 (constant potential model, CPM)。

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.ksampling = 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
↪
```

(续下页)

(接上页)

```
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：



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 ($\sigma \rightarrow 0$)，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：



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

其中，E0 对应水溶液浸润的模型下体系的总能 ($\sigma \rightarrow 0$)，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。

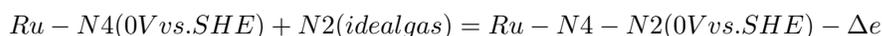
CPM_water

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($\sigma \rightarrow 0$): 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：



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

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 E_0 应当用 Ω_0 来代替。吸附反应式可写作：

$$(Ru - N4)(0V vs. SHE) + N2(ideal\ gas) = (Ru - N4 - N2)(0V vs. SHE)$$

$$\Delta E = \Omega_0(Ru - N4 - N2) - \Omega_0(Ru - N4) - \Omega_0(N2)$$

其中， Ω_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	E_0	nE_0	n_e	dne	μ_e	Ω_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：

$$Ru - N4(0V vs. SHE) + N2(ideal\ gas) = Ru - N4 - N2(0V vs. SHE) - \Delta e$$

$$\Delta E = E_0(Ru - N4 - N2) - \Delta e * \mu_e - E_0(Ru - N4) - E_0(N2)$$

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

$$= -1.4042eV$$

方法二考虑电子巨正则系综下的体系总能 Ω_0 ，反应能计算过程如下：

$$(Ru - N4)(0V vs. SHE) + N2(ideal\ gas) = (Ru - N4 - N2)(0V vs. SHE)$$

$$\Delta E = \Omega_0(Ru - N4 - N2) - \Omega_0(Ru - N4) - \Omega_0(N2)$$

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

$$= -1.4027eV$$

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 Ω_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：

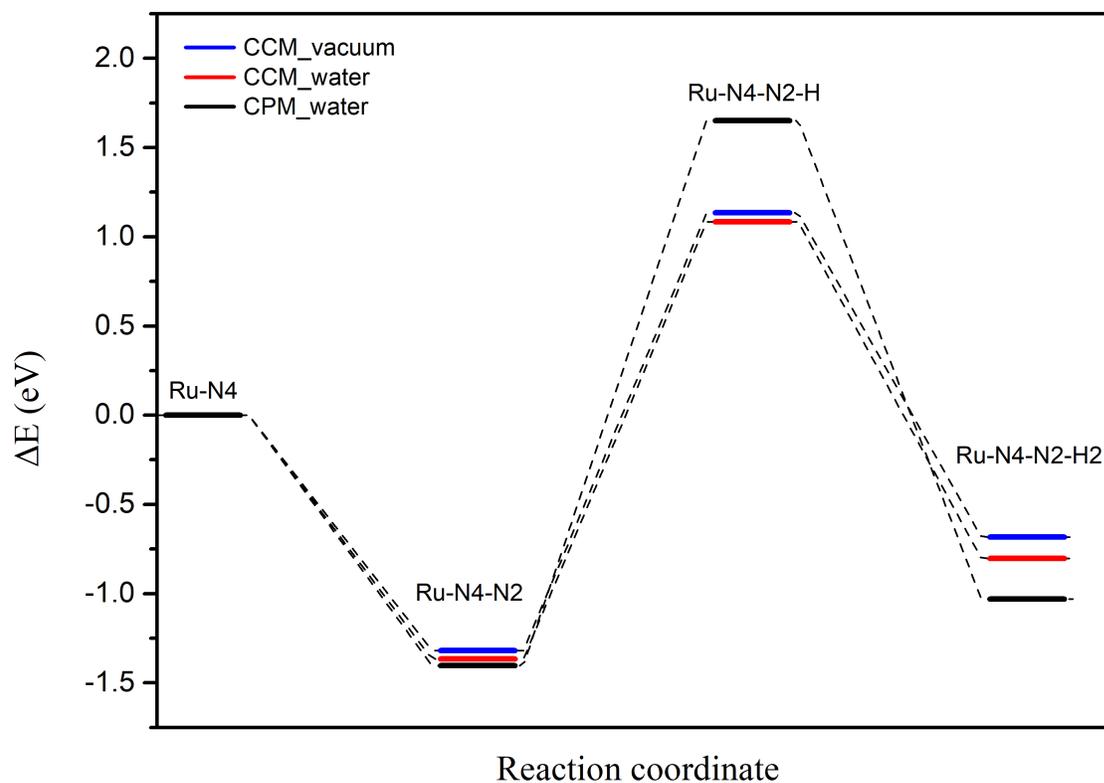


图 3. 反应坐标-反应能曲线

3.5 ref 参考文献

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:

LDA

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

PBE

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

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。

更新的 1.1 版本赝势评估的结果正在整理中，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）。

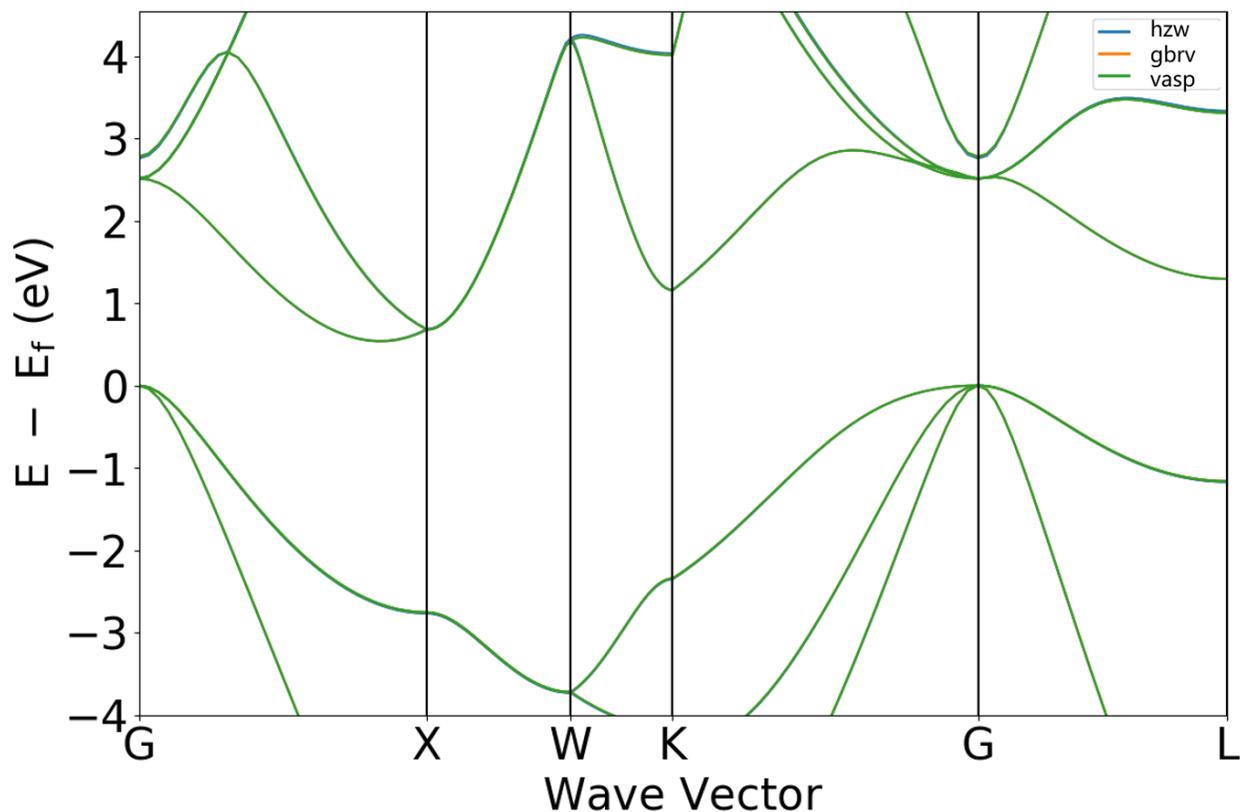
4.3 gbrv 赝势

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

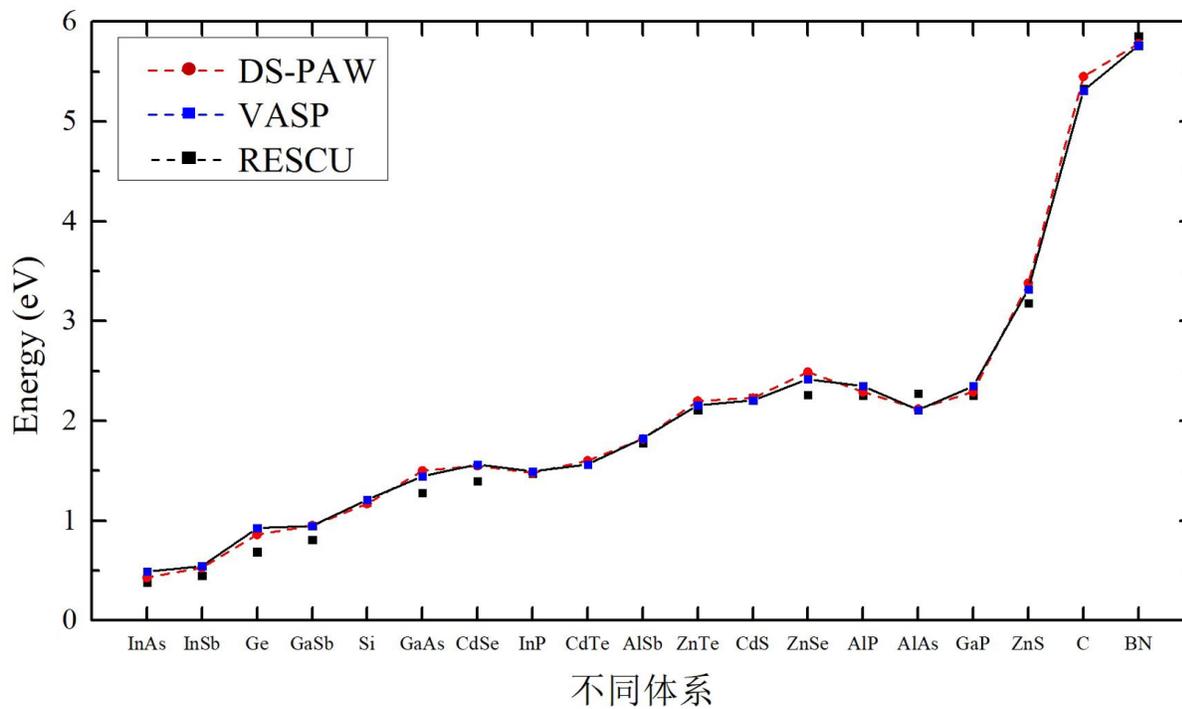
为对比三种赝势的计算结果，本节以 Si 体系为例，分别使用以上三种赝势进行了能带计算，下图为能带对比图，该对比图说明三种赝势在描述 Si 的能带时，体系出较高的一致性，验证了 hzw 赝势的准确性。



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>

5.1 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.ksampling*
 - *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.qsampling*
 - *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 $\text{eV}/\text{\AA}^2$, 该参数建议设置为小于 $1\text{e-}3$ 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 \AA , 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`

i 备注

1. 关于德拜长度 `sys.solLambdaD`, 其表达式为 $\lambda_D = \sqrt{\frac{\epsilon\epsilon_0 k_B T}{2c^0 z^2 q^2}}$

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览；

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.ksampling`

默认值: MP

可选值: MP/G

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

案例: `cal.ksampling = 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=true* , 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=true`, 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=true`, `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-Kronig 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*, *potential.h5* 文件写入静电势 (离子势和 *hartree* 势之和), 当 *potential.type = total*, *potential.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*, *potential.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 $\text{eV}/\text{\AA}$, 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: $n \times 1$ 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: $n \times 1$ 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: $n \times 1$ 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.qsampling*

默认值: MP

可选值: MP/G

描述: `phonon.qsampling` 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.qsampling = 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 matrix 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 $E_f=0$;

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:

```

1 Total number of atoms
2 2
3 Lattice
4 0.00 2.75 2.75
5 2.75 0.00 2.75
6 2.75 2.75 0.00
7 Direct WannProj
8 Si -0.125000000 -0.125000000 -0.125000000 [s,p]
9 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]

i 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。

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。

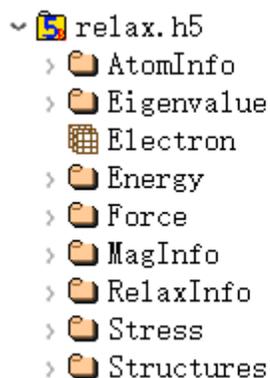
除电荷密度文件 *rho.h5*、溶剂束缚电荷密度输出文件 *rhoBound.h5*、其余输出文件的文件名取决于 **task** 类型，现 DS-PAW 支持 task 类型有 14 种，对应的 h5 文件名分别为：*relax.h5*、*scf.h5*、*band.h5*、*dos.h5*、*potential.h5*、*elf.h5*、*pcharge.h5*、*frequency.h5*、*elastic.h5*、*neb.h5*、*phonon.h5*、*aimd.h5*、*epsilon.h5*、*wannier.h5*。

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。

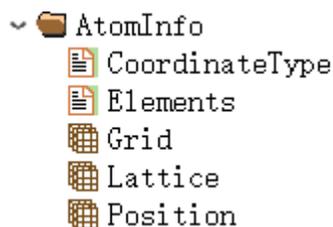
6.1 relax.h5

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。

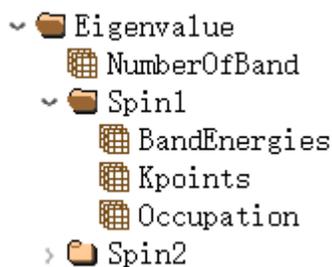
relax.h5 至少包含 9 个基本结构体：



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

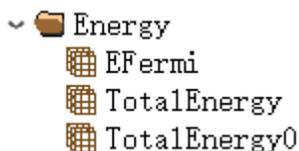


(2) *Eigenvalue* 中保存能带计算数量、自旋信息、k 点数量及坐标、各能带在各 k 点下对应的轨道占据数及能量本征值；

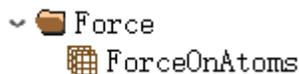


(3) *Electron* 中保存体系总的价电子数；

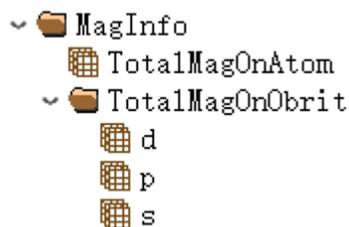
(4) *Energy* 中保存总能及费米能；



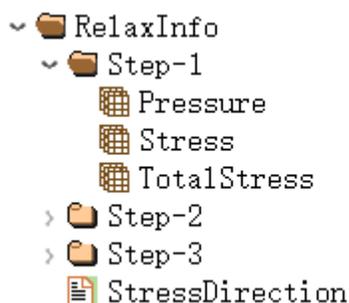
(5) *Force* 中保存弛豫过程每个原子的受力；



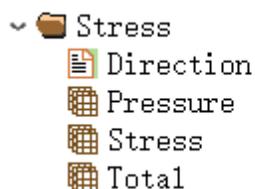
(6) *MagInfo* 中保存原子总的磁矩信息，若打开了投影则保存投影磁矩信息；



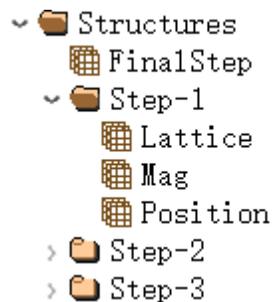
(7) *RelaxInfo* 中保存体系在结构弛豫过程中每步的应力及压力数据；



(8) *Stress* 中保存晶胞各方向的应力大小，体系压力大小；



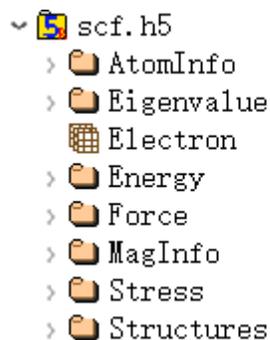
(9) *Structures* 中保存弛豫过程中结构和磁矩数据；



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:

```
~ scf.h5
  > AtomInfo
  > Eigenvalue
  > Electron
  > Energy
  > Force
  > MagInfo
  > OpticalInfo
  > Stress
  > Structures

~ OpticalInfo
  > AbsorptionCoefficient
  > EnergyAxe
  > EnergyLoss
  > ExtinctionCoefficient
  > ImDielectricFunction
  > OpticalConductivity
  > ReDielectricFunction
  > Reflectance
  > RefractiveIndex

~ WaveDerivative
  > DerivativeIndex
  > DerivativeValue
```

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

```
~ scf.h5
  > AtomInfo
  > BaderInfo
  > Eigenvalue
  > Electron
  > Energy
  > Force
  > MagInfo
  > Stress
  > Structures
```

```

  ~ BaderInfo
    ~ AtomicVolume
    ~ Charge
    ~ MinDistance

```

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

```

  ~ scf.h5
    > AtomInfo
    > Eigenvalue
    ~ Electron
    > Energy
    > Force
    > MagInfo
    > PolarizationInfo
    > Stress
    > Structures

  ~ PolarizationInfo
    ~ Electronic
    ~ Ionic
    ~ Quantum
    ~ Total

```

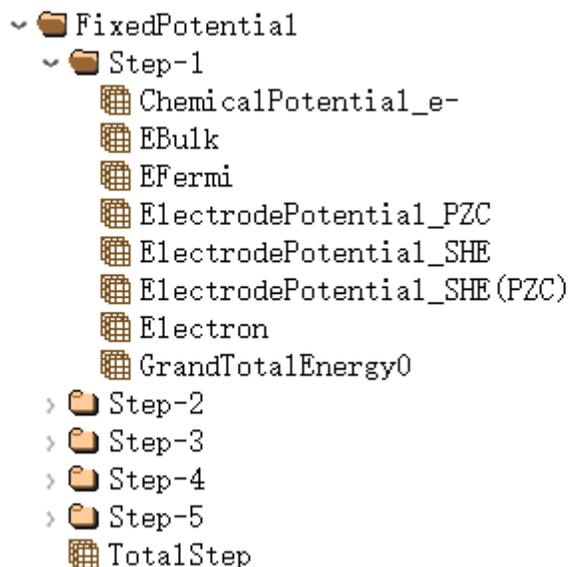
(4) 通过设置 `sys.fixedP = true` 在自洽计算中进行固定电势计算：

```

  ~ scf.h5
    > AtomInfo
    ~ ChemicalPotential_e-
    ~ Converged
    ~ EBulk
    ~ EFermi
    > Eigenvalue
    ~ ElectrodePotential_PZC
    ~ ElectrodePotential_SHE
    ~ ElectrodePotential_SHE(PZC)
    ~ Electron
    > Energy
    > FixedPotential
    > Force
    ~ GrandTotalEnergy0
    > Stress
    > Structures

```

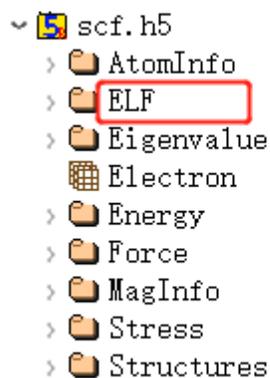
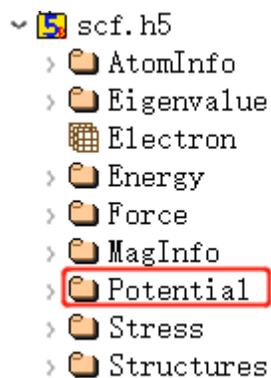
其中 $ChemicalPotential_e$ 为体系电子化学势值； $EBulk$ 取值为“隐式溶剂模型下费米能级 shift 值 * -1”； $ElectrodePotential$ 给出不同标定标准下的电势值； $GrandTotalEnergy0$ 给出电子巨正则系综下的体系总能。



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：

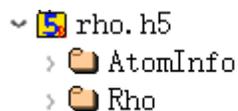




6.3 rho.h5

rho.h5 为各 *task* 下的电荷密度输出文件。

rho.h5 包含 2 个结构体：



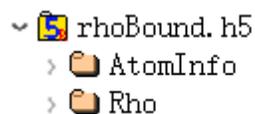
其中 *AtomInfo* 与 *relax.h5* 文件的 *AtomInfo* 结构一致，*Rho* 中保存电荷密度数据：



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：

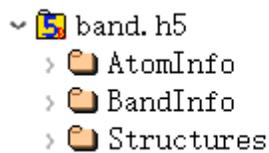


其中 *AtomInfo* 与 *relax.h5* 文件的 *AtomInfo* 结构基本一致，*Rho* 中保存溶剂束缚电荷密度数据：

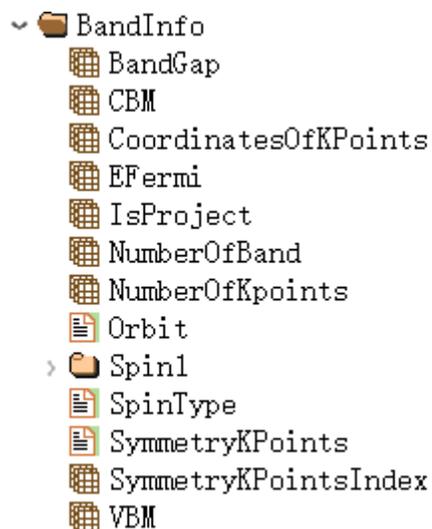
6.5 band.h5

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。

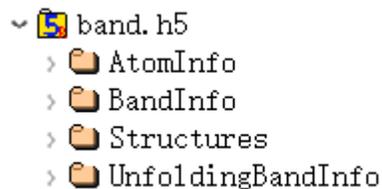
band.h5 至少包含 3 个结构体：



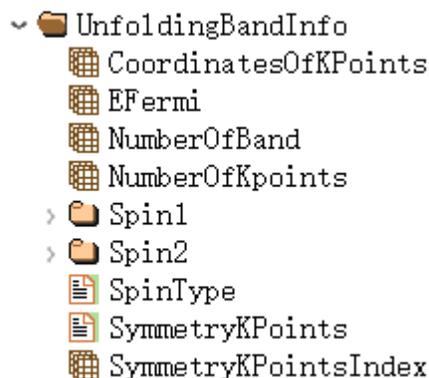
其中 *AtomInfo*、*Structures* 与 *relax.h5* 文件相对应的结构体结构一致，*BandInfo* 中保存能带数据：



能带去折叠计算对应的 *band.h5* 至少包含 4 个结构体：



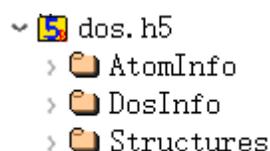
其中 *AtomInfo*、*Structures* 与 *relax.h5* 文件相对应的结构体结构一致，*BandInfo* 中保存能带数据，*UnfoldingBandInfo* 中保存能带反折叠数据：



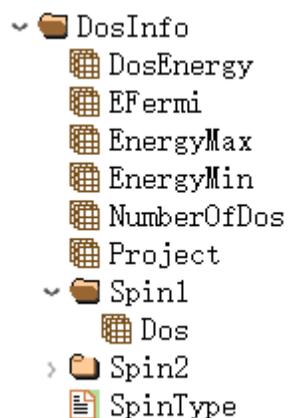
6.6 dos.h5

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：

- ~ potential.h5
 - > AtomInfo
 - > Potential
 - > Structures

其中 *Potential* 中保存势函数数据:

- ~ Potential
 - ▣ SpinElectrostaticPotential
 - ▣ SpinLocalPotential
 - ▣ TotalElectrostaticPotential
 - ▣ TotalLocalPotential

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:

- ~ elf.h5
 - > AtomInfo
 - > ELF
 - > Structures

其中 *elf* 中保存局域密度数据:

- ~ ELF
 - ▣ SpinELF
 - ▣ TotalELF

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:

- ~ pcharge.h5
 - > AtomInfo
 - > Pcharge

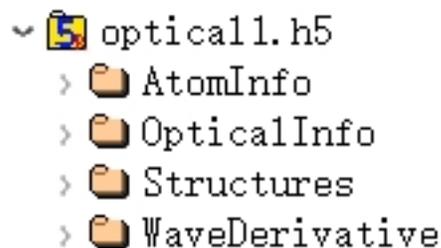
其中 *Pcharge* 中保存部分电荷密度数据:

- ~ Pcharge
 - > 1
 - > 2
 - ▣ SumK

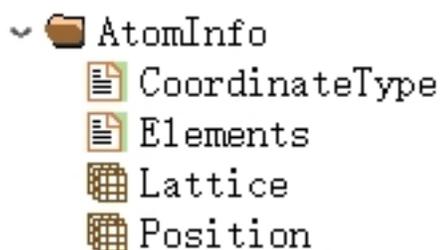
6.10 optical.h5

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。

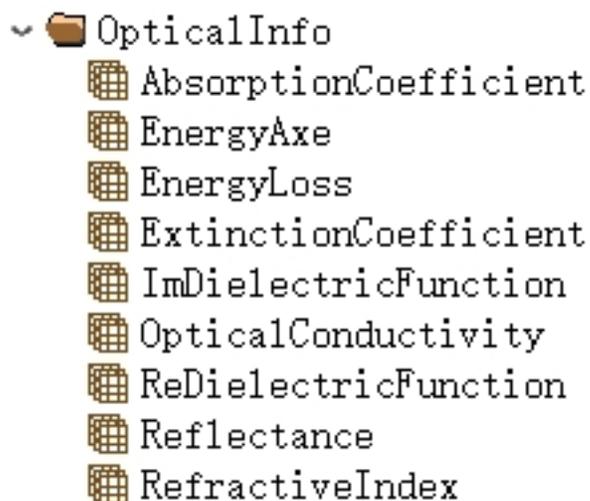
optical.h5 包含 4 个结构体：



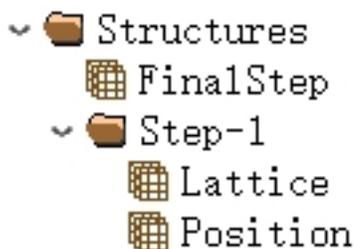
(1) 其中 *AtomInfo* 中保存体系的基本结构信息，如晶胞大小，原子位置等：



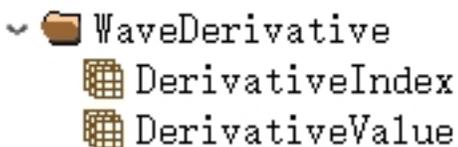
(2) *opticalInfo* 中保存光学计算各种性质的数据信息：



(3) *Structures* 中保存光学计算的结构信息：



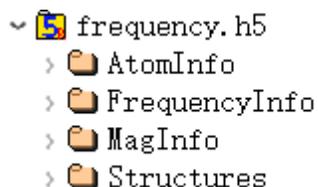
(4) *WaveDerivate* 中保存波函数对 k 点的导数数组，大小为：(实部，虚部) *NumberOfBand* 优选后 band 数目 *NumberOfKPoints* *NumberOfSpin* (x,y,z); *DerivativeIndex* 给出导数数组的维度; *DerivativeValue* 给出导数数组的值。



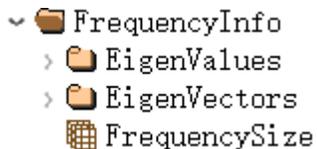
6.11 frequency.h5

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：

```

  ~ elastic.h5
    > AtomInfo
    > ElasticInfo
    > MagInfo
    > Structures

```

其中 *ElasticInfo* 中保存弹性数据:

```

  ~ ElasticInfo
    ElasticModulus
    ~ Hill
      BulkModulus
      PoissonRatio
      ShearModulus
      YoungModulus
    > Reuss
    > Voigt

```

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:

```

  ~ neb.h5
    > BarrierInfo
    IniFin
    > LoopInfo
    > RelaxedStructure
    > UnrelaxStructure

```

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:

```

  ~ BarrierInfo
    MaxForce
    ReactionCoordinate
    Tangent
    TotalEnergy

```

其中 *IniFin* 中保存计算初末态的开关:

```

  IniFin

```

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:

- ~ LoopInfo
 - ~ 01
 - MaxForce
 - TotalEnergy
 - > 02
 - > 03
 - > 04
 - > 05
 - > 06
 - Keys

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

- ~ RelaxedStructure
 - ~ Image00
 - CoordinateType
 - Elements
 - Lattice
 - Position
 - > Image01
 - > Image02
 - > Image03
 - > Image04
 - > Image05
 - > Image06
 - > Image07

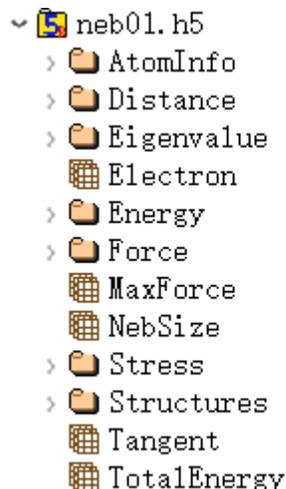
其中 *UnrelaxStructure* 中保存各 image 优化之前的结构数据:

- ~ UnrelaxStructure
 - ~ Image00
 - CoordinateType
 - Elements
 - Lattice
 - Position
 - > Image01
 - > Image02
 - > Image03
 - > Image04
 - > Image05
 - > Image06
 - > Image07

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：

```

  ~ 📁 phonon.h5
    > 📁 BandInfo
    > 📁 DosInfo
    > 📁 EpsilonInfo
    > 📁 ForceConstant
    > 📁 PrimitiveAtomInfo
    > 📁 SupercellAtomInfo
    > 📁 ThermalInfo
    > 📁 UnitAtomInfo
    > 📁 phonon

```

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：

```

  ~ 📁 EpsilonInfo
    📁 BornEffectiveCharge
    ~ 📁 Epsilon
      📁 Electronic
      📁 Ionic
      📁 Total
    ~ 📁 Piezoelectric
      📁 Electronic
      📁 Ionic
      📁 Total

```

其中 *ForceConstant* 中保存力学常数数据：

```

  ~ 📁 ForceConstant
    📁 ConstantIndex
    📁 ConstantValue

```

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：

```

  ~ 📁 UnitAtomInfo
    📄 CoordinateType
    📄 Elements
    📁 Lattice
    📁 Position

```

其中 *ThermalInfo* 中保存声子热力学数据：

```

  ~ ThermalInfo
    ~ Entropy
    ~ HeatCapacity
    ~ HelmholtzFreeEnergy
    ~ Temperatures

```

其中 *Phonon* 中保存声子计算输入参数数据:

```

  ~ phonon
    ~ dfptEpsilon
    ~ dosRange
    ~ dosResolution
    ~ dosSigma
    ~ eigenVectors
    ~ fdDisplacement
    ~ isDisplacement
    ~ method
    ~ nac
    ~ primitiveUvw
    ~ qpoints
    ~ qpointsCoord
    ~ qpointsLabel
    ~ qpointsNumber
    ~ qsampling
    ~ structureSize
    ~ thermal
    ~ thermalRange
    ~ type

```

(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:

```

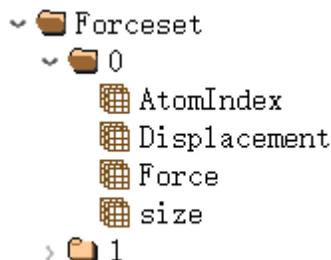
  ~ phonon.h5
    > BandInfo
    > Displacements
    > DosInfo
    > ForceConstant
    > Forceset
    > PrimitiveAtomInfo
    > SupercellAtomInfo
    > UnitAtomInfo
    > phonon

```

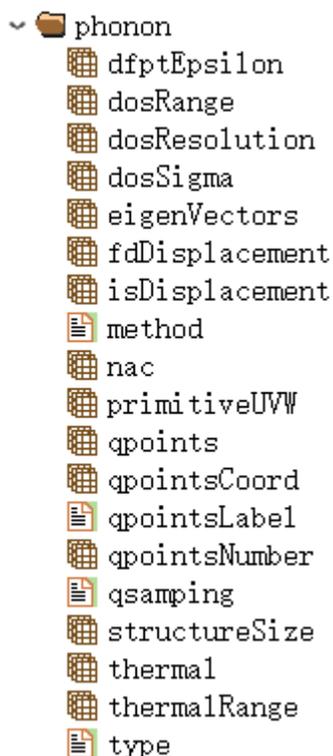
其中 *BandInfo*、*DosInfo* 分别保存能态和态密度数据, 其结构分别与 `band.h5`、`dos.h5` 文件相对应的结构体结构一致;

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` 子文件夹下会输出文件 `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:

```

  v 5 phonon001.h5
    > AtomInfo
    > Eigenvalue
    > Energy
    > Force
    > MagInfo
    > Stress
    > Structures

```

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：

```

  v 5 aimd.h5
    > AimdInfo
    > AtomInfo
    > Eigenvalue
    > Electron
    > Energy
    > Force
    > MagInfo
    > Stress
    > Structures

```

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：

```

  v 5 AimdInfo
    v 5 Step-1
      IonsKineticEnergy
      Pressure
      Stress
      Temperature
      TotalEnergy
      TotalEnergy0
      TotalStress
    > Step-2
    > Step-3

```

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：

```
└─ epsilon.h5
  ├─ AtomInfo
  ├─ EpsilonInfo
  ├─ MagInfo
  └─ Structures
```

其中 *EpsilonInfo* 中保存介电常数数据：

```
└─ EpsilonInfo
  ├─ BornEffectiveCharge
  └─ Epsilon
    ├─ Electronic
    ├─ Ionic
    └─ Total
  └─ Piezoelectric
    ├─ Electronic
    ├─ Ionic
    └─ Total
```

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：

```
└─ wannier.h5
  ├─ AtomInfo
  ├─ Eigenvalue
  ├─ Electron
  ├─ Energy
  ├─ Force
  ├─ Stress
  ├─ Structures
  ├─ WannBandInfo
  └─ WannierInfo
```

其中 *WannBandInfo* 中保存插值能带数据，可用于绘制能带图：

- ~ WannBandInfo
 - CoordinatesOfKPoints
 - EFermi
 - NumberOfBand
 - NumberOfKpoints
 - ~ Spinl
 - BandEnergies
 - BandProject
 - SpinType
 - SymmetryKPoints
 - SymmetryKPointsIndex

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：

- ~ WannierInfo
 - > BlochBandsKpoints
 - > Projections
 - > Spinl

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

- ~ Spinl
 - HaveDisentangled
 - Lattice
 - NumberOfBlochBands
 - NumberOfDFTBands
 - NumberOfWannierFunctions
 - OverlapMatrix
 - RecipLattice
 - RotationMatrix
 - ~ Spreads
 - Invariants
 - Total
 - WannierFunctionCenters
 - ~ WannierSpaceHamiltonian
 - ImaginaryHamiltonian
 - NumberOfRpoints
 - RealHamiltonian
 - RpointsDegeneracy
 - RpointsVector

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。

```
.
├── 00
│   ├── 00.tar.xz
│   ├── latestStructure00.as
│   └── structure00.as
├── 01
│   ├── 01.tar.xz
│   ├── latestStructure01.as
│   └── structure01.as
├── 02
│   ├── 02.tar.xz
│   ├── latestStructure02.as
│   └── structure02.as
├── 03
│   ├── 03.tar.xz
│   ├── latestStructure03.as
│   └── structure03.as
├── 04
│   ├── 04.tar.xz
│   ├── latestStructure04.as
│   └── structure04.as
├── DS-PAW.log
└── neb.tar.xz
```

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，各子文件夹下的压缩包为初次 *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。

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 `dspawpy` 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):

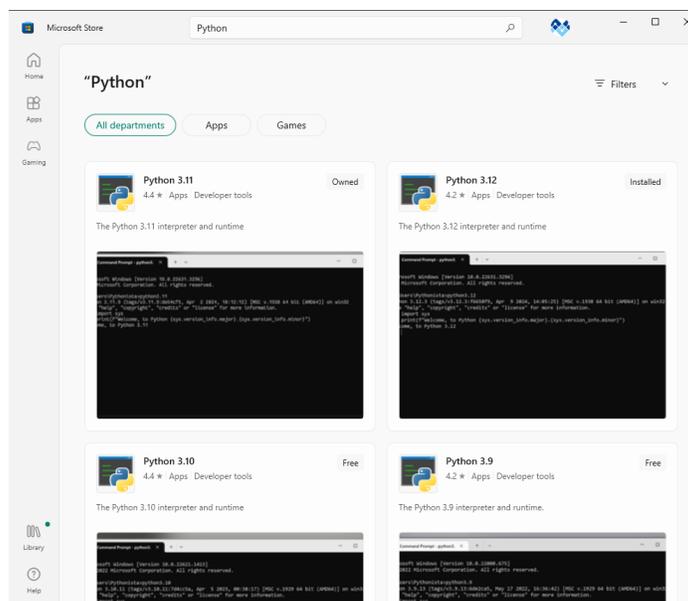
- 使用 `mamba` 或 `conda`, 安装包可前往 <https://conda-forge.org/download/> 下载

```
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 8-12 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
```

i 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` **pypi package 1.7.3** 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 `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)
```

从 `infile` 中读取结构信息，完成格式转化后写入 `outfile`

- 多构型 -> 单构型，仅写入最后一个离子步信息
- 晶体结构 -> 分子结构，将丢失晶胞信息
- 分子结构 -> 晶体结构，将添加一个 2 倍最大原子 xyz 坐标的晶胞
- `pdb` 和 `dump` 格式可能存在浮点数精度损失

参数

- **infile** --
 - `h5/json/as/hzw/cif/poscar/cssr/xsf/mcsqs/prismatic/yaml/fleur-inp` 文件路径；
 - 若给定文件夹路径，可配合 `task` 参数读取内部的 `{task}.h5/json` 文件
 - 若给定字符串列表，将依次读取数据并合并成一个 `Structures` 列表
- **si** (`int`, `list` or `str`) --
 - 构型编号，从 1 开始
 - * `si=1`, 读取第一个构型
 - * `si=[1,2]`, 读取第一个和第二个构型
 - * `si=':'`, 读取所有构型
 - * `si='-3:'`, 读取最后三个构型

- 若为空，多构型文件将读取所有构型，单构型文件将读取最新构型
- 此参数仅对 h5/json 文件有效
- **ele** --
 - 元素符号，写法参考：'H' 或 [H;'O']
 - 若为空，将读取所有元素的原子信息
 - 此参数仅对 h5/json 文件有效
- **ai** --
 - 原子编号，从 1 开始
 - 用法同 si
 - 若为空，将读取所有原子信息
 - 此参数仅对 h5/json 文件有效
- **infmt** --
 - 输入结构文件类型，例如'h5'，如果为 None，则根据文件后规则判断
- **task** --
 - 用于当 datafile 为文件夹路径时，寻找内部的 {task}.h5/json 文件。
 - 计算任务类型，包括'scf', 'relax', 'neb', 'aimd' 四种，其他值将被忽略。
- **outfile** --
 - 输出文件名
- **outfmt** --
 - 输出结构文件类型，例如'xyz'，如果为 None，则根据文件后规则判断
- **coords_are_cartesian** --
 - 是否写作笛卡尔坐标，默认为 True；否则写成分数坐标形式
 - 此选项暂时仅对 as 和 json 格式有效

示例

```
>>> from dspawpy.io.structure import convert
>>> convert('dspawpy_proj/dspawpy_tests/inputs/supplement/PtH.as', outfile='dspawpy_proj/
↳ dspawpy_tests/outputs/doctest/PtH.hzw')
==> ...PtH.hzw...
```

格式转换批量测试

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

(续下页)

```

==> ...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'*)

读取一/多个 h5/json 文件，返回 pymatgen 的 Structures 列表

参数

- **datafile** --
 - h5/json/as/hzw/cif/poscar/cssr/xsf/mcsqs/prismatic/yaml/fleur-inpgen 文件路径;
 - 若给定文件夹路径，可配合 `task` 参数读取内部的 {task}.h5/json 文件
 - 若给定字符串列表，将依次读取数据并合并成一个 Structures 列表
- **si** (*int, list or str*) --
 - 构型编号，从 1 开始
 - * `si=1`, 读取第一个构型
 - * `si=[1,2]`, 读取第一个和第二个构型
 - * `si=':'`, 读取所有构型
 - * `si='-3:'`, 读取最后三个构型
 - 若为空，多构型文件将读取所有构型，单构型文件将读取最新构型
 - 此参数仅对 h5/json 文件有效
- **ele** --
 - 元素符号，写法参考: 'H' 或 ['H','O']
 - 若为空，将读取所有元素的原子信息

- 此参数仅对 h5/json 文件有效
- **ai** --
 - 原子编号, 从 1 开始
 - 用法同 si
 - 若为空, 将读取所有原子信息
 - 此参数仅对 h5/json 文件有效
- **fmt** --
 - 文件格式, 包括 'as', 'hzw', 'xyz', 'pdb', 'h5', 'json' 6 种, 其他值将被忽略。
 - 若为空, 文件类型将依据文件名称惯例判断。
- **task** --
 - 用于当 datafile 为文件夹路径时, 寻找内部的 {task}.h5/json 文件。
 - 计算任务类型, 包括 'scf', 'relax', 'neb', 'aimd' 四种, 其他值将被忽略。

返回

结构列表

返回类型

pymatgen_Structures

示例

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

读取单个文件生成 Structures 列表

```
>>> pymatgen_Structures = read(datafile='dspawpy_proj/dspawpy_tests/inputs/supplement/PtH.
↳as')
>>> len(pymatgen_Structures)
1
>>> pymatgen_Structures = read(datafile='dspawpy_proj/dspawpy_tests/inputs/supplement/PtH.
↳hzw')
>>> len(pymatgen_Structures)
1
>>> pymatgen_Structures = read(datafile='dspawpy_proj/dspawpy_tests/inputs/supplement/Si2.
↳xyz')
>>> len(pymatgen_Structures)
1
>>> pymatgen_Structures = read(datafile='dspawpy_proj/dspawpy_tests/inputs/supplement/aimd.
↳pdb')
>>> len(pymatgen_Structures)
1000
>>> pymatgen_Structures = read(datafile='dspawpy_proj/dspawpy_tests/inputs/2.1/relax.h5')
>>> len(pymatgen_Structures)
3
>>> pymatgen_Structures = read(datafile='dspawpy_proj/dspawpy_tests/inputs/2.1/relax.json')
>>> len(pymatgen_Structures)
3
```

注意 pymatgen_Structures 是由多个 Structure 对象组成的列表, 每个 Structure 对象分别对应一个构型。如果只有一个构型, 也会返回列表, 请使用 pymatgen_Structures[0] 获取 Structure 对象

当 datafile 为列表时, 将依次读取多个文件, 合并成一个 Structures 列表

```
>>> pymatgen_Structures = read(datafile=['dspawpy_proj/dspawpy_tests/inputs/supplement/
↳aimd1.h5', 'dspawpy_proj/dspawpy_tests/inputs/supplement/aimd2.h5'])
```

`dspawpy.io.structure.write` (*structure*, *filename*: *str*, *fmt*: *str* | *None* = *None*, *coords_are_cartesian*: *bool* = *True*)
往结构文件中写入信息

参数

- **structure** -- pymatgen 的 Structure 对象
- **filename** -- 结构文件名
- **fmt** --
 - 结构文件类型, 原生支持 'json', 'as', 'hzw', 'pdb', 'xyz', 'dump' 六种
- **coords_are_cartesian** --
 - 是否写作笛卡尔坐标, 默认为 True; 否则写成分数坐标形式
 - 此选项暂时仅对 as 和 json 格式有效

示例

先读取结构信息:

```
>>> from dspawpy.io.structure import read
>>> s = read('dspawpy_proj/dspawpy_tests/inputs/2.15/01/neb01.h5')
>>> len(s)
17
```

将结构信息写入文件:

```
>>> from dspawpy.io.structure import write
>>> write(s, filename='dspawpy_proj/dspawpy_tests/outputs/doctest/PtH.json', coords_are_
↳cartesian=True)
==> ...PtH.json...
>>> write(s, filename='dspawpy_proj/dspawpy_tests/outputs/doctest/PtH.as', coords_are_
↳cartesian=True)
==> ...PtH.as...
>>> write(s, filename='dspawpy_proj/dspawpy_tests/outputs/doctest/PtH.hzw', coords_are_
↳cartesian=True)
==> ...PtH.hzw...
```

`pdb`, `xyz`, `dump` 三种类型的文件, 可以写入多个构型, 形成“轨迹”。生成的 `xyz` 等轨迹文件可使用 OVITO 等可视化软件打开观察。

```
>>> write(s, filename='dspawpy_proj/dspawpy_tests/outputs/doctest/PtH.pdb', coords_are_
↳cartesian=True)
==> ...PtH.pdb...
>>> write(s, filename='dspawpy_proj/dspawpy_tests/outputs/doctest/PtH.xyz', coords_are_
↳cartesian=True)
==> ...PtH.xyz...
>>> write(s, filename='dspawpy_proj/dspawpy_tests/outputs/doctest/PtH.dump', coords_are_
↳cartesian=True)
==> ...PtH.dump...
```

单结构信息推荐使用 `as` 格式存储, 如果 Structure 中有磁矩或自由度信息, 将会按最完整的格式统一写入, 形如 `Fix_x`, `Fix_y`, `Fix_z`, `Mag_x`, `Mag_y`, `Mag_z`, 自由度信息默认为 F, 磁矩默认为 0.0。可视情况自行手动删除生成的 `as` 文件中的这些默认信息。写成其他类型的结构文件, 将忽略磁矩和自由度信息

可参考 2conversion.py 脚本完成转化：

```

1 # coding:utf-8
2 from dspawpy.io.structure import convert
3
4 convert (
5     infile="dspawpy_proj/dspawpy_tests/inputs/2.1/relax.h5", #_
    ↪待转化结构，如果在当前路径，可以只写文件名
6     si=None, # 筛选构型编号，如果不指定，默认读取全部
7     ele=None, # 筛选元素符号，默认读取所有元素的原子信息
8     ai=None, # 筛选原子编号，从 1 开始，默认读取所有原子信息
9     infmt=None, # 输入结构文件类型，例如 'h5'，如果为None，将根据文件名规则模糊匹配
10    task="relax", # 任务类型，此参数仅当 infile 为文件夹而非文件名时有效
11    outfile="dspawpy_proj/dspawpy_tests/outputs/us/relaxed.xyz", # 结构文件名
12    outfmt=None, # 输出结构文件类型，例如 'xyz'，如果为None，将根据文件名规则模糊匹配
13    coords_are_cartesian=True, # 默认以笛卡尔坐标写入
14 )

```

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	*rnd-str.in/*.*lat.in/*	Monte Carlo Special Quasirandom Structure
X	inp*.xml/*.*in/*.*inp_*	fleur-inp-gen	*.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

i 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

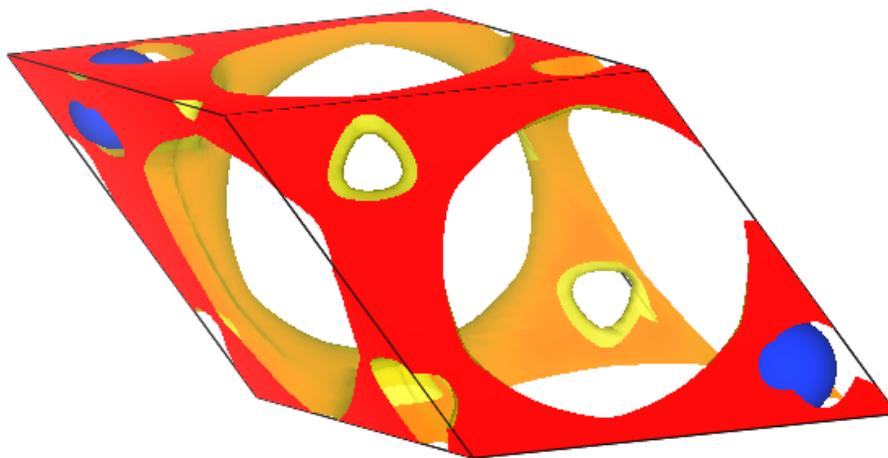
- 参考 3vis_vol.py :

```

1  # coding:utf-8
2  from dspawpy.io.write import write_VESTA
3
4  # 读取数据文件（h5或json格式），处理后输出到cube文件
5  write_VESTA(
6      in_filename="dspawpy_proj/dspawpy_tests/inputs/2.2/rho.h5", #_
7      ↪包含电子体系信息的json或h5文件路径
8      data_type="rho", # 数据类型，支持 "rho", "potential", "elf", "pcharge", "rhoBound"
9      out_filename="dspawpy_proj/dspawpy_tests/outputs/us/DS-PAW_rho.cube", # 输出文件路径
10     gridsize=(10, 10, 10), # 指定插值网格大小
11     format="cube", # 支持cube, vesta和txt格式（xyz格点坐标+数值）
12 )

```

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 :

```

1 # coding:utf-8
2 from dspawpy.io.write import write_delta_rho_vesta
3
4 # 读取数据文件 (h5或json格式), 处理后输出到cube文件, 可直接使用vesta打开且体积小
5 write_delta_rho_vesta(
6     total="dspawpy_proj/dspawpy_tests/inputs/supplement/AB.h5", #_
7     ↪包含所有组元的体系的数据文件
8     individuals=[
9         "dspawpy_proj/dspawpy_tests/inputs/supplement/A.h5",
10        "dspawpy_proj/dspawpy_tests/inputs/supplement/B.h5",
11    ], # 包含各组元的体系的数据文件
12    output="dspawpy_proj/dspawpy_tests/outputs/us/3delta_rho.cube", # 输出文件路径
13 )

```

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

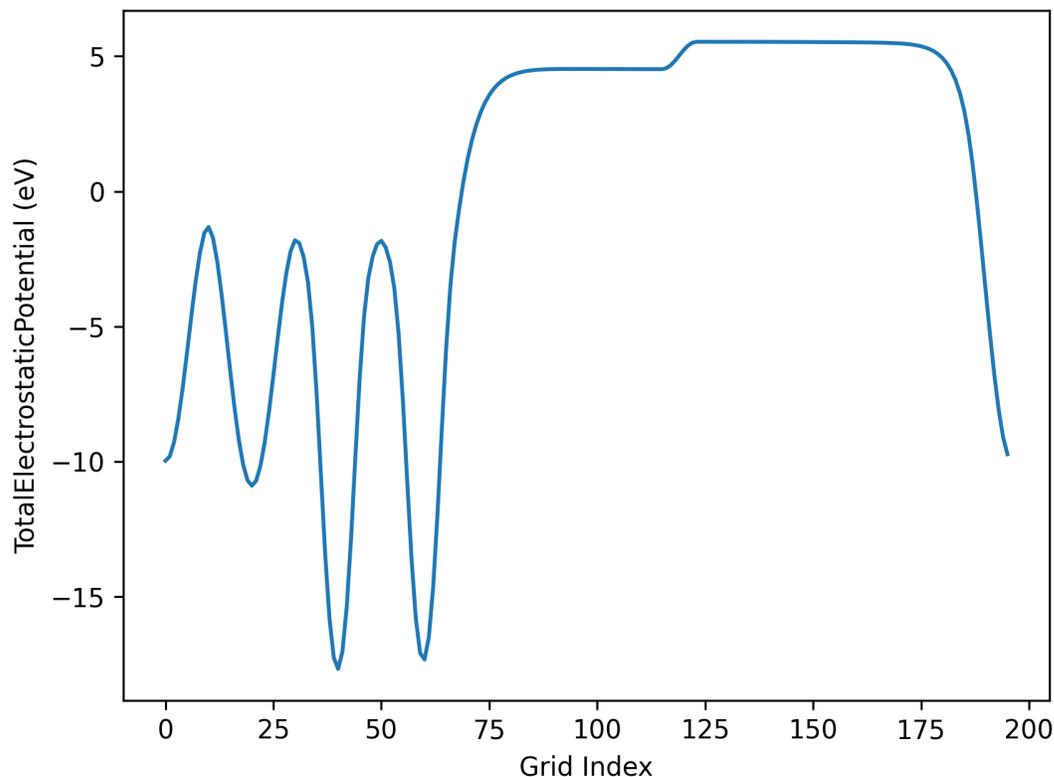
- 参考 3planar_ave.py :

```

1 # coding:utf-8
2 from dspawpy.plot import average_along_axis
3
4 axes = ["2"] # "0", "1", "2" 分别对应xyz轴向, 选择要沿着哪些轴平均
5 axes_indices = [int(i) for i in axes]
6 for ai in axes_indices:
7     plt = average_along_axis(
8         datafile="dspawpy_proj/dspawpy_tests/inputs/3.3/scf.h5", # 数据文件路径
9         task="potential", # 任务名称, 可以是 'rho', 'potential', 'elf', 'pcharge', 'rhoBound'
10        axis=ai, # 沿着哪个轴向绘制势能曲线
11        smooth=False, # 是否平滑
12        smooth_frac=0.8, # 平滑系数
13        subtype=None, # 用于指定task数据子类, 目前仅用于 Potential
14        label=f"axis{ai}", # 图例标签
15    )
16 if len(axes_indices) > 1:
17     plt.legend()
18
19 plt.xlabel("Grid Index")
20 plt.ylabel("TotalElectrostaticPotential (eV)")
21 plt.savefig("dspawpy_proj/dspawpy_tests/outputs/us/3pot_ave.png", dpi=300) # 图片名称

```

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)
```

从包含电子体系信息的 json 或 h5 文件中读取数据并写入 VESTA 格式的文件中

参数

- `in_filename` -- 包含电子体系信息的 json 或 h5 文件路径
- `data_type` -- 数据类型, 支持"rho", "potential", "elf", "pcharge", "rhoBound"
- `out_filename` -- 输出文件路径, 默认"DS-PAW.cube"
- `subtype` -- 用于指定 `data_type` 的数据子类型, 默认为 None, 将读取 potential 的 TotalElectrostaticPotential 数据
- `format` -- 输出的数据格式, 支持"cube" 和"vesta" ("vasp"), 默认为"cube", 大小写不敏感
- `compact` -- 每个格点的数据都换行, 通过减少空格数量减小文件体积 (不会影响 VESTA 软件的解析), 默认为 False
- `inorm` -- 是否归一化体积数据, 使其总和为 1, 默认为 False

- **gridsize** -- 重新定义的格点数, 格式为 (ngx, ngy, ngz), 默认为 None, 即使用原始格点数

返回

VESTA 格式的文件

返回类型

out_filename

示例

```
>>> from dspawpy.io.write import write_VESTA
>>> write_VESTA("dspawpy_proj/dspawpy_tests/inputs/2.2/rho.json", "rho", out_filename=
↳ 'dspawpy_proj/dspawpy_tests/outputs/doctest/rho.cube')
==> ...rho.cube...
```

```
>>> from dspawpy.io.write import write_VESTA
>>> write_VESTA(
...     in_filename="dspawpy_proj/dspawpy_tests/inputs/2.7/potential.h5",
...     data_type="potential",
...     out_filename="dspawpy_proj/dspawpy_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="dspawpy_proj/dspawpy_tests/inputs/2.8/elf.h5",
...     data_type="elf",
...     out_filename="dspawpy_proj/dspawpy_tests/outputs/doctest/elf.cube",
... )
==> ...elf.cube...
>>> write_VESTA(
...     in_filename="dspawpy_proj/dspawpy_tests/inputs/2.9/pcharge.h5",
...     data_type="pcharge",
...     out_filename="dspawpy_proj/dspawpy_tests/outputs/doctest/pcharge.cube",
... )
==> ...pcharge.cube...
>>> write_VESTA(
...     in_filename="dspawpy_proj/dspawpy_tests/inputs/2.7/potential.h5",
...     data_type="potential",
...     out_filename="dspawpy_proj/dspawpy_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...
>>> write_VESTA(
...     in_filename="dspawpy_proj/dspawpy_tests/inputs/2.8/elf.h5",
...     data_type="elf",
...     out_filename="dspawpy_proj/dspawpy_tests/outputs/doctest/elf.vasp",
... )
==> ...elf.vasp...
>>> write_VESTA(
```

(续下页)

(接上页)

```

...     in_filename="dspawpy_proj/dspawpy_tests/inputs/2.9/pcharge.h5",
...     data_type="pcharge",
...     out_filename="dspawpy_proj/dspawpy_tests/outputs/doctest/pcharge.vasp",
... )
==> ...pcharge.vasp...

```

```

>>> write_VESTA(
...     in_filename="dspawpy_proj/dspawpy_tests/inputs/2.7/potential.h5",
...     data_type="potential",
...     out_filename="dspawpy_proj/dspawpy_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("dspawpy_proj/dspawpy_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     -0.8811763
0.165 0.165 0.000     -2.1283865
0.220 0.220 0.000     -4.0559145
0.275 0.275 0.000     -6.8291030
0.330 0.330 0.000    -10.1550909

```

`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)

```

电荷密度差分可视化

DeviceStudio 暂不支持大文件，临时写成可以用 VESTA 打开的格式

参数

- `total` -- 体系总电荷密度文件路径，可以是 h5 或 json 格式
- `individuals` -- 体系各组分电荷密度文件路径，可以是 h5 或 json 格式
- `output` -- 输出文件路径，默认 "delta_rho.cube"
- `format` -- 输出的数据格式，支持 "cube" 和 "vasp"，默认为 "cube"
- `compact` -- 每个格点的数据都换行，通过减少空格数量减小文件体积（不会影响 VESTA 软件的解析），默认为 False
- `inorm` -- 是否归一化体积数据，使其总和为 1，默认为 False

- **gridsize** -- 重新定义的格点数, 格式为 (ngx, ngy, ngz), 默认为 None, 即使用原始格点数

返回

电荷差分 (total-individual1-individual2-...) 后的电荷密度文件,

返回类型

output

示例

```
>>> from dspawpy.io.write import write_delta_rho_vesta
>>> write_delta_rho_vesta(total='dspawpy_proj/dspawpy_tests/inputs/supplement/AB.h5',
...     individuals=['dspawpy_proj/dspawpy_tests/inputs/supplement/A.h5', 'dspawpy_
↳proj/dspawpy_tests/inputs/supplement/B.h5'],
...     output='dspawpy_proj/dspawpy_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)
```

绘制沿着某个轴向的物理量平均值曲线

参数

- **datafile** -- h5 或 json 文件路径或包含任意这些文件的文件夹, 默认'potential.h5'
- **task** -- 任务类型, 可以是'rho', 'potential', 'elf', 'pcharge', 'rhoBound'
- **axis** -- 沿着哪个轴向绘制势能曲线, 默认 2
- **smooth** -- 是否平滑, 默认 False
- **smooth_frac** -- 平滑系数, 默认 0.8
- **raw** -- 是否返回绘图数据到 csv 文件
- **subtype** -- 用于指定 task 数据子类, 默认 None, 代表绘制 Potential/TotalElectrostaticPotential
- ****kwargs** -- 其他参数, 传递给 matplotlib.pyplot.plot

返回

可传递给其他函数进行进一步处理

返回类型

axes

示例

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

读取 potential.h5 文件中的数据, 绘图并保存原始绘图数据到 csv 文件

```
>>> plt = average_along_axis(datafile='dspawpy_proj/dspawpy_tests/inputs/3.3/rho.h5',
↳task='rho', axis=2, smooth=True, smooth_frac=0.8)
>>> plt.savefig('dspawpy_proj/dspawpy_tests/outputs/doctest/rho_h5.png')
>>> plt = average_along_axis(datafile='dspawpy_proj/dspawpy_tests/inputs/3.3/rho.json
```

(续下页)

(接上页)

```

↳', task='rho', axis=2, smooth=True, smooth_frac=0.8)
>>> plt.savefig('dspawpy_proj/dspawpy_tests/outputs/doctest/rho_json.png')

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

>>> plt = average_along_axis(datafile='dspawpy_proj/dspawpy_tests/inputs/2.8/elf.h5',
↳task='elf', axis=2, smooth=True, smooth_frac=0.8)
>>> plt.savefig('dspawpy_proj/dspawpy_tests/outputs/doctest/elf_h5.png')
>>> plt = average_along_axis(datafile='dspawpy_proj/dspawpy_tests/inputs/2.8/elf.json
↳', task='elf', axis=2, smooth=True, smooth_frac=0.8)
>>> plt.savefig('dspawpy_proj/dspawpy_tests/outputs/doctest/elf_json.png')

>>> plt = average_along_axis(datafile='dspawpy_proj/dspawpy_tests/inputs/2.9/pcharge.
↳h5', task='pcharge', axis=2, smooth=True, smooth_frac=0.8)
>>> plt.savefig('dspawpy_proj/dspawpy_tests/outputs/doctest/pcharge_h5.png')
>>> plt = average_along_axis(datafile='dspawpy_proj/dspawpy_tests/inputs/2.9/pcharge.
↳json', task='pcharge', axis=2, smooth=True, smooth_frac=0.8)
>>> plt.savefig('dspawpy_proj/dspawpy_tests/outputs/doctest/pcharge_json.png')

>>> plt = average_along_axis(datafile='dspawpy_proj/dspawpy_tests/inputs/2.28/
↳rhoBound.h5', task='rhoBound', axis=2, smooth=True, smooth_frac=0.8)
>>> plt.savefig('dspawpy_proj/dspawpy_tests/outputs/doctest/rhoBound_h5.png')
>>> plt = average_along_axis(datafile='dspawpy_proj/dspawpy_tests/inputs/2.28/
↳rhoBound.json', task='rhoBound', axis=2, smooth=True, smooth_frac=0.8)
>>> plt.savefig('dspawpy_proj/dspawpy_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`，可将能量零点修改向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`：

```

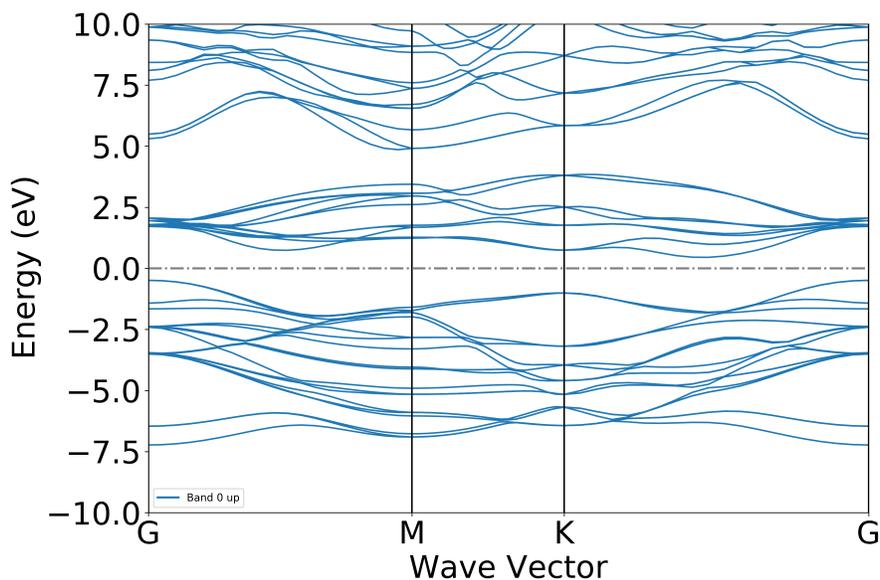
1  # coding:utf-8
2  import os
3
4  import matplotlib.pyplot as plt
5  from pymatgen.electronic_structure.plotter import BSPlotter
6
7  from dspawpy.io.read import get_band_data
8
9  datafile = "dspawpy_proj/dspawpy_tests/inputs/supplement/pband.h5" # 指定数据文件路径
10 band_data = get_band_data(
11     band_dir=datafile,
12     syst_dir=None, # system.json文件路径，仅当band_dir为json文件时需要
13     efermi=None, # 用于手动修正费米能级
14     zero_to_efermi=True, # 非金属体系应将零点能移动到费米能级
15 )
16
17 bsp = BSPlotter(band_data)
18 axes_or_plt = bsp.get_plot(
19     zero_to_efermi=False, # 上面读取数据时已经平移，此处应关闭
20     ylim=[-10, 10], # 能带图的纵坐标范围
21     smooth=False, # 是否对能带图进行平滑处理
22     vbm_cbm_marker=False, # 是否在能带图中标记导带和价带
23     smooth_tol=0, # 平滑处理的阈值
24     smooth_k=3, # 平滑处理的阶数
25     smooth_np=100, # 平滑处理的点数
26 )
27
28 if isinstance(axes_or_plt, plt.Axes):
29     fig = axes_or_plt.get_figure() # version newer than v2023.8.10
30 else:
31     fig = axes_or_plt.gcf() # older version pymatgen
32
33 # 添加能量零点参考线
34 for ax in fig.axes:
35     ax.axhline(0, lw=2, ls="-.", color="gray")
36
37 filename = "dspawpy_proj/dspawpy_tests/outputs/us/4bandplot.png" # 输出的能带图文件名
38 os.makedirs(os.path.dirname(os.path.abspath(filename)), exist_ok=True)
39 fig.savefig(filename, dpi=300)

```

i 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：



二硫化铋能带示意图

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`：

```

1 # coding:utf-8
2 import os
3
4 import matplotlib.pyplot as plt
5 import numpy as np
6 from pymatgen.electronic_structure.plotter import BSPlotterProjected
7
8 from dspawpy.io.read import get_band_data
9
10 datafile = "dspawpy_proj/dspawpy_tests/inputs/supplement/pband.h5" # 指定数据文件路径
11 band_data = get_band_data(
12     band_dir=datafile,
13     syst_dir=None, # system.json文件路径，仅当band_dir为json文件时需要
14     efermi=None, # 用于手动修正费米能级
15     zero_to_efermi=True, # 非金属体系应将零点能移动到费米能级
16 )
17
18 bsp = BSPlotterProjected(bs=band_data) # 初始化BSPlotterProjected类
19 axes_or_plt = bsp.get_elt_projected_plots(
20     zero_to_efermi=False, # 上面读取数据时已经平移，此处应关闭
21     ylim=[-8, 5], # 设置能量范围

```

(续下页)

(接上页)

```

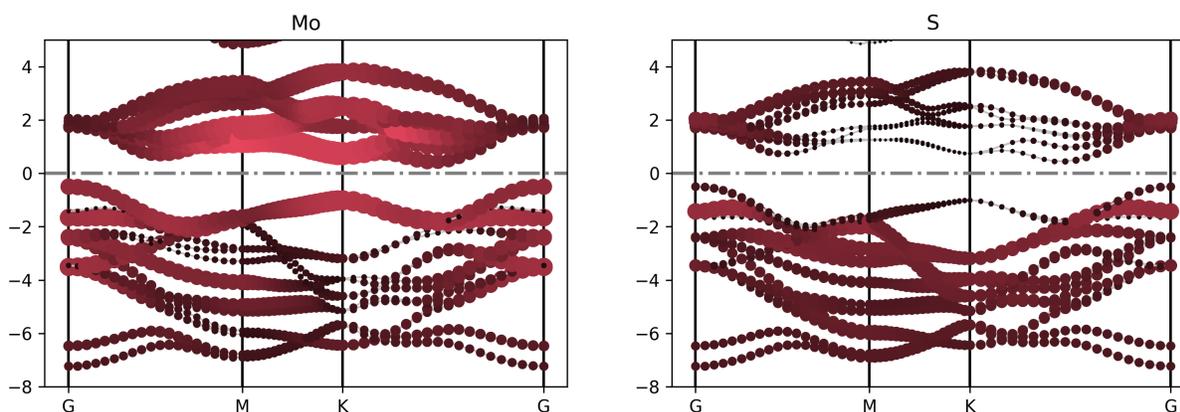
22     vbm_cbm_marker=False, # 是否标记导带底和价带顶
23 )
24
25 if isinstance(axes_or_plt, plt.Axes):
26     fig = axes_or_plt.get_figure() # version newer than v2023.8.10
27 elif np.iterable(axes_or_plt):
28     fig = np.asarray(axes_or_plt).flatten()[0].get_figure()
29 else:
30     fig = axes_or_plt.gcf() # older version pymatgen
31
32 # 添加能量零点参考线
33 for ax in fig.axes:
34     ax.axhline(0, lw=2, ls="-.", color="gray")
35
36 figname = (
37     "dspawpy_proj/dspawpy_tests/outputs/us/4bandplot_elt.png" # 输出的能带图文件名
38 )
39 os.makedirs(os.path.dirname(os.path.abspath(figname)), exist_ok=True)
40 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 :

```

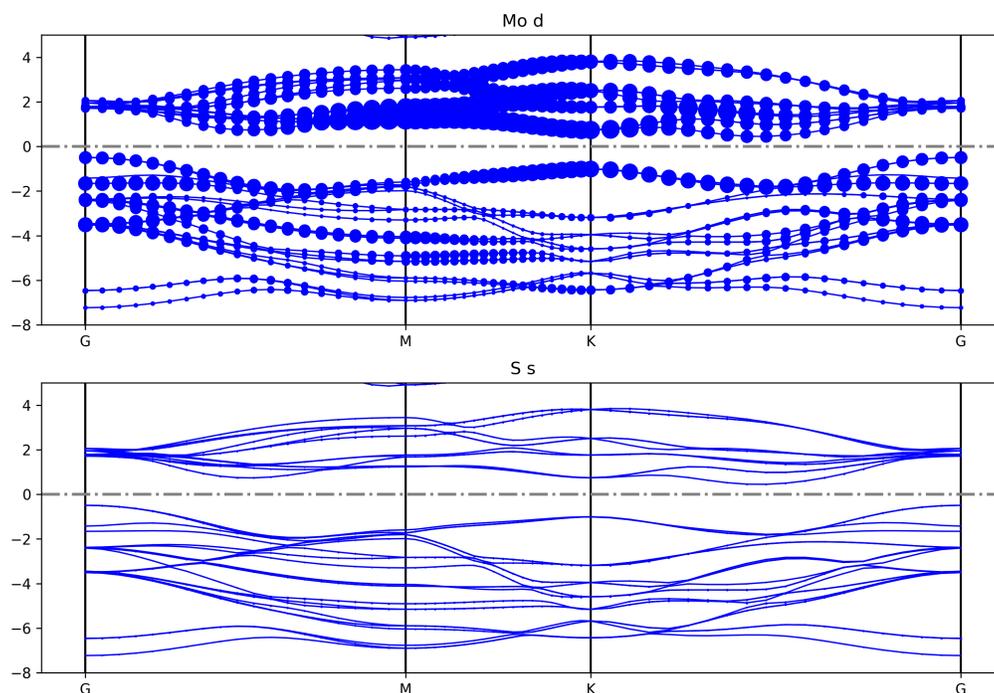
1  # coding:utf-8
2  import os
3
4  import matplotlib.pyplot as plt
5  import numpy as np
6  from pymatgen.electronic_structure.plotter import BSPlotterProjected
7
8  from dspawpy.io.read import get_band_data
9
10 datafile = "dspawpy_proj/dspawpy_tests/inputs/supplement/pband.h5" # 指定数据文件路径
11 band_data = get_band_data(
12     band_dir=datafile,
13     syst_dir=None, # system.json文件路径, 仅当band_dir为json文件时需要
14     efermi=None, # 用于手动修正费米能级
15     zero_to_efermi=True, # 非金属体系应将零点能移动到费米能级
16 )
17
18 bsp = BSPlotterProjected(bs=band_data) # 初始化BSPlotterProjected类
19 # ! 选定元素与轨道, 创建字典
20 dict_elem_orbit = {"Mo": ["d"], "S": ["s"]}
21
22 axes_or_plt = bsp.get_projected_plots_dots(
23     dictio=dict_elem_orbit,
24     zero_to_efermi=False, # 上面读取数据时已经平移, 此处应关闭
25     ylim=[-8, 5], # 设置能量范围
26     vbm_cbm_marker=False, # 是否标记导带底和价带顶
27 )
28
29 if isinstance(axes_or_plt, plt.Axes):
30     fig = axes_or_plt.get_figure() # version newer than v2023.8.10
31 elif np.iterable(axes_or_plt):
32     fig = np.asarray(axes_or_plt).flatten()[0].get_figure()
33 else:
34     fig = axes_or_plt.gcf() # older version pymatgen
35
36 # 添加能量零点参考线
37 for ax in fig.axes:
38     ax.axhline(0, lw=2, ls="-.", color="gray")
39
40 filename = "dspawpy_proj/dspawpy_tests/outputs/us/4bandplot_elt_orbit.png" # 输出的能带图文件名
41 os.makedirs(os.path.dirname(os.path.abspath(filename)), exist_ok=True)
42 fig.savefig(filename, 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 :

```
1 # coding:utf-8
2 import os
3
4 import matplotlib.pyplot as plt
5 import numpy as np
6 from pymatgen.electronic_structure.plotter import BSPlotterProjected
7
8 from dspawpy.io.read import get_band_data
9
```

(续下页)

(接上页)

```

10 datafile = "dspawpy_proj/dspawpy_tests/inputs/supplement/pband.h5" # 指定数据文件路径
11 band_data = get_band_data(
12     band_dir=datafile,
13     syst_dir=None, # system.json文件路径, 仅当band_dir为json文件时需要
14     efermi=None, # 用于手动修正费米能级
15     zero_to_efermi=True, # 非金属体系应将零点能移动到费米能级
16 )
17
18 bsp = BSPlotterProjected(bs=band_data)
19 # ! 指定元素和轨道和原子序号
20 dict_elem_orbit = {"Mo": ["px", "py", "pz"]}
21 dict_elem_index = {"Mo": [1]}
22
23 axes_or_plt = bsp.get_projected_plots_dots_patom_pmorb(
24     dictio=dict_elem_orbit, # 指定元素-轨道字典
25     dictpa=dict_elem_index, # 指定元素-原子序号字典
26     sum_atoms=None, # 是否对原子求和
27     sum_morbs=None, # 是否对轨道求和
28     zero_to_efermi=False, # 上面读取数据时已经平移, 此处应关闭
29     ylim=None, # 设置能量范围
30     vbm_cbm_marker=False, # 是否标记导带底和价带顶
31     selected_branches=None, # 指定绘制的能带分支
32     w_h_size=(12, 8), # 设置图片宽高
33     num_column=None, # 设置每行显示的图片数量
34 )
35
36 if isinstance(axes_or_plt, plt.Axes):
37     fig = axes_or_plt.get_figure() # version newer than v2023.8.10
38 elif np.iterable(axes_or_plt):
39     fig = np.asarray(axes_or_plt).flatten()[0].get_figure()
40 else:
41     fig = axes_or_plt.gcf() # older version pymatgen
42
43 # 添加能量零点参考线
44 for ax in fig.axes:
45     ax.axhline(0, lw=2, ls="--", color="gray")
46
47 figname = (
48     "dspawpy_proj/dspawpy_tests/outputs/us/4band_patom_porbit.png" #_
49     ↪输出的能带图文件名
50 )
51 os.makedirs(os.path.dirname(os.path.abspath(figname)), exist_ok=True)
52 fig.savefig(figname, dpi=300)

```

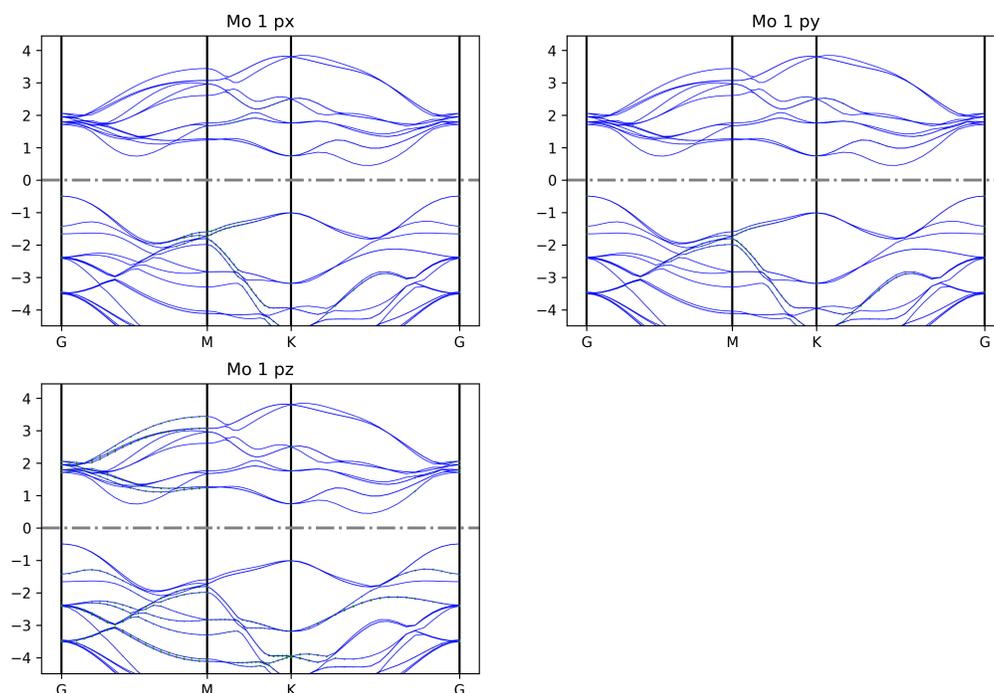
i 知识点:

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_pmor` 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

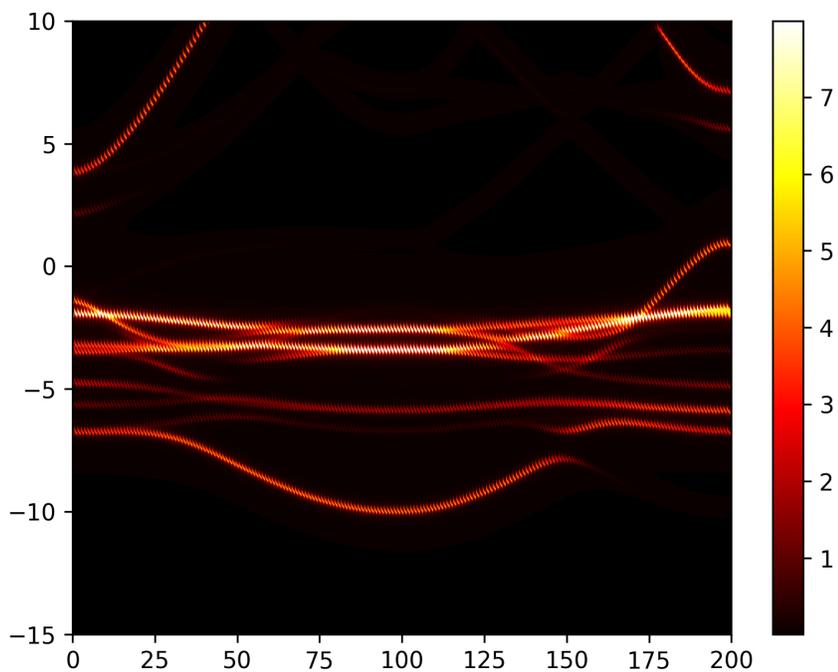
参考 `4bandunfolding.py`：

```
1 # coding:utf-8
2 import os
```

(续下页)

```
3
4 from dspawpy.plot import plot_bandunfolding
5
6 plt = plot_bandunfolding(
7     datafile="dspawpy_proj/dspawpy_tests/inputs/2.22.1/band.h5", # 读取数据
8     ef=None, # 费米能级, 从文件中读取
9     de=0.05, # 能带宽度, 默认0.05
10    dele=0.06, # 能带间隔, 默认0.06
11)
12
13 plt.ylim(-15, 10)
14 figname = (
15     "dspawpy_proj/dspawpy_tests/outputs/us/4bandunfolding.png" # 输出的能带图文件名
16 )
17 os.makedirs(os.path.dirname(os.path.abspath(figname)), exist_ok=True)
18 plt.savefig(figname, dpi=300)
19 # 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 :

```
1 # coding:utf-8
2 import os
3
4 from pymatgen.electronic_structure.plotter import BSPlotter
5
6 from dspawpy.io.read import get_band_data
7
8 band_data = get_band_data(
9     band_dir="dspawpy_proj/dspawpy_tests/inputs/2.30/wannier.h5", # 瓦尼尔能带文件路径
10    syst_dir=None, # system.json文件路径, 仅当band_dir为json文件时需要
11    efermi=None, # 用于手动修正费米能级
12    zero_to_efermi=False, # 是否将零点能移动到费米能级
13 )
14 bsp = BSPlotter(bs=band_data)
15 band_data = get_band_data(
16    band_dir="dspawpy_proj/dspawpy_tests/inputs/2.3/band.h5", # 读取DFT能带
17    syst_dir=None, # system.json文件路径, 仅当band_dir为json文件时需要
18    efermi=None, # 用于手动修正费米能级
19    zero_to_efermi=False, # 是否将零点能移动到费米能级
20 )
21
22 bsp2 = BSPlotter(bs=band_data)
23 bsp.add_bs(bsp2._bs)
24 axes_or_plt = bsp.get_plot(
25     zero_to_efermi=True, # 将零点能移动到费米能级
26     ylim=[-10, 10], # 能带图的纵坐标范围
27     smooth=False, # 是否对能带图进行平滑处理
28     vbm_cbm_marker=False, # 是否在能带图中标记导带和价带
29     smooth_tol=0, # 平滑处理的阈值
30     smooth_k=3, # 平滑处理的阶数
31     smooth_np=100, # 平滑处理的点数
32     bs_labels=["wannier interpolated", "DFT"], # 能带图中的标签
33 )
34
35 import matplotlib.pyplot as plt # noqa: E402
36
37 if isinstance(axes_or_plt, plt.Axes):
38     fig = axes_or_plt.get_figure() # version newer than v2023.8.10
39 else:
40     fig = axes_or_plt.gcf() # older version pymatgen
41
```

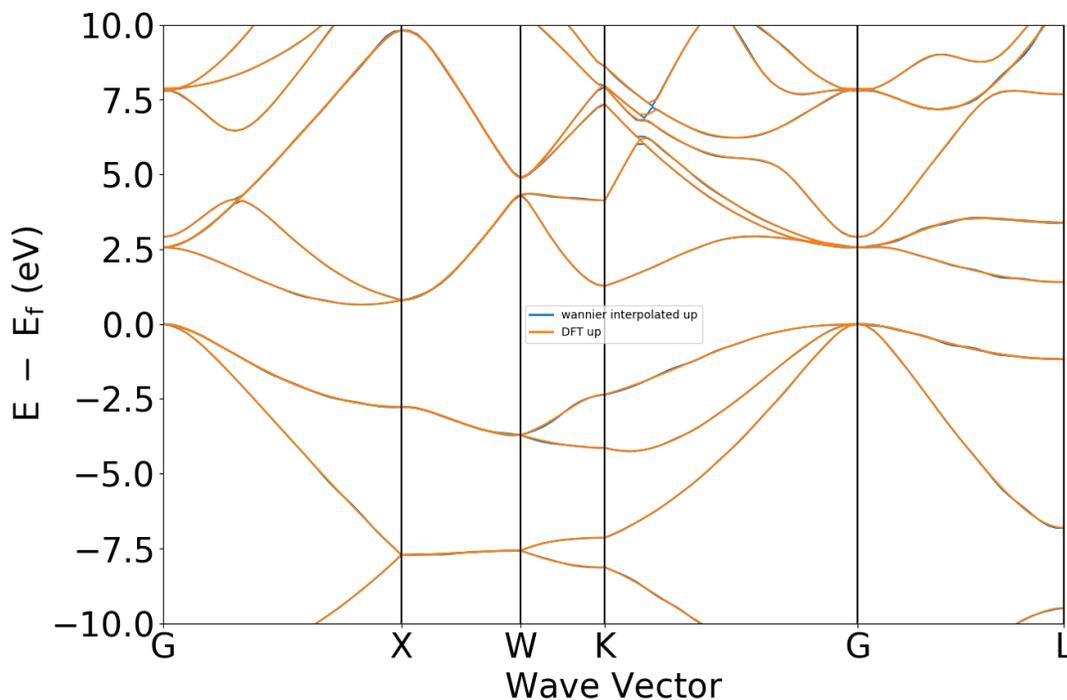
(续下页)

```

42 # 添加能量零点参考线
43 for ax in fig.axes:
44     ax.axhline(0, lw=2, ls="-.", color="gray")
45
46 filename = "dspawpy_proj/dspawpy_tests/outputs/us/4wannierBand.png" # 输出的能带图文件名
47 os.makedirs(os.path.dirname(os.path.abspath(filename)), exist_ok=True)
48 fig.savefig(filename, 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

```

读取 h5 或 json 文件中的能带数据，构建 `BandStructureSymmLine` 对象

参数

- `band_dir` --
 - * 能带文件路径，band.h5 / band.json 或包含 band.h5 / band.json 的文件夹
 - * 注意，wannier.h5 也可以使用此函数读取，但 `band_dir` 不支持文件夹类型
- `syst_dir` -- system.json 路径，仅为辅助处理 Wannier 数据而准备（从中读取结构和费米能级）
- `efermi` -- 费米能级，如果 h5 文件中的费米能级不正确，可以通过此参数指定费米能级

- `zero_to_efermi` -- 是否将费米能级移动到 0

返回类型

BandStructureSymmLine

示例

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

如果希望通过指定 `wannier.json` 来处理瓦尼尔能带，需要额外指定 `syst_dir` 参数

```
>>> band = get_band_data(band_dir='dspawpy_proj/dspawpy_tests/inputs/2.30/wannier.h5')
>>> band = get_band_data(band_dir='dspawpy_proj/dspawpy_tests/inputs/2.30/wannier.json
↳', syst_dir='dspawpy_proj/dspawpy_tests/inputs/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` :

```
1 # coding:utf-8
2 import os
3
4 from pymatgen.electronic_structure.plotter import DosPlotter
5
6 from dspawpy.io.read import get_dos_data
7 from dspawpy.plot import plot_dos
8
9 dos_data = get_dos_data(
10     dos_dir="dspawpy_proj/dspawpy_tests/inputs/3.2.4/dos.h5", # 读取投影态密度数据
11     return_dos=False, # 如果为False, 则统一返回 CompleteDos对象 (无论计算时是否开了投影)
12 )
13 dos_plotter = DosPlotter(
14     zero_at_efermi=True, # 是否将费米能级作为零点
15     stack=False, # True表示绘制面积图
16     sigma=None, # 高斯展宽, None表示不进行平滑处理
17 )
18 dos_plotter.add_dos(label="total dos", dos=dos_data) # 设置态密度图例 # 传入态密度数据
19
20 ax = plot_dos(
21     dosplotter=dos_plotter,
```

(续下页)

```

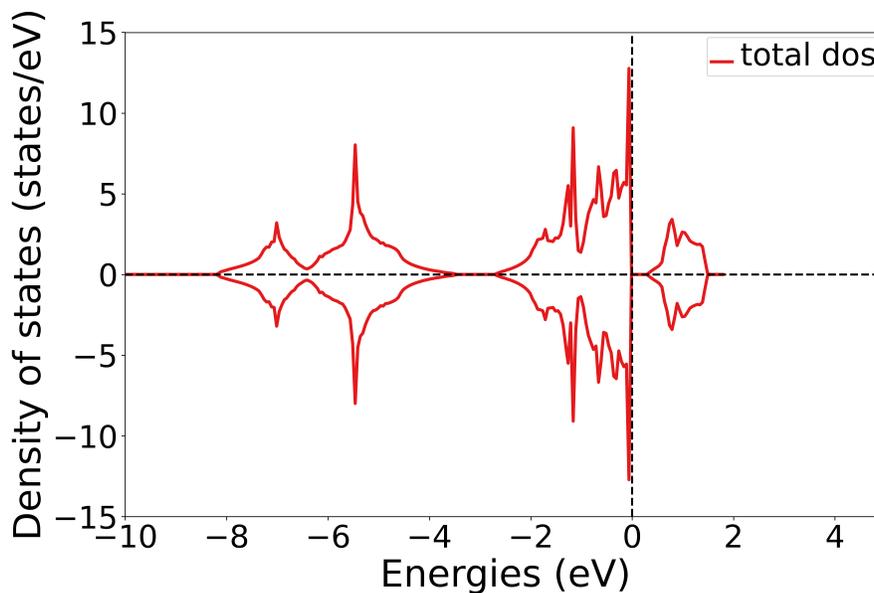
22     xlim=[-10, 5], # 设置能量范围
23     ylim=[-15, 15], # 设置态密度范围
24 )
25 ax.axhline(0, lw=2, ls="-.", color="gray")
26
27 filename = (
28     "dspawpy_proj/dspawpy_tests/outputs/us/5dos_total.png" # 输出的态密度图文件名
29 )
30 os.makedirs(os.path.dirname(os.path.abspath(filename)), exist_ok=True)
31
32 fig = ax.get_figure()
33 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

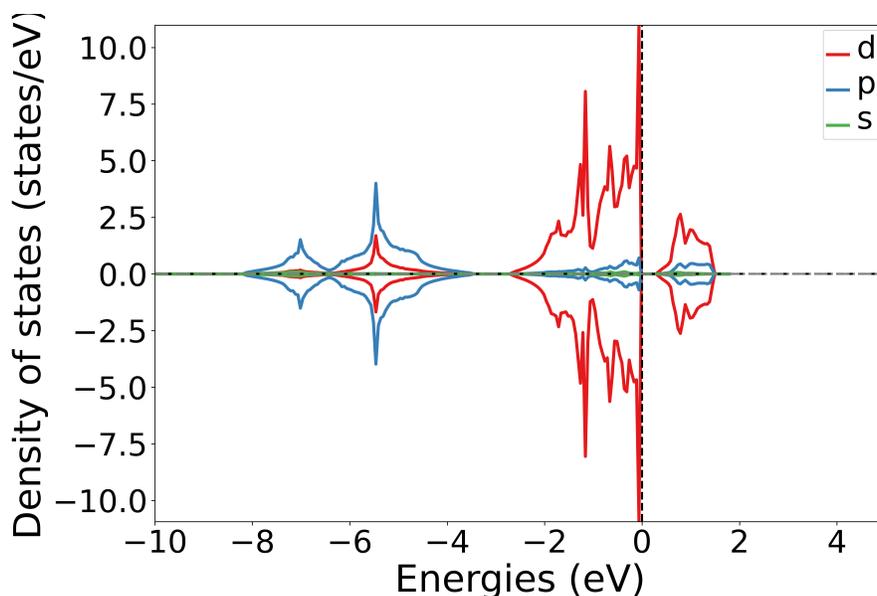
参考 5dosplot_spd.py :

```
1 # coding:utf-8
2 import os
3
4 from pymatgen.electronic_structure.plotter import DosPlotter
5
6 from dspawpy.io.read import get_dos_data
7 from dspawpy.plot import plot_dos
8
9 dos_data = get_dos_data(
10     dos_dir="dspawpy_proj/dspawpy_tests/inputs/3.2.4/dos.h5", # 读取投影态密度数据
11     return_dos=False, # 如果为False, 则统一返回 CompleteDos对象 (无论计算时是否开了投影)
12 )
13 dos_plotter = DosPlotter(
14     zero_at_efermi=True, # 是否将费米能级作为零点
15     stack=False, # True表示绘制面积图
16     sigma=None, # 高斯展宽, None表示不进行平滑处理
17 )
18 dos_plotter.add_dos_dict(
19     dos_dict=dos_data.get_spd_dos(),
20     key_sort_func=None, # 轨道投影 # 指定排序函数
21 )
22 ax = plot_dos(
23     dosplotter=dos_plotter,
24     xlim=[-10, 5], # 设置能量范围
25     ylim=None, # 设置态密度范围
26 )
27
28 ax.axhline(0, lw=2, ls="-.", color="gray")
29
30 filename = "dspawpy_proj/dspawpy_tests/outputs/us/5dos_spd.png" # 输出的态密度图文件名
31 os.makedirs(os.path.dirname(os.path.abspath(filename)), exist_ok=True)
32
33 fig = ax.get_figure()
34 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 :

```
1 # coding:utf-8
2 import os
3
4 from pymatgen.electronic_structure.plotter import DosPlotter
5
6 from dspawpy.io.read import get_dos_data
7 from dspawpy.plot import plot_dos
8
9 dos_data = get_dos_data(
10     dos_dir="dspawpy_proj/dspawpy_tests/inputs/3.2.4/dos.h5", # 读取投影态密度数据
11     return_dos=False, # 如果为False, 则统一返回 CompleteDos对象 (无论计算时是否开了投影)
12 )
13 dos_plotter = DosPlotter(
14     zero_at_efermi=True, # 是否将费米能级作为零点
15     stack=False, # True表示绘制面积图
16     sigma=None, # 高斯展宽, None表示不进行平滑处理
17 )
18 dos_plotter.add_dos_dict(
19     dos_dict=dos_data.get_element_dos(),
```

(续下页)

(接上页)

```

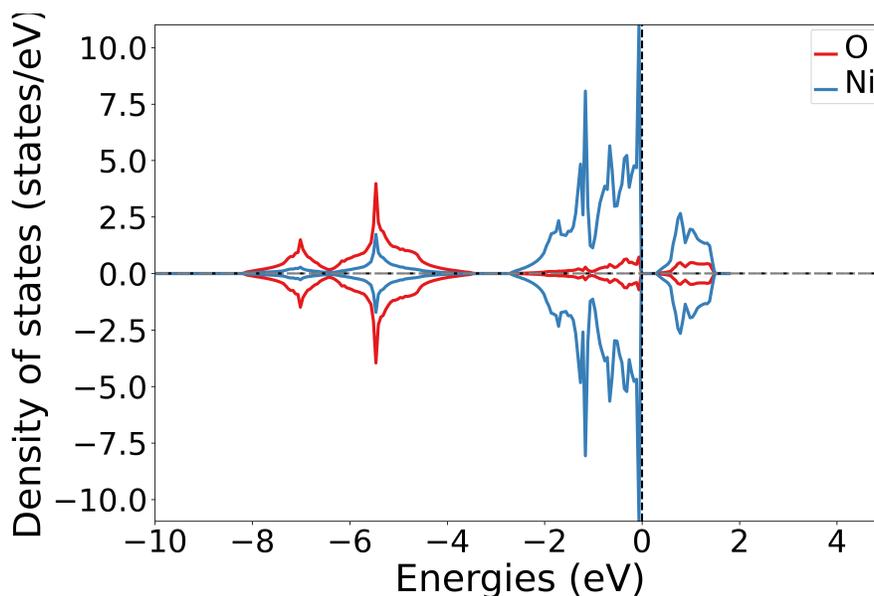
20     key_sort_func=None, # 元素投影态密度 # 指定排序函数
21 )
22
23 ax = plot_dos(
24     dosplotter=dos_plotter,
25     xlim=[-10, 5], # 设置能量范围
26     ylim=None, # 设置态密度范围
27 )
28 ax.axhline(0, lw=2, ls="-.", color="gray")
29
30 filename = "dspawpy_proj/dspawpy_tests/outputs/us/5dos_elt.png" # 输出的态密度图文件名
31 os.makedirs(os.path.dirname(os.path.abspath(filename)), exist_ok=True)
32
33 fig = ax.get_figure()
34 fig.savefig(filename, dpi=300)

```

i 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 :

```

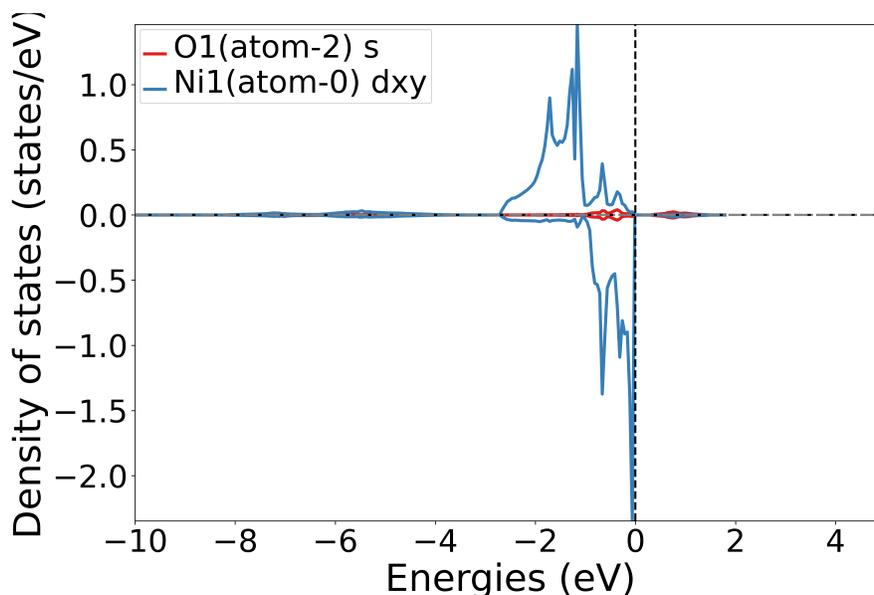
1  # coding:utf-8
2  import os
3
4  from pymatgen.electronic_structure.core import Orbital
5  from pymatgen.electronic_structure.plotter import DosPlotter
6
7  from dspawpy.io.read import get_dos_data
8  from dspawpy.plot import plot_dos
9
10 dos_data = get_dos_data(
11     dos_dir="dspawpy_proj/dspawpy_tests/inputs/3.2.4/dos.h5", # 读取投影态密度数据
12     return_dos=False, # 如果为False, 则统一返回CompleteDos对象 (无论计算时是否开了投影)
13 )
14 dos_plotter = DosPlotter(
15     zero_at_efermi=True, # 是否将费米能级作为零点
16     stack=False, # True表示绘制面积图
17     sigma=None, # 高斯展宽, None表示不进行平滑处理
18 )
19
20 # ! 指定原子序号和轨道
21 dict_index_orbit = {0: ["dxy"], 2: ["s"]}
22
23 print("正在绘图...")
24 for index in dict_index_orbit:
25     _os = dict_index_orbit[index]
26     _e = str(dos_data.structure.sites[index].species)
27     for _orb in _os:
28         dos_plotter.add_dos(
29             f"{_e}(atom-{index}) {_orb}", # label
30             dos_data.get_site_orbital_dos(
31                 dos_data.structure[index], getattr(Orbital, _orb),
32             ),
33         )
34
35 ax = plot_dos(
36     dosplotter=dos_plotter,
37     xlim=[-10, 5], # 设置能量范围
38     ylim=None, # 设置态密度范围
39 )
40 ax.axhline(0, lw=2, ls="-. ", color="gray")
41
42 filename = (
43     "dspawpy_proj/dspawpy_tests/outputs/us/5dos_atom_orbit.png" # 输出的态密度图文件名
44 )
45 os.makedirs(os.path.dirname(os.path.abspath(filename)), exist_ok=True)
46
47 fig = ax.get_figure()
48 fig.savefig(filename, 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` :

```
1 # coding:utf-8
2 import os
3
4 from pymatgen.electronic_structure.plotter import DosPlotter
5
6 from dspawpy.io.read import get_dos_data
7 from dspawpy.plot import plot_dos
8
9 dos_data = get_dos_data(
```

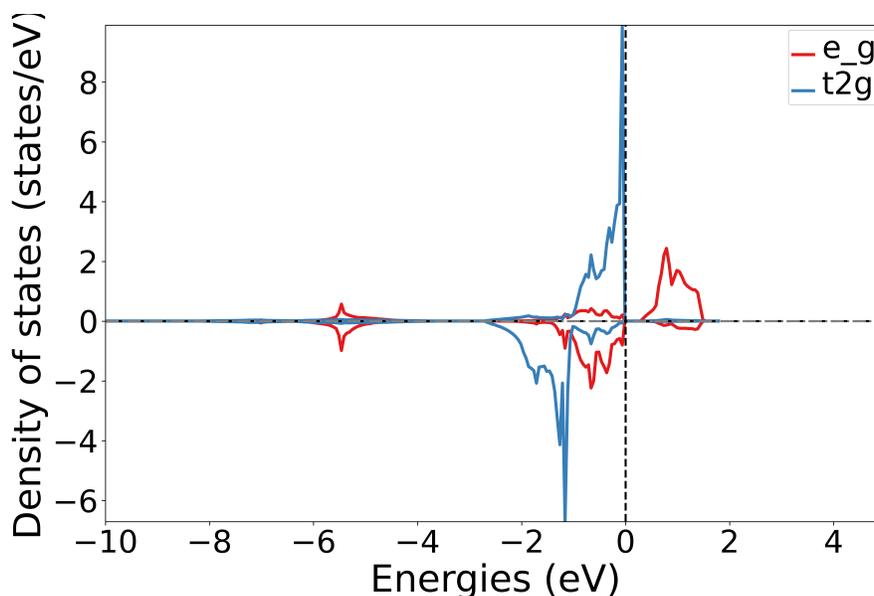
(续下页)

```
10     dos_dir="dspawpy_proj/dspawpy_tests/inputs/3.2.4/dos.h5", # 读取投影态密度数据
11     return_dos=False, # 如果为False, 则统一返回CompleteDos对象 (无论计算时是否开了投影)
12 )
13 dos_plotter = DosPlotter(
14     zero_at_efermi=True, # 是否将费米能级作为零点
15     stack=False, # True表示绘制面积图
16     sigma=None, # 高斯展宽, None表示不进行平滑处理
17 )
18 # print(dos_data.structure)
19
20 # ! 指定原子序号, 从0开始
21 ais = [1]
22
23 print("正在绘图...")
24 atom_indices = [int(ai) for ai in ais]
25 for atom_index in atom_indices:
26     dos_plotter.add_dos_dict(
27         dos_data.get_site_t2g_eg_resolved_dos(dos_data.structure[atom_index]),
28     )
29
30 ax = plot_dos(
31     dosplotter=dos_plotter,
32     xlim=[-10, 5], # 设置能量范围
33     ylim=None, # 设置态密度范围
34 )
35 ax.axhline(0, lw=2, ls="-.", color="gray")
36
37 filename = (
38     "dspawpy_proj/dspawpy_tests/outputs/us/5dos_t2g_eg.png" # 输出的态密度图文件名
39 )
40 os.makedirs(os.path.dirname(os.path.abspath(filename)), exist_ok=True)
41
42 fig = ax.get_figure()
43 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：

```
1 # coding:utf-8
2 from dspawpy.io.read import get_dos_data
3 from dspawpy.io.utils import d_band
4
5 dos_data = get_dos_data(
6     dos_dir="dspawpy_proj/dspawpy_tests/inputs/supplement/dos.h5", # 读取投影态密度数据
7     return_dos=False, # 如果为 False, 则统一返回 CompleteDos 对象 (无论计算时是否开了投影)
8 )
9 for spin in dos_data.densities:
10     print("spin=", spin)
11     c = d_band(spin, dos_data)
12     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)

读取 h5 或 json 文件中的态密度数据，构建 CompleteDos 或 DOS 对象

参数

- **dos_dir** -- 态密度文件路径，dos.h5 / dos.json 或包含 dos.h5 / dos.json 的文件夹
- **return_dos** (bool, optional) -- 是否返回 DOS 对象，如果为 False，则统一返回 CompleteDos 对象（无论计算时是否开了投影）

返回类型

CompleteDos or Dos

示例

```
>>> from dspawpy.io.read import get_dos_data
>>> dos = get_dos_data(dos_dir='dspawpy_proj/dspawpy_tests/inputs/2.5/dos.h5')
>>> dos = get_dos_data(dos_dir='dspawpy_proj/dspawpy_tests/inputs/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：

```
1 # coding:utf-8
2 import os
3
4 import numpy as np
5 from matplotlib.axes import Axes
6 from pymatgen.electronic_structure.plotter import BSDOSPlotter
7
8 from dspawpy.io.read import get_band_data, get_dos_data
9
10 bandfile = "dspawpy_proj/dspawpy_tests/inputs/2.3/band.h5" # 普通能带数据
11 band_data = get_band_data(
12     band_dir=bandfile,
```

(续下页)

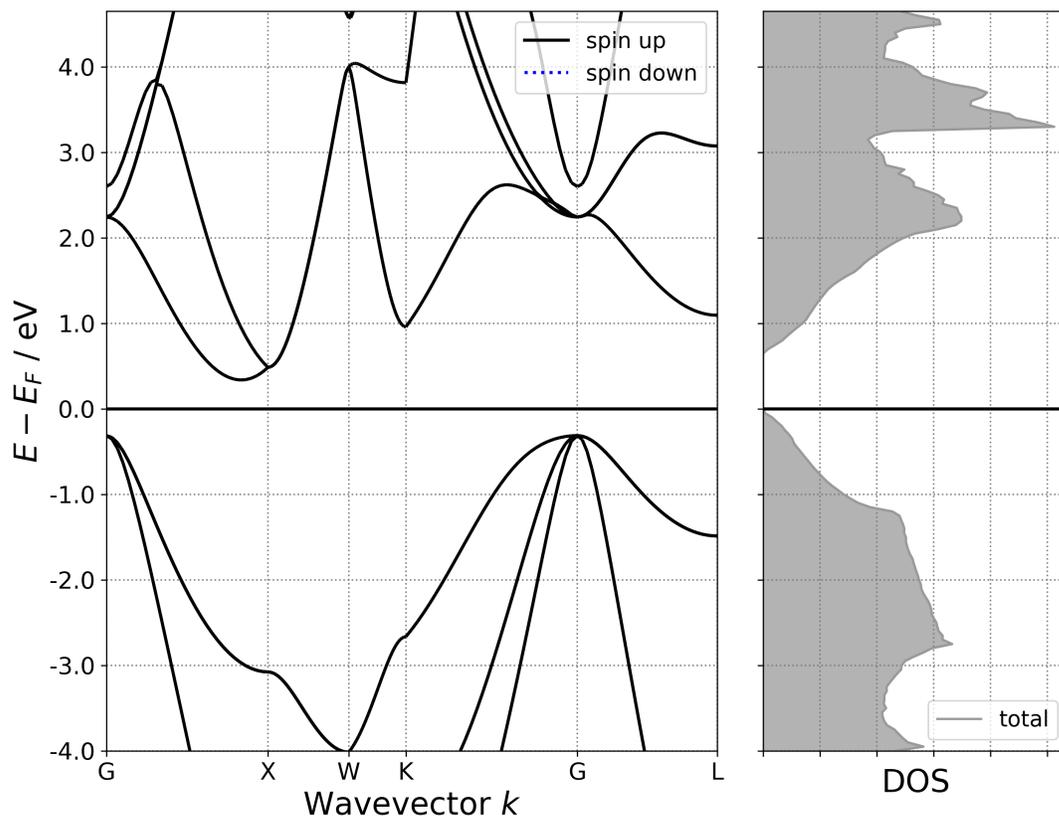
(接上页)

```

13     syst_dir=None, # system.json文件路径, 仅当band_dir为json文件时需要
14     efermi=None, # 用于手动修正费米能级
15 )
16 band_efermi = band_data.efermi
17 dosfile = "dspawpy_proj/dspawpy_tests/inputs/2.5/dos.h5" # 态密度数据
18 dos_data = get_dos_data(
19     dos_dir=dosfile,
20     return_dos=False, # 如果为False, 则统一返回CompleteDos对象 (无论计算时是否开了投影)
21 )
22 dos_efermi = dos_data.efermi
23 bdp = BSDOSPlotter(
24     bs_projection=None, # 能带的投影方式, None表示不投影
25     dos_projection=None, # 态密度的投影方式, None表示不投影
26     vb_energy_range=4, # 价带能量范围
27     cb_energy_range=4, # 导带能量范围
28     fixed_cb_energy=False, # 是否固定导带能量范围
29     egrid_interval=1, # 能量格点间隔
30     font="DejaVu Sans", # 默认Times New Roman, 修改成 DejaVu_
    ↪ Sans, 可避免在linux上因字体未安装而出现的警告
31     axis_fontsize=20, # 坐标轴字体大小
32     tick_fontsize=15, # 刻度字体大小
33     legend_fontsize=14, # 图例字体大小
34     bs_legend="best", # 能带图图例位置
35     dos_legend="best", # 态密度图图例位置
36     rgb_legend=True, # 是否使用彩色图例
37     fig_size=(11, 8.5), # 图片大小
38 )
39 if band_efermi != dos_efermi:
40     print(f"{band_efermi:.4f} eV")
41     print(f"{dos_efermi:.4f} eV")
42     d_efermi = band_efermi - dos_efermi
43
44     print("! Band Dos费米能级不一致, 将以Dos为准")
45     band_data.bands = {spin: v + d_efermi for spin, v in band_data.bands.items()}
46
47     # print("! Band Dos费米能级不一致, 将以Band为准")
48     # dos_data.energies -= d_efermi
49
50 axes_or_plt = bdp.get_plot(bs=band_data, dos=dos_data) # 传入能带数据 # 传入态密度数据
51
52 if isinstance(axes_or_plt, Axes):
53     fig = axes_or_plt.get_figure() # version newer than v2023.8.10
54 elif np.iterable(axes_or_plt):
55     fig = np.asarray(axes_or_plt).flatten()[0].get_figure()
56 else:
57     fig = axes_or_plt.gcf() # older version pymatgen
58
59 filename = (
60     "dspawpy_proj/dspawpy_tests/outputs/us/6bandDos.png" # 输出的能带-态密度图文件名
61 )
62 os.makedirs(os.path.dirname(os.path.abspath(filename)), exist_ok=True)
63 fig.savefig(filename, dpi=300)
64 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 :

```
1 # coding:utf-8
2 import os
3
4 import numpy as np
5 from matplotlib.axes import Axes
6 from pymatgen.electronic_structure.plotter import BSDOSPlotter
7
```

(续下页)

(接上页)

```

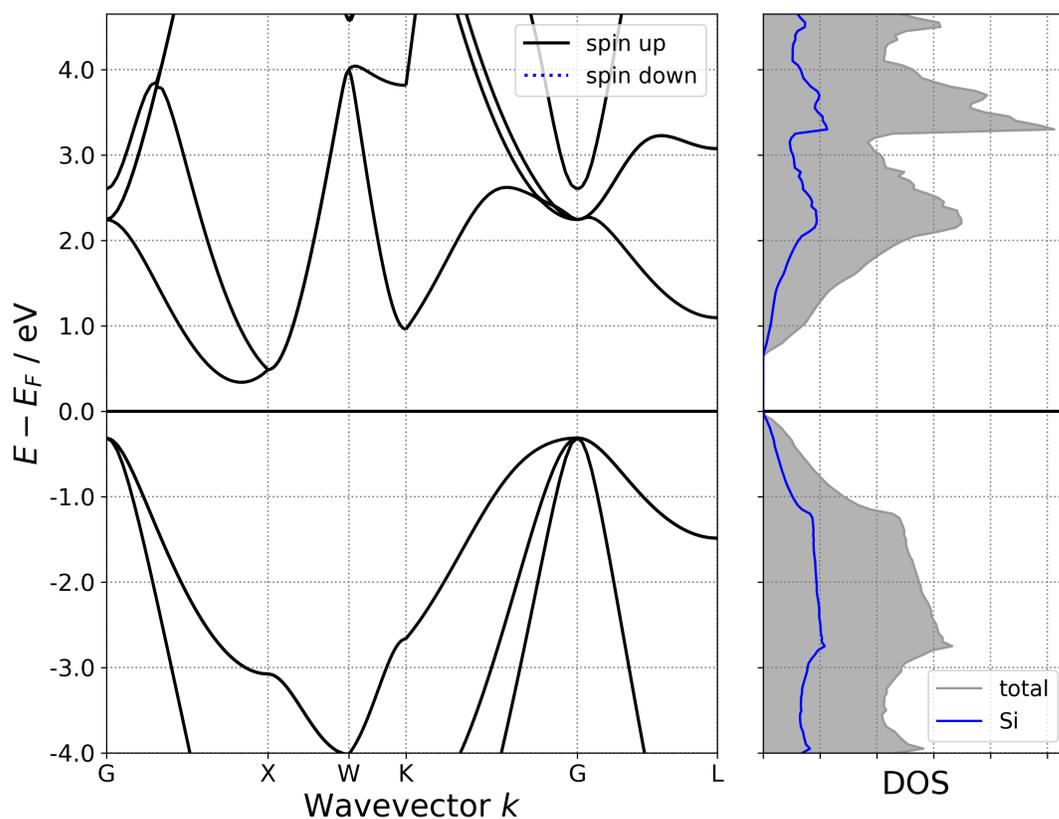
8  from dspawpy.io.read import get_band_data, get_dos_data
9
10 bandfile = "dspawpy_proj/dspawpy_tests/inputs/2.4/band.h5" # 普通能带数据
11 band_data = get_band_data(
12     band_dir=bandfile,
13     syst_dir=None, # system.json文件路径, 仅当band_dir为json文件时需要
14     efermi=None, # 用于手动修正费米能级
15 )
16 band_efermi = band_data.efermi
17 dosfile = "dspawpy_proj/dspawpy_tests/inputs/2.6/dos.h5" # 投影态密度数据
18 dos_data = get_dos_data(
19     dos_dir=dosfile,
20     return_dos=False, # 如果为False, 则统一返回CompleteDos对象 (无论计算时是否开了投影)
21 )
22 dos_efermi = dos_data.efermi
23 bdp = BSDOSPlotter(
24     bs_projection="elements", # 能带的投影方式, None表示不投影
25     dos_projection="elements", # 投影态密度到元素上
26     vb_energy_range=4, # 价带能量范围
27     cb_energy_range=4, # 导带能量范围
28     fixed_cb_energy=False, # 是否固定导带能量范围
29     egrid_interval=1, # 能量格点间隔
30     font="DejaVu Sans", # 默认Times New Roman, 修改成 DejaVu_
    ↪ Sans, 可避免在linux上因字体未安装而出现的警告
31     axis_fontsize=20, # 坐标轴字体大小
32     tick_fontsize=15, # 刻度字体大小
33     legend_fontsize=14, # 图例字体大小
34     bs_legend="best", # 能带图图例位置
35     dos_legend="best", # 投影态密度图图例位置
36     rgb_legend=True, # 是否使用彩色图例
37     fig_size=(11, 8.5), # 图片大小
38 )
39 if band_efermi != dos_efermi:
40     print(f"{band_efermi:.4f} eV")
41     print(f"{dos_efermi:.4f} eV")
42     d_efermi = band_efermi - dos_efermi
43
44     print("! Band Dos费米能级不一致, 将以Dos为准")
45     band_data.bands = {spin: v + d_efermi for spin, v in band_data.bands.items()}
46
47     # print("! Band Dos费米能级不一致, 将以Band为准")
48     # dos_data.energies -= d_efermi
49
50 axes_or_plt = bdp.get_plot(
51     bs=band_data, dos=dos_data,
52 ) # 传入能带数据 # 传入投影态密度数据
53
54 if isinstance(axes_or_plt, Axes):
55     fig = axes_or_plt.get_figure() # version newer than v2023.8.10
56 elif np.iterable(axes_or_plt):
57     fig = np.asarray(axes_or_plt).flatten()[0].get_figure()
58 else:
59     fig = axes_or_plt.gcf() # older version pymatgen
60
61 filename = "dspawpy_proj/dspawpy_tests/outputs/us/6bandPdos.png" # 输出的能带-
    ↪ 投影态密度图文件名
62 os.makedirs(os.path.dirname(os.path.abspath(filename)), exist_ok=True)

```

(续下页)

```
63 fig.savefig(filename, dpi=300)
64 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):

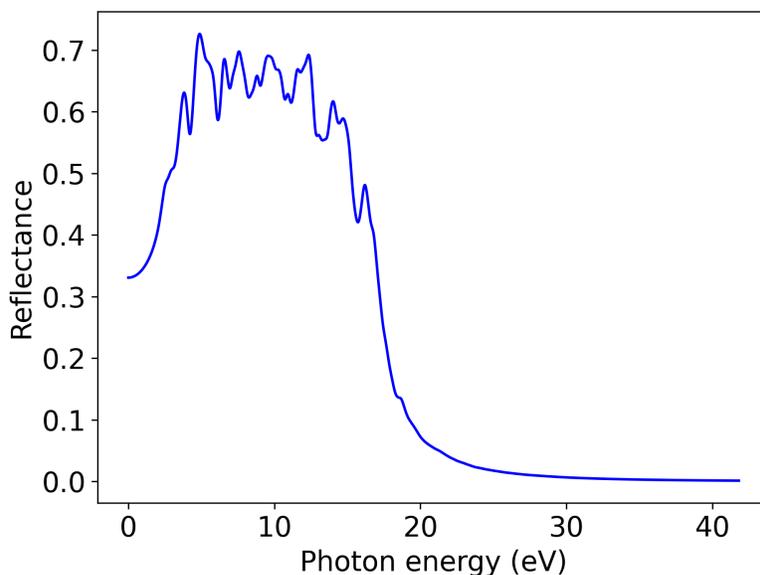
反射率数据处理, 参考 7optical.py :

```
1 # coding:utf-8
2 from dspawpy.plot import plot_optical
3
4 plot_optical(
5     datafile="dspawpy_proj/dspawpy_tests/inputs/2.12/scf.h5",
6     keys=["ExtinctionCoefficient", "Reflectance"],
7     axes=["X"], # ["X", "Y", "Z", "XY", "YZ", "ZX"]
8     prefix="dspawpy_proj/dspawpy_tests/outputs/optical", #
9     ↪ 保存到哪个文件夹中, 若为空, 表示当前文件夹
10    save=True, # 是否工具性质名称保存图片, 若为 False, 请参考下方脚本自行保存
11 )
12 # 以上函数将绘制并分别保存 ExtinctionCoefficient 和 Reflectance 的图像
13 # 要将多个性质绘制在同一张图中, 请取消注释以下代码, 并将上方 save 参数设置为 False
14 # import os
15 # import matplotlib.pyplot as plt
16 #
17 # plt.tick_params(labelsize=16)
18 # plt.tight_layout()
19 # filename = (
20 #     "dspawpy_proj/dspawpy_tests/outputs/us/7optical.png" # 输出的光学性质图文件名
21 # )
22 # os.makedirs(os.path.dirname(os.path.abspath(filename)), exist_ok=True)
23 # 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)
```

光学性质计算任务完成后，读取数据并绘制预览图

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

参数

- **datafile** -- h5 或 json 文件路径或包含任意这些文件的文件夹，默认'optical.h5'
- **keys** -- 可选"AbsorptionCoefficient"，"ExtinctionCoefficient"，"RefractiveIndex"，"Reflectance" 中的任意一个，默认"AbsorptionCoefficient"
- **axes** -- 序号，默认"X"，"Y"，"Z"，"XY"，"YZ"，"ZX"
- **raw** -- 是否保存绘图数据到 csv
- **prefix** -- 保存图片的文件夹路径，若留空则保存在当前目录
- **save** -- 是否保存图片，默认 True

示例

绘图并保存绘图数据到 rawoptical.csv

```
>>> from dspawpy.plot import plot_optical
>>> plot_optical("dspawpy_proj/dspawpy_tests/inputs/2.12/scf.h5", "AbsorptionCoefficient", [
↪ ['X', 'Y'], prefix='dspawpy_proj/dspawpy_tests/outputs/doctest')
>>> plot_optical("dspawpy_proj/dspawpy_tests/inputs/2.12/optical.json", [
↪ "AbsorptionCoefficient"], ['X', 'Y'], prefix='dspawpy_proj/dspawpy_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 :

```
1 # coding:utf-8
2 from dspawpy.diffusion.neb import NEB, write_neb_structures
3 from dspawpy.diffusion.nebtools import write_json_chain
4 from dspawpy.io.structure import read
5
6 # 读取初态构型
7 init_struct = read("dspawpy_proj/dspawpy_tests/inputs/2.15/00/structure00.as")[0]
8 # 读取末态构型
9 final_struct = read("dspawpy_proj/dspawpy_tests/inputs/2.15/04/structure04.as")[0]
10
11 neb = NEB(
12     initial_structure=init_struct, # 初态构型
13     final_structure=final_struct, # 末态构型
14     nimages=8, # 共8个构型，包括初末态
15 )
16 structures = neb.linear_interpolate() # 线性插值
17 # structures = neb.idpp_interpolate() # idpp插值
18
19 # 保存 as 结构文件到 dest 路径下
20 write_neb_structures(
21     structures=structures, # 插入插值后的构型链
22     coords_are_cartesian=True, # 是否以笛卡尔坐标保存
23     fmt="as", # 保存格式，支持'json', 'as', 'hzw', 'pdb', 'xyz', 'dump' 六种
24     path="dspawpy_proj/dspawpy_tests/outputs/us/8neb_interpolate_structures", # 保存路径
25     prefix="structure", # 文件名前缀
26 )
27
28 # 预览初插构型链
29 write_json_chain(
30     preview=True, # 是否为预览模式
31     directory="dspawpy_proj/dspawpy_tests/outputs/us/8neb_interpolate_structures", #
32     ↪NEB计算目录
33     step=-1, # 默认保存最后一个离子步（最新）的构型链
34     dst="dspawpy_proj/dspawpy_tests/outputs/us/8neb", # 保存路径
35 )
36 # write_xyz_chain(preview=True, # 是否为预览模式
37 #                 directory="dspawpy_proj/dspawpy_tests/outputs/us/8neb_interpolate_
38     ↪structures", # NEB计算目录
```

(续下页)

(接上页)

```

37 #                 step=-1, # 默认保存最后一个离子步（最新）的构型链
38 #                 dst='dspawpy_proj/dspawpy_tests/outputs/us/8neb' # 保存路径
39 # )

```

i 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，`pbic` 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。`pbic = True` 将强制设置扩散路径为 0.2 -> -0.2。`pbic = 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`：

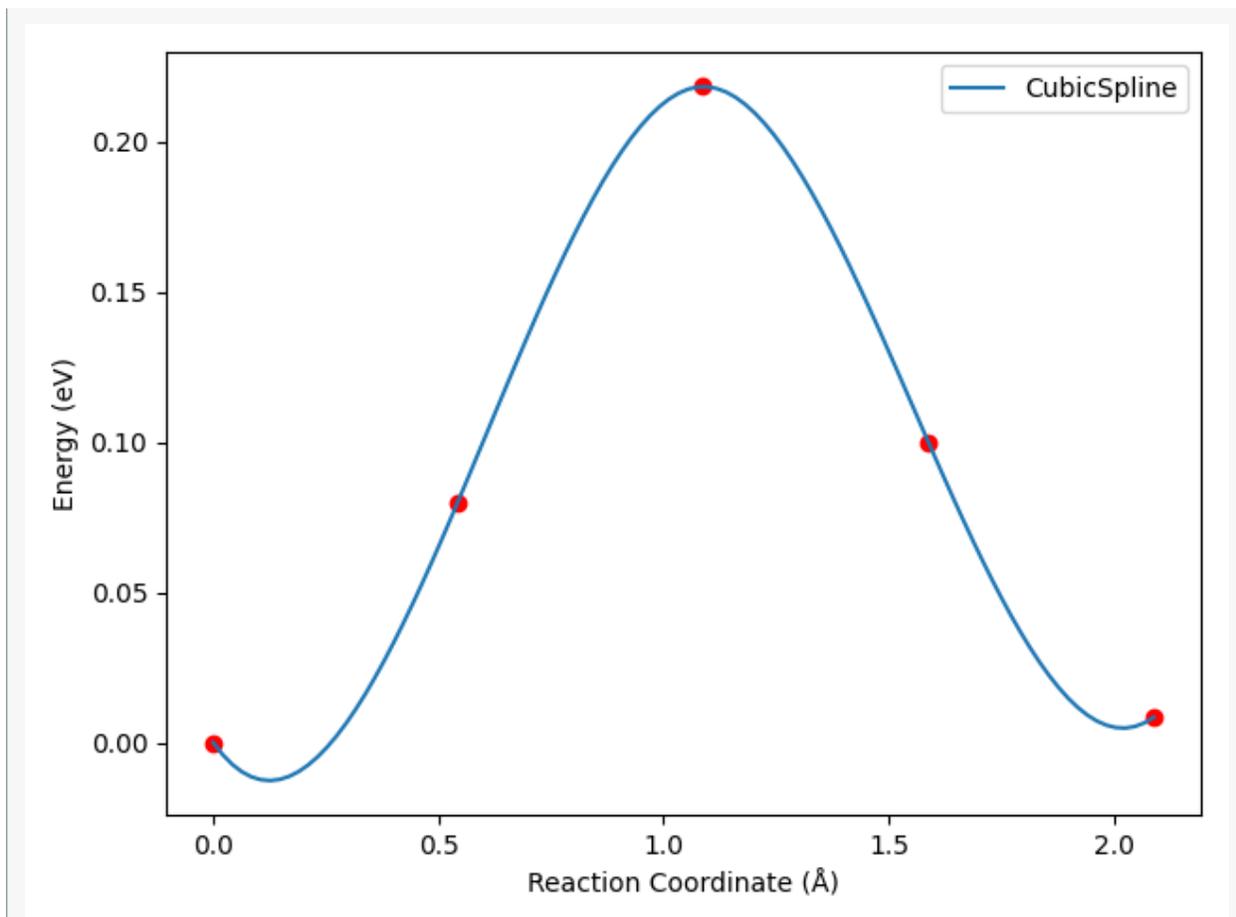
```

1 # coding:utf-8
2 from dspawpy.diffusion.nebtools import plot_barrier
3
4 directory_of_neb_task = (
5     "dspawpy_proj/dspawpy_tests/inputs/2.15" # <-- 请修改成实际NEB路径
6 )
7
8 # 三次样条插值 (CubicSpline) 绘制能垒图
9 plot_barrier(
10     directory=directory_of_neb_task, # neb任务的路径
11     method="CubicSpline", # 三次样条插值
12     figname="dspawpy_proj/dspawpy_tests/outputs/us/8neb_barrier_CubicSpline.png", # 输出的能垒图文件名
13     show=False, # 是否显示能垒图
14 )

```

i 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：

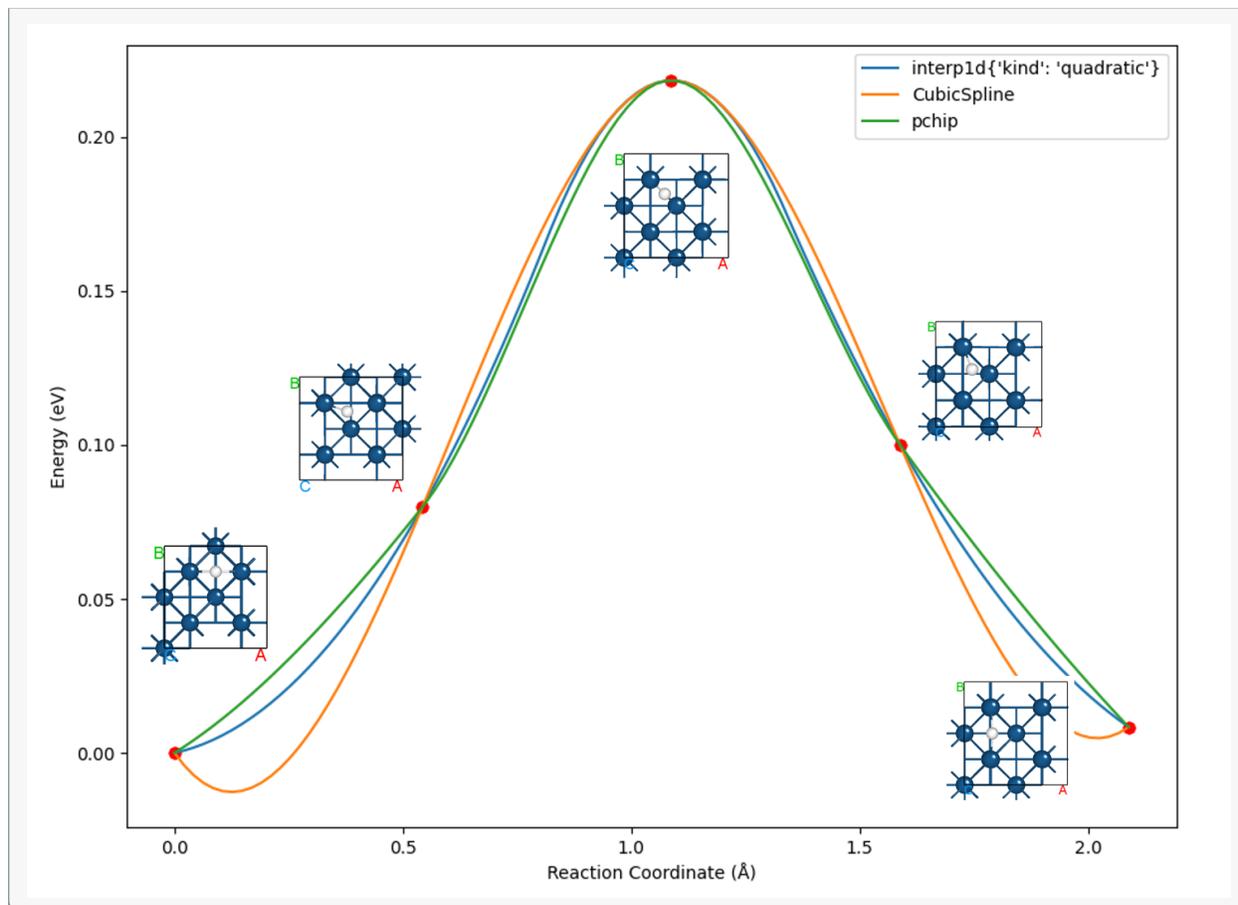
```
class scipy.interpolate.CubicSpline(x, y, axis=0, bc_type='not-a-knot', extrapolate=None)
```

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：

```
1 # coding:utf-8
2 import os
3
4 import matplotlib.pyplot as plt
5
6 from dspawpy.diffusion.nebtools import plot_barrier
7
8 # 对比不同插值方法绘制出的能垒曲线的区别，此时show应当设置为False
9 # 1. interp1d
10 plot_barrier(
11     directory="dspawpy_proj/dspawpy_tests/inputs/2.15", # neb计算的路径
12     ri=None, # 初始构型与第二个构型之间的反应坐标，当NEB任务只计算了中间构型时需要
```

```
13     rf=None, # 最后一个构型与倒数第二个构型之间的反应坐标, 当NEB任务只计算了中间构型时需要
14     ei=None, # 初始构型的能量, 当NEB任务只计算了中间构型时需要
15     ef=None, # 最后一个构型的能量, 当NEB任务只计算了中间构型时需要
16     method="interp1d", # 插值方法
17     figname=None, # 输出的能垒图文件名
18     show=False, # 是否显示能垒图
19     kind="quadratic", # 插值方法的参数
20 )
21 # 2. CubicSpline
22 plot_barrier(
23     directory="dspawpy_proj/dspawpy_tests/inputs/2.15",
24     method="CubicSpline",
25     figname=None,
26     show=False,
27 )
28 # 3. pchip
29 plot_barrier(
30     directory="dspawpy_proj/dspawpy_tests/inputs/2.15",
31     method="pchip",
32     figname=None,
33     show=False,
34 )
35
36 filename = "dspawpy_proj/dspawpy_tests/outputs/us/8neb_barrier_comparison.png" #_
37 ↪输出的能垒图文件名
38 os.makedirs(os.path.dirname(os.path.abspath(filename)), exist_ok=True)
39 plt.savefig(filename, dpi=300)
40 # 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` :

```

1  # coding:utf-8
2  from dspawpy.diffusion.nebtools import plot_barrier
3
4  # 三次样条插值 (CubicSpline) 绘制能垒图
5  plot_barrier(
6      datafile="dspawpy_proj/dspawpy_tests/inputs/2.15/neb.h5", # neb.h5的路径
7      method="CubicSpline", # 三次样条插值
8      figname="dspawpy_proj/dspawpy_tests/outputs/us/8neb_barrier_.png", #
9      ↪输出的能垒图文件名
10     show=False, # 是否显示能垒图
11 )

```

处理得到的势垒图与之前读取路径效果一致。

i 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）

A 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：

```
1 # coding:utf-8
2 from dspawpy.diffusion.nebtools import summary
3
4 # 导入neb计算目录，需提供neb计算完成后的完整文件夹
5 summary(
6     directory="dspawpy_proj/dspawpy_tests/inputs/2.15",
7     show_converge=False, # 是否显示能量和受力收敛过程图
8     outdir="dspawpy_proj/dspawpy_tests/outputs/us/8neb", # 能量和受力收敛过程图保存路径
9     figname="dspawpy_proj/dspawpy_tests/outputs/us/8neb/neb_barrier_summary.png", #
10    ↪ 能垒图保存路径
11 )
12 # 可以进一步设置其他关键字参数，用于绘制能垒图，例如：
13 # summary(directory = 'dspawpy_proj/dspawpy_tests/inputs/2.15', method='CubicSpline') #
14    ↪ 改为三次样条插值CubicSpline
```

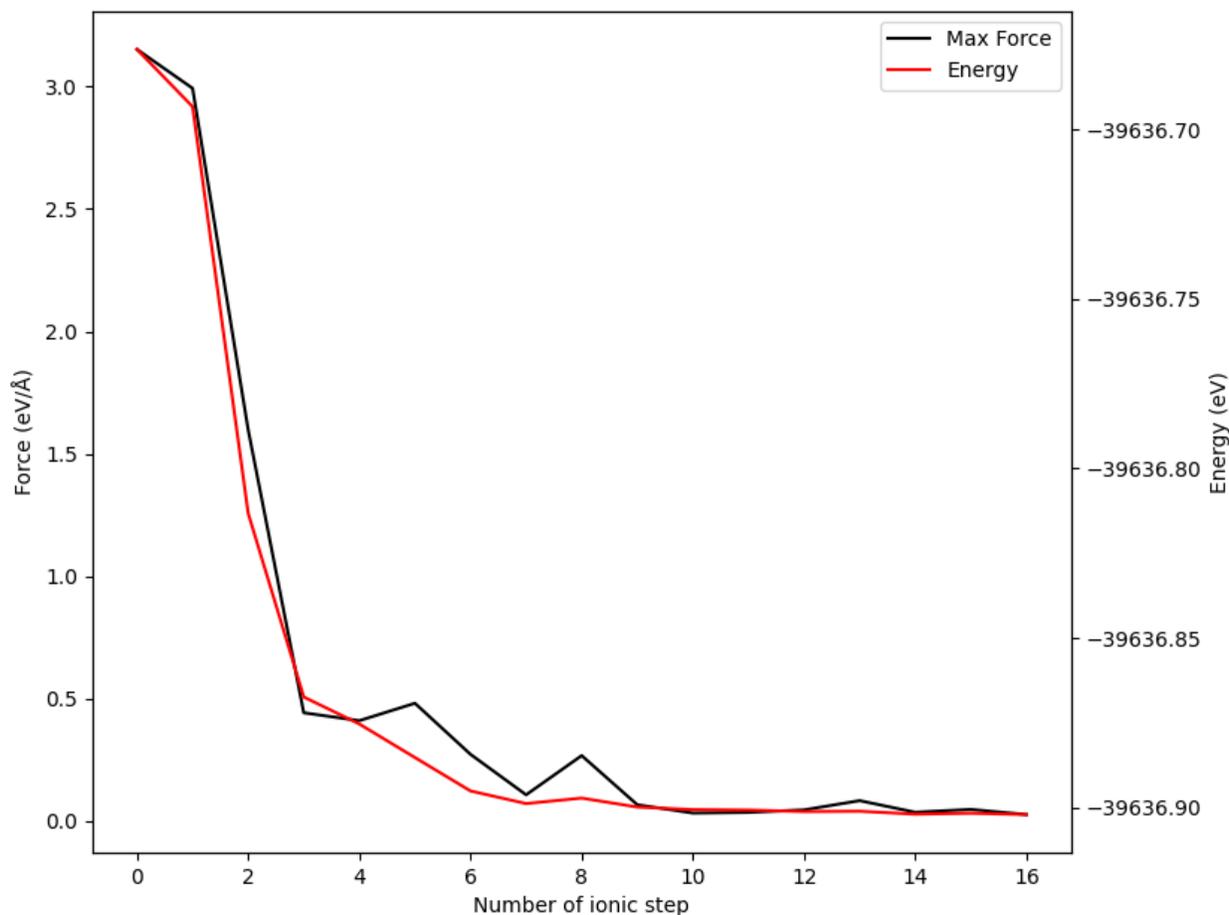
i 知识点：

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:

```

1 # coding:utf-8
2 from dspawpy.diffusion.nebtools import write_json_chain, write_xyz_chain
3
4 # 将NEB计算路径下的构型链转成json格式文件
5 write_json_chain(
6     preview=False, # NEB计算如果已经完成, 则可以不是预览模式
7     directory="dspawpy_proj/dspawpy_tests/inputs/2.15", # NEB计算目录
8     step=-1, # 默认保存最后一个离子步(最新)的构型链
9     dst="dspawpy_proj/dspawpy_tests/outputs/us/8neb", # 保存路径
10    ignorels=False, # 设为True则忽略 latestStructureXX.as 文件
11)
12
13 # 将NEB计算路径下的构型链转成xyz格式文件
14 write_xyz_chain(
15     preview=False, # NEB计算如果已经完成, 则可以不是预览模式
16     directory="dspawpy_proj/dspawpy_tests/inputs/2.15", # NEB计算目录
17     step=-1, # 默认保存最后一个离子步(最新)的构型链
18     dst="dspawpy_proj/dspawpy_tests/outputs/us/8neb", # 保存路径
19     ignorels=False, # 设为True则忽略 latestStructureXX.as 文件
20)

```

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:

```

1 # coding:utf-8
2 from dspawpy.diffusion.nebtools import get_distance
3 from dspawpy.io.structure import read
4
5 # 请根据实际情况修改 structure01.as 和 structure02.as 结构文件的路径
6 # 先读取两个构型的分数坐标、元素列表和晶胞信息
7 s1 = read("dspawpy_proj/dspawpy_tests/inputs/2.15/01/structure01.as")[0]
8 s2 = read("dspawpy_proj/dspawpy_tests/inputs/2.15/02/structure02.as")[0]
9 # 计算两个构型的距离, 注意这个函数只接受分数坐标

```

(续下页)

(接上页)

```

10 dist = get_distance(
11     spo1=s1.frac_coords,
12     spo2=s2.frac_coords,
13     lat1=s1.lattice.matrix,
14     lat2=s2.lattice.matrix,
15 )
16 print("两个构型的距离为: ", dist, "Angstrom")

```

8.8.6 neb 续算

- 如果需对 neb 进行续算，可参考 8neb_restart.py：

```

1 # coding:utf-8
2 import os
3 from shutil import copytree, rmtree
4
5 from dspawpy.diffusion.nebtools import restart
6
7 if os.path.isdir("dspawpy_proj/dspawpy_tests/outputs/us/neb4bk"):
8     rmtree("dspawpy_proj/dspawpy_tests/outputs/us/neb4bk")
9
10 copytree(
11     "dspawpy_proj/dspawpy_tests/inputs/2.15",
12     "dspawpy_proj/dspawpy_tests/outputs/us/neb4bk",
13 )
14 restart(
15     directory="dspawpy_proj/dspawpy_tests/outputs/us/neb4bk", # NEB任务路径
16     output="dspawpy_proj/dspawpy_tests/outputs/us/8neb_restart", # 备份到哪
17 )

```

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

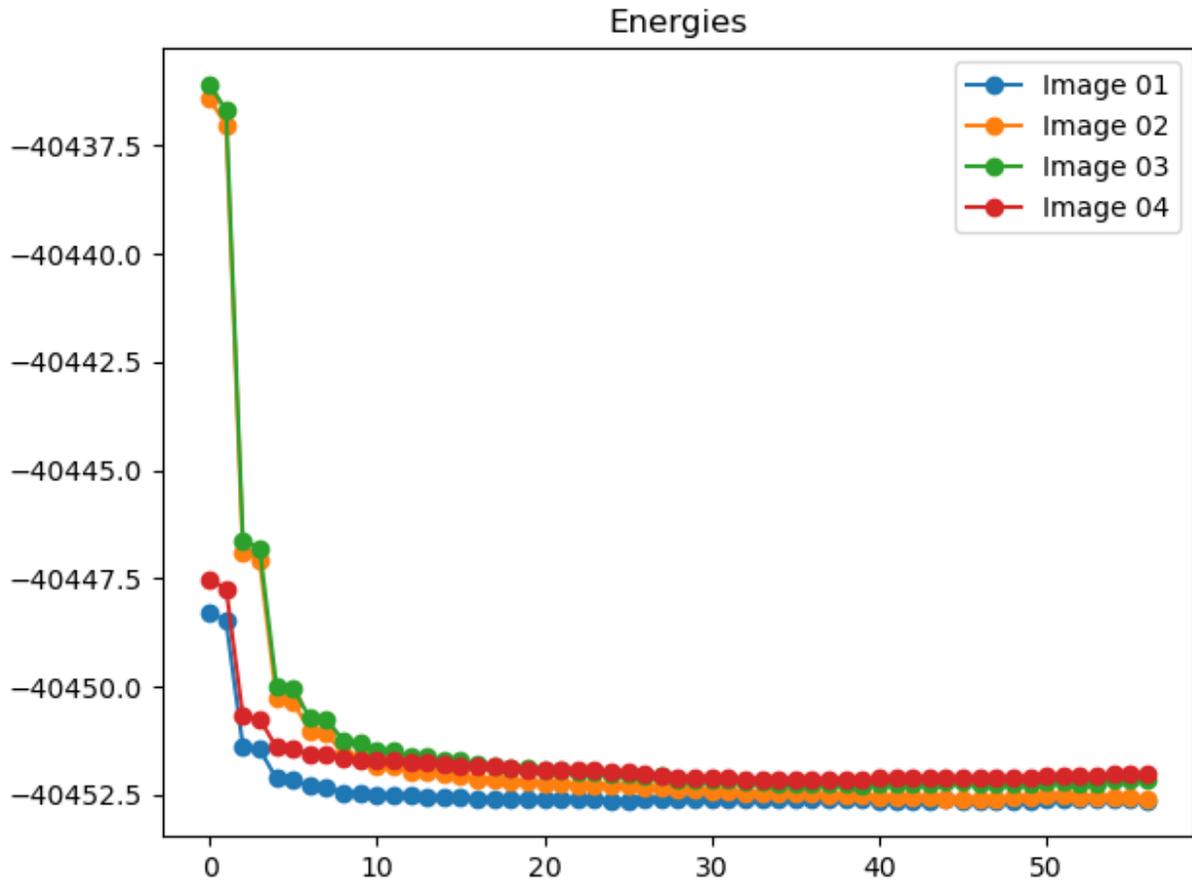
- 要查看 neb 计算过程中能量和最大原子受力的变化趋势图，可参考 8neb_energy_force_curves.py：

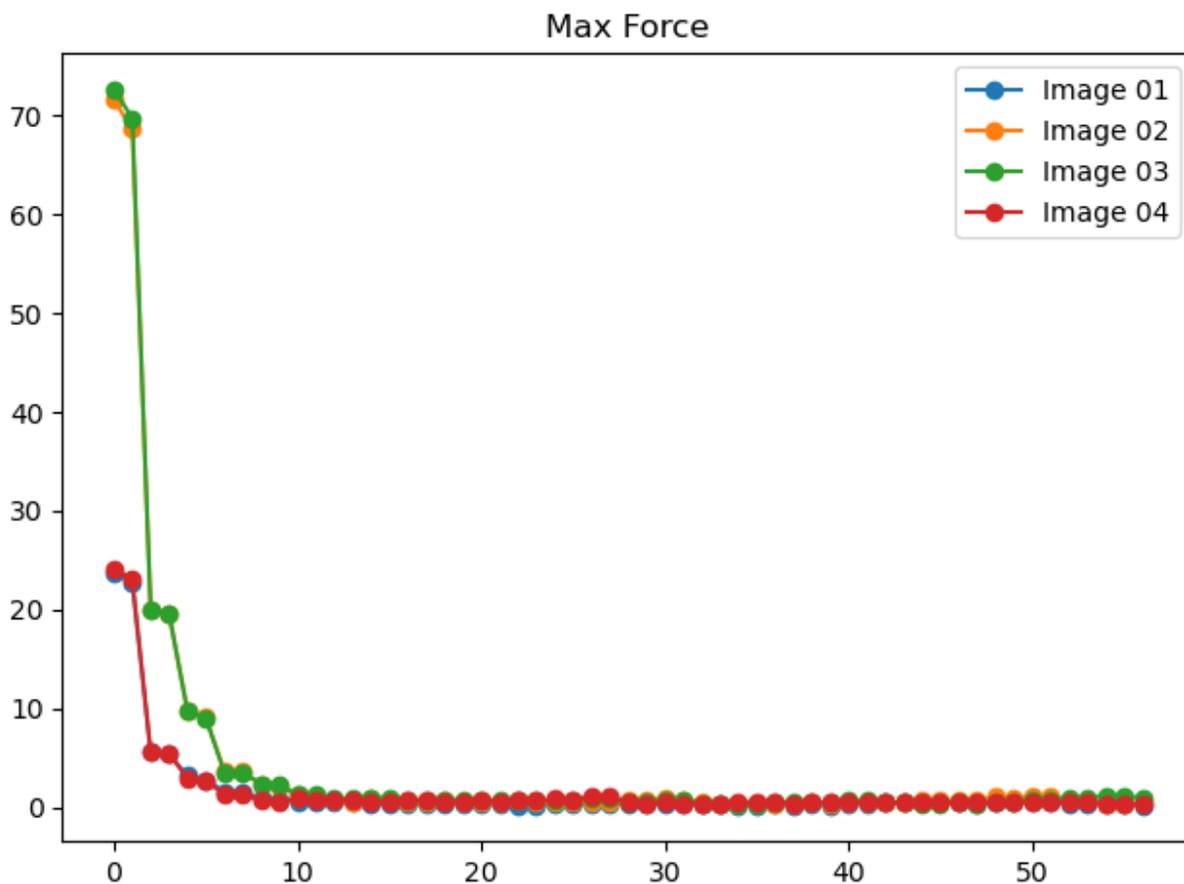
```

1 # coding:utf-8
2 from dspawpy.diffusion.nebtools import monitor_force_energy
3
4 # 指定NEB计算文件夹路径，运行后会在指定目录生成 Energies.png 和 MaxForce.png 图片
5 unfinished_neb_folder = "dspawpy_proj/dspawpy_tests/inputs/supplement/neb_unfinished"
6 monitor_force_energy(
7     directory=unfinished_neb_folder,
8     outdir="imgs", # 输出图片路径
9 )

```

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')
```

插值并生成中间构型文件

参数

- `structures` -- 构型列表
- `coords_are_cartesian` -- 坐标是否为笛卡尔坐标
- `fmt` -- 结构文件类型，默认为"as"
- `path` -- 保存路径
- `prefix` -- 文件名前缀，默认为"structure"，这样的话生成的就是 structure00.as, structure01.as, ...

返回

保存构型文件

返回类型

file

示例

先读取 as 文件创建 structure 对象

```
>>> from dspawpy.io.structure import read
>>> init_struct = read("dspawpy_proj/dspawpy_tests/inputs/2.15/00/structure00.as")[0]
>>> final_struct = read("dspawpy_proj/dspawpy_tests/inputs/2.15/04/structure04.as")[0]
```

然后, 插值并生成中间构型文件

```
>>> from dspawpy.diffusion.neb import NEB, write_neb_structures
>>> neb = NEB(init_struct, final_struct, 8)
>>> structures = neb.linear_interpolate() #线性插值
```

插值完成的构型可指定保存到 neb 文件夹下

```
>>> write_neb_structures(structures, path="dspawpy_proj/dspawpy_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)
```

调用 `scipy.interpolate` 插值算法, 拟合 NEB 能垒并绘图

参数

- `datafile` -- neb.h5 或 neb.json 文件路径
- `directory` -- NEB 计算路径
- `ri` -- 初态反应坐标
- `rf` -- 末态反应坐标
- `ei` -- 初态自洽能量
- `ef` -- 末态自洽能量
- `method` (`str`, `optional`) -- 插值算法, 默认 'PchipInterpolator'
- `figname` (`str`, `optional`) -- 能垒图名称, 默认 'neb_barrier.png'
- `show` (`bool`, `optional`) -- 是否展示交互界面, 默认 True
- `raw` (`bool`, `optional`) -- 是否返回绘图数据到 csv

抛出

- `ImportError` -- 指定了 `scipy.interpolate` 中不存在的插值算法
- `ValueError` -- 传递给插值算法的参数不符合该算法要求

示例

```
>>> from dspawpy.diffusion.nebtools import plot_barrier
>>> import matplotlib.pyplot as plt
```

对比不同插值算法

```
>>> plot_barrier(directory='dspawpy_proj/dspawpy_tests/inputs/2.15', method='interp1d',
↳ kind=2, figname=None, show=False)
>>> plot_barrier(directory='dspawpy_proj/dspawpy_tests/inputs/2.15', method='interp1d',
↳ kind=3, figname=None, show=False)
>>> plot_barrier(directory='dspawpy_proj/dspawpy_tests/inputs/2.15', method='
↳ CubicSpline', figname=None, show=False)
>>> plot_barrier(directory='dspawpy_proj/dspawpy_tests/inputs/2.15', method='pchip',
↳ figname='dspawpy_proj/dspawpy_tests/outputs/doctest/barrier_comparison.png',
↳ show=False)
==> ...barrier_comparison.png...
```

尝试读取 neb.h5 文件或 neb.json 文件

```
>>> plot_barrier(datafile='dspawpy_proj/dspawpy_tests/inputs/2.15/neb.h5', method=
↳ 'pchip', figname='dspawpy_proj/dspawpy_tests/outputs/doctest/barrier_h5.png',
↳ show=False)
==> ...barrier_h5.png
>>> plot_barrier(datafile='dspawpy_proj/dspawpy_tests/inputs/2.15/neb.json', method=
↳ 'pchip', figname='dspawpy_proj/dspawpy_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)
```

NEB 任务完成总结，依次执行以下步骤：

1. 打印各构型受力、反应坐标、能量、与初始构型的能量差
2. 绘制能垒图
3. 绘制并保存结构优化过程的能量和受力收敛过程图

参数

- **directory** -- NEB 路径, 默认当前路径
- **raw** -- 是否保存绘图数据到 csv 文件
- **show_converge** -- 是否展示结构优化过程的能量和受力收敛过程图, 默认不展示
- **outdir** -- 收敛过程图保存路径, 默认为 **directory**
- ****kwargs** (*dict*) -- 传递给 `plot_barrier` 的参数

示例

```
>>> from dspawpy.diffusion.nebtools import summary
>>> directory = 'dspawpy_proj/dspawpy_tests/inputs/2.15' # NEB计算路径, 默认当前路径
>>> summary(directory, show=False, figname='dspawpy_proj/dspawpy_tests/outputs/
↳ doctest/neb_barrier.png')
shape: (5, 5)
```

(续下页)

(接上页)

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='dspawpy_proj/dspawpy_tests/outputs/
↳doctest/neb_barrier.png', outdir="dspawpy_proj/dspawpy_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...

```

若 `inifin=false`, 用户必须将自洽的 `scf.h5` 或 `system.json` 放到初末态子文件夹中

- `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)
```

根据两个结构的分数坐标和晶胞计算距离

参数

- `spo1` (`np.ndarray`) -- 分数坐标列表 1
- `spo2` (`np.ndarray`) -- 分数坐标列表 2
- `lat1` (`np.ndarray`) -- 晶胞 1
- `lat2` (`np.ndarray`) -- 晶胞 2

返回

距离

返回类型

float

示例

先读取结构信息

```
>>> from dspawpy.io.structure import read
>>> s1 = read('dspawpy_proj/dspawpy_tests/inputs/2.15/01/structure01.as')[0]
>>> s2 = read('dspawpy_proj/dspawpy_tests/inputs/2.15/02/structure02.as')[0]
```

计算两个构型的距离

```
>>> from dspawpy.diffusion.nebtools import get_distance
>>> dist = get_distance(s1.frac_coords, s2.frac_coords, s1.lattice.matrix, s2.lattice.
↳matrix)
>>> print('两个构型的距离为: ', dist, 'Angstrom')
两个构型的距离为: 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')
```

将旧 NEB 任务归档压缩, 并在原路径下准备续算

参数

- **directory** -- 旧 NEB 任务所在路径, 默认当前路径
- **output** -- 备份文件夹路径, 默认将在当前路径新建一个 bakfile 文件夹用于备份; 也可以任意指定一个路径, 但不能与当前路径相同

示例

```
>>> from dspawpy.diffusion.nebtools import restart
>>> from shutil import copytree
>>> copytree('dspawpy_proj/dspawpy_tests/inputs/2.15', 'dspawpy_proj/dspawpy_tests/
↳outputs/doctest/neb4bk2', dirs_exist_ok=True)
'dspawpy_proj/dspawpy_tests/outputs/doctest/neb4bk2'
>>> restart(directory='dspawpy_proj/dspawpy_tests/outputs/doctest/neb4bk2', output=
↳'dspawpy_proj/dspawpy_tests/outputs/doctest/neb_backup')
==> ...neb_backup...
```

续算准备工作可能需要较长时间才能完成, 请耐心等待

- 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)
```

从 xx/DS-PAW.log 中读取 NEB 计算过程中的受力和能量, 绘制曲线

计算过程中无 json 文件输出, nebXX.h5 文件中只有受力信息, 因此只能读取 DS-PAW.log

示例

```
>>> from dspawpy.diffusion.nebtools import monitor_force_energy
>>> monitor_force_energy(
...     directory="dspawpy_proj/dspawpy_tests/inputs/supplement/neb_unfinished",
...     outdir="imgs"
... )
Max Force shape: (57, 4)
```

(续下页)

(接上页)

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* :

```

1 # coding:utf-8
2 import os
3
4 from pymatgen.phonon.plotter import PhononBSPlotter
5
6 from dspawpy.io.read import get_phonon_band_data
7
8 band_data = get_phonon_band_data(
9     "dspawpy_proj/dspawpy_tests/inputs/2.16.1/phonon.h5",
10 ) # 读取声子能带

```

(续下页)

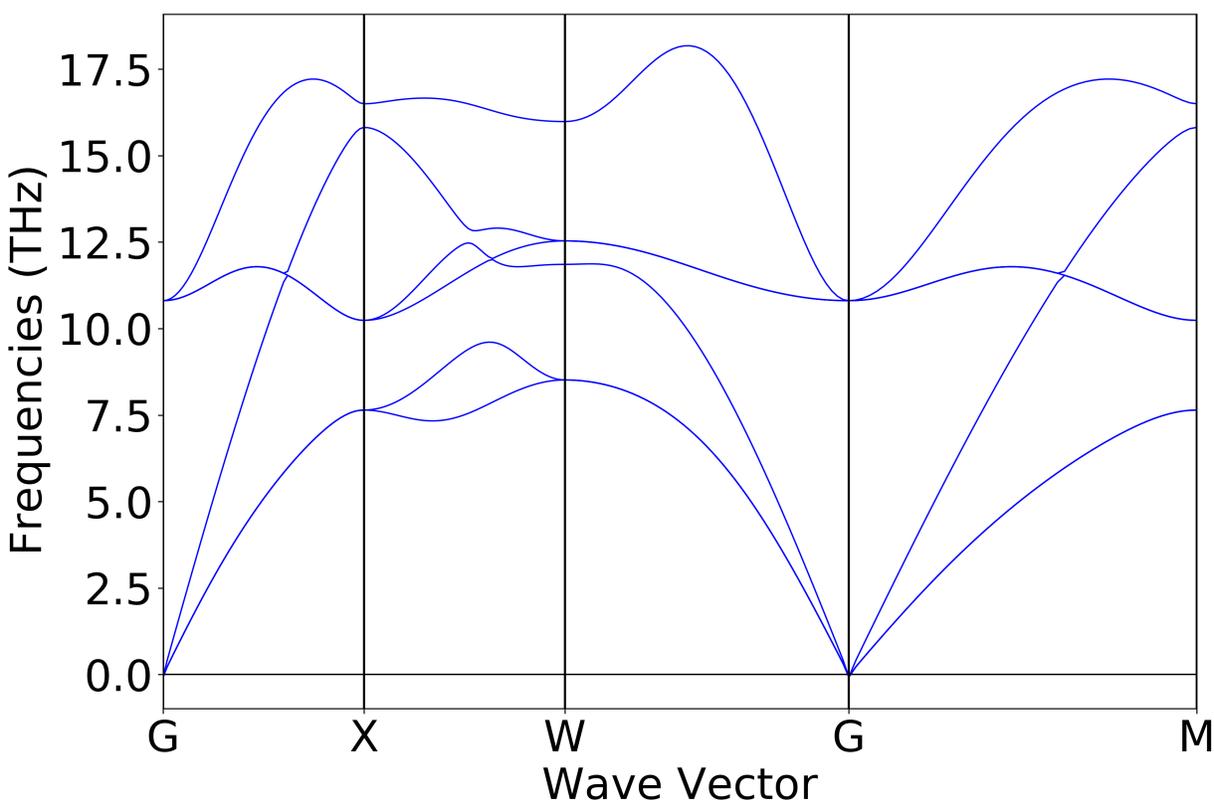
(接上页)

```

11 bsp = PhononBSPlotter(band_data)
12 axes_or_plt = bsp.get_plot(ylim=None, units="thz") # y轴范围 # 单位
13 import matplotlib.pyplot as plt # noqa: E402
14
15 if isinstance(axes_or_plt, plt.Axes):
16     fig = axes_or_plt.get_figure() # version newer than v2023.8.10
17 elif isinstance(axes_or_plt, tuple):
18     fig = axes_or_plt[0].get_figure()
19 else:
20     fig = axes_or_plt.gcf() # older version pymatgen
21
22 filename = "dspawpy_proj/dspawpy_tests/outputs/us/9phonon_bandplot.png" #_
    ↪输出的声子能带图文件名
23 os.makedirs(os.path.dirname(os.path.abspath(filename)), exist_ok=True)
24 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

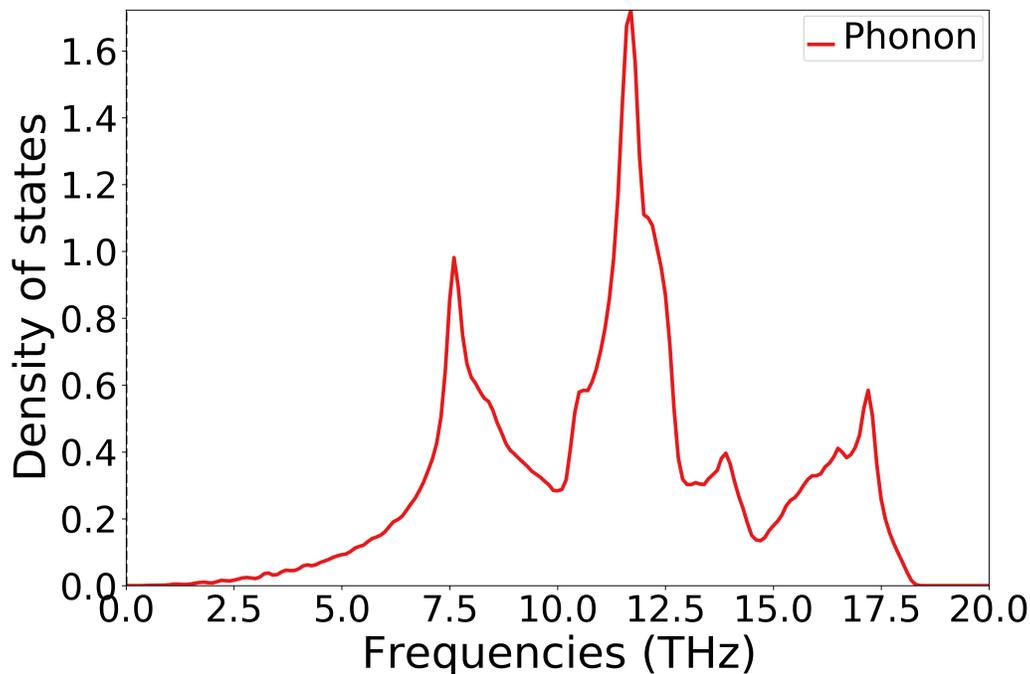
- 参考 9phonon_dosplot.py :

```

1  # coding:utf-8
2  import os
3
4  from pymatgen.phonon.plotter import PhononDosPlotter
5
6  from dspawpy.io.read import get_phonon_dos_data
7
8  dos = get_phonon_dos_data("dspawpy_proj/dspawpy_tests/inputs/2.16.1/phonon.h5")
9  dp = PhononDosPlotter(
10     stack=False, # True表示绘制面积图
11     sigma=None, # 高斯模糊参数
12 )
13 dp.add_dos(label="Phonon", dos=dos) # 图例 # 要绘制的声子态密度
14 axes_or_plt = dp.get_plot(
15     xlim=[0, 20], # x轴范围
16     ylim=None, # y轴范围
17     units="thz", # 单位
18 )
19 import matplotlib.pyplot as plt # noqa: E402
20
21 if isinstance(axes_or_plt, plt.Axes):
22     fig = axes_or_plt.get_figure() # version newer than v2023.8.10
23 elif isinstance(axes_or_plt, tuple):
24     fig = axes_or_plt[0].get_figure()
25 else:
26     fig = axes_or_plt.gcf() # older version pymatgen
27
28 filename = (
29     "dspawpy_proj/dspawpy_tests/outputs/us/9phonon_dosplot.png" # 输出的能垒图文件名
30 )
31 os.makedirs(os.path.dirname(os.path.abspath(filename)), exist_ok=True)
32 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：

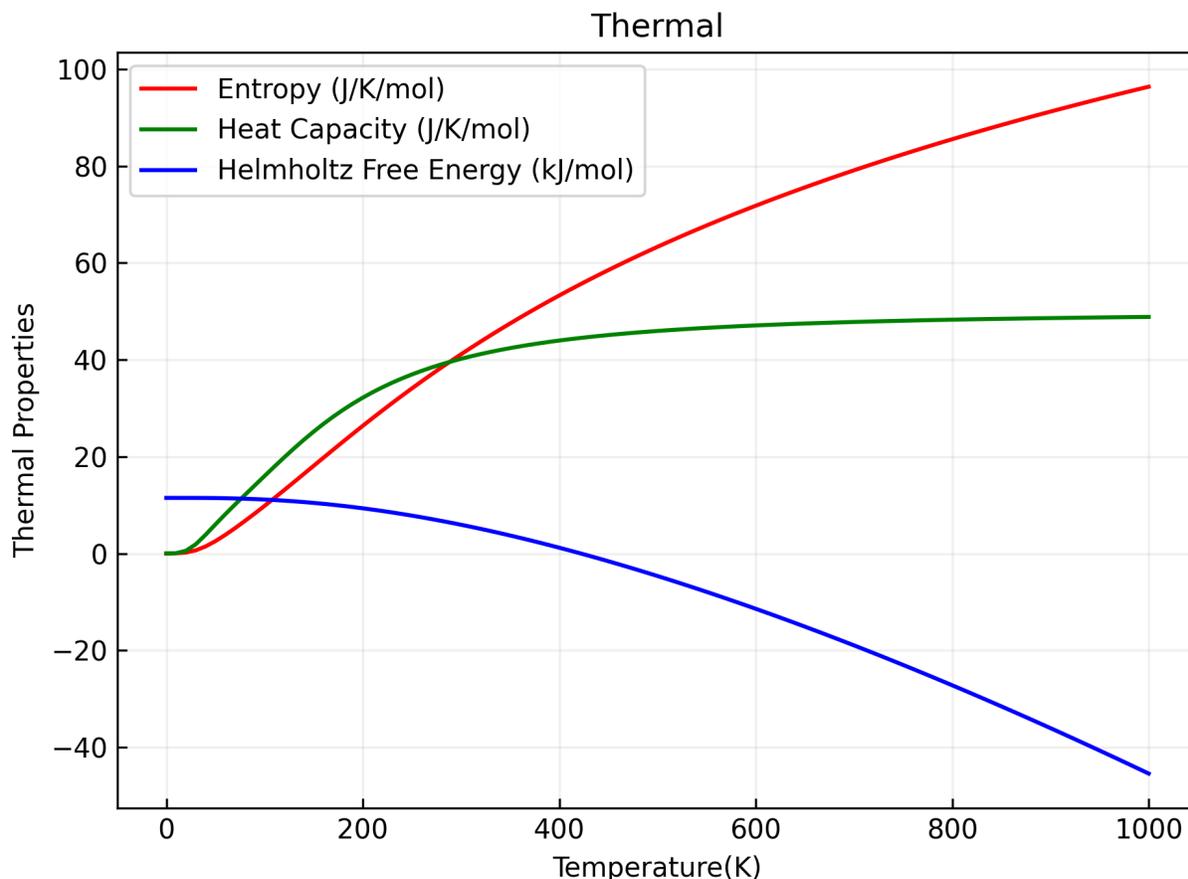
```
import matplotlib
matplotlib.use('agg')
```

8.9.3 声子热力学数据处理

可以参考 `9phonon_thermal.py`：

```
1 # coding:utf-8
2 from dspawpy.plot import plot_phonon_thermal
3
4 plot_phonon_thermal(
5     datafile="dspawpy_proj/dspawpy_tests/inputs/2.26/phonon.h5", # phonon.h5数据文件路径
6     figname="dspawpy_proj/dspawpy_tests/outputs/us/9phonon.png", # 输出的声子热力学图文件名
7     show=False, # 是否显示图片
8 )
```

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)`

读取 h5 或 json 文件中的声子能带数据, 构建 `PhononBandStructureSymmLine` 对象

参数

`phonon_band_dir` -- 能带文件路径, `phonon.h5` / `phonon.json` 或包含这两个文件的文件夹

返回类型

`PhononBandStructureSymmLine`

示例

```
>>> from dspawpy.io.read import get_phonon_band_data
>>> band_data = get_phonon_band_data("dspawpy_proj/dspawpy_tests/inputs/2.16/phonon.h5
↳") # 读取声子能带
>>> band_data = get_phonon_band_data("dspawpy_proj/dspawpy_tests/inputs/2.16/phonon.
↳json") # 读取声子能带
```

- `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)`

读取 h5 或 json 文件中的声子态密度数据，构建 PhononDos 对象

参数

phonon_dos_dir -- 声子态密度文件路径，phonon_dos.h5 / phonon_dos.json 或包含这两个文件的文件夹

返回类型

PhononDos

示例

```
>>> from dspawpy.io.read import get_phonon_dos_data
>>> phdos = get_phonon_dos_data(phonon_dos_dir='dspawpy_proj/dspawpy_tests/inputs/2.
↳16.1/phonon.json')
>>> phdos = get_phonon_dos_data(phonon_dos_dir='dspawpy_proj/dspawpy_tests/inputs/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,
        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)`

声子热力学计算任务完成后，绘制相关物理量随温度变化曲线

phonon.h5/phonon.json -> phonon.png

参数

- **datafile** -- h5 或 json 文件路径或包含任意这些文件的文件夹，默认'phonon.h5'
- **figname** -- 保存图片的文件名
- **show** -- 是否弹出交互界面
- **raw** -- 是否保存绘图数据到 rawphonon.csv 文件

返回

图片路径，默认'phonon.png'

返回类型

figname

示例

```
>>> from dspawpy.plot import plot_phonon_thermal
>>> plot_phonon_thermal('dspawpy_proj/dspawpy_tests/inputs/2.26/phonon.h5', figname=
↳ 'dspawpy_proj/dspawpy_tests/outputs/doctest/phonon_thermal_h5.png', show=False)
>>> plot_phonon_thermal('dspawpy_proj/dspawpy_tests/inputs/2.26/phonon.json', figname=
↳ 'dspawpy_proj/dspawpy_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：

```
1 # coding:utf-8
2 from dspawpy.io.structure import convert
3
4 convert (
5     infile="dspawpy_proj/dspawpy_tests/inputs/2.18/aimd.h5", #
↳ 待转化结构，如果在当前路径，可以只写文件名
6     si=None, # 筛选构型编号，如果不指定，默认读取全部
7     ele=None, # 筛选元素符号，默认读取所有元素的原子信息
8     ai=None, # 筛选原子编号，从 1 开始，默认读取所有原子信息
9     outfile="dspawpy_proj/dspawpy_tests/outputs/us/10aimdTraj.xyz", # 也可以生成 .dump
↳ 文件（精度较低），暂时只支持正交晶胞
10 )
```

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：

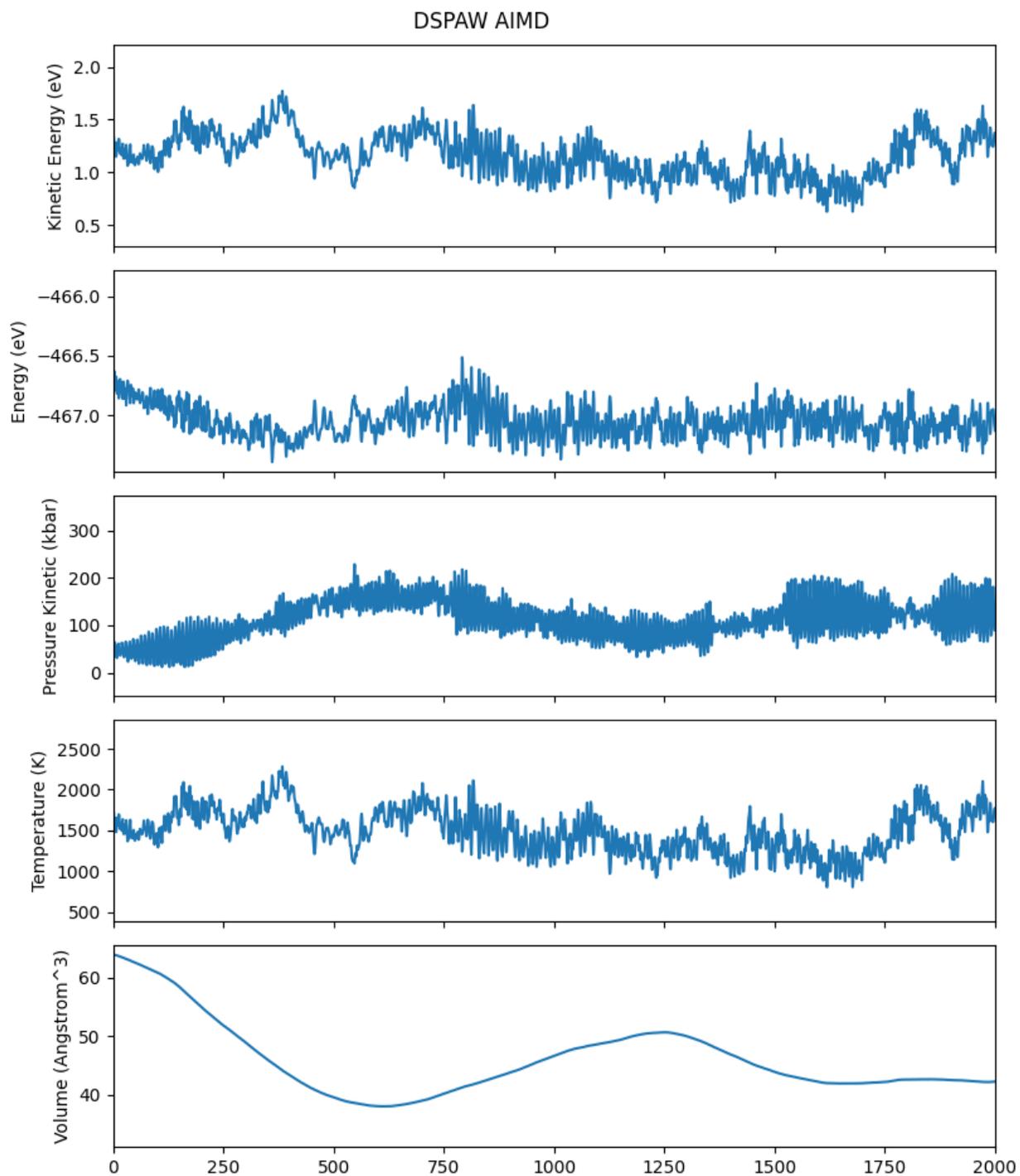
OVITO 与 dspawpy 都不支持将非正交晶胞的体系保存为 dump 文件

8.10.2 动力学过程中能量、温度等变化曲线

- 参考 10check_aimd_conv.py :

```
1 # coding:utf-8
2 from dspawpy.plot import plot_aimd
3
4 plot_aimd(
5     datafile="dspawpy_proj/dspawpy_tests/inputs/2.18/aimd.h5", # 数据文件路径
6     show=False, # 是否弹出图像窗口
7     figname="dspawpy_proj/dspawpy_tests/outputs/us/10aimd.png", # 输出的图像文件名
8     flags_str="1 2 3 4 5", # 选择数据类型
9 )
10 # 其中 flags_str 的含义如下
11 # 1. 动能
12 # 2. 总能
13 # 3. 压力
14 # 4. 温度
15 # 5. 体积
```

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：

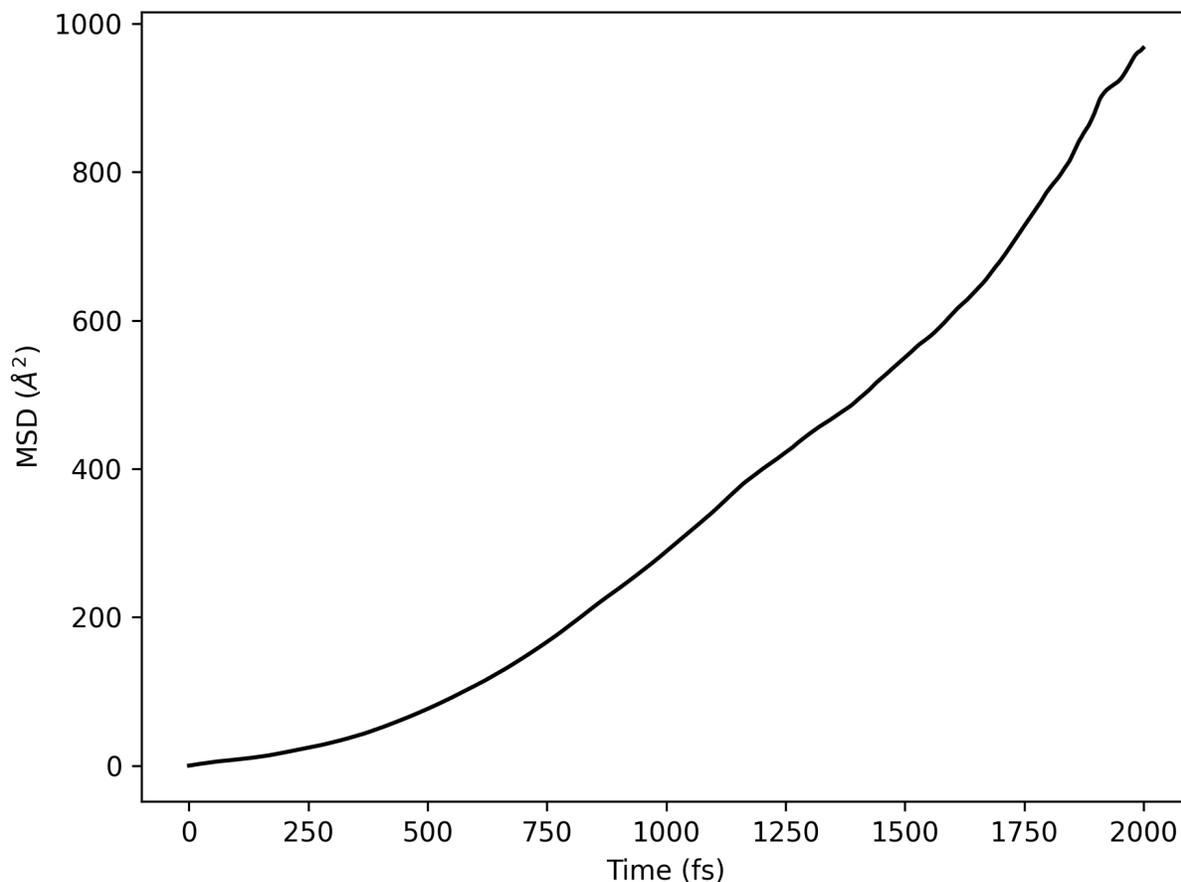
```
import matplotlib
matplotlib.use('agg')
```

8.10.3 均方位移 (MSD) 分析

- 参考 10aimd_msd.py :

```
1 # coding:utf-8
2 from dspawpy.analysis.aimdtools import get_lagtime_msd, plot_msd
3
4 # 如果AIMD非一次性完成, 可以将多个h5文件路径以列表形式赋值给datafile参数
5 lagtime, msd = get_lagtime_msd(
6     datafile="dspawpy_proj/dspawpy_tests/inputs/2.18/aimd.h5", # 数据文件路径
7     select="all", # 默认选择所有原子
8     msd_type="xyz", # 默认计算xyz方向的msd
9     timestep=None, # 默认从datafile文件中读取时间步长
10 )
11 # 用获取的数据画图并保存
12 plot_msd(
13     lagtime, # 横坐标
14     msd, # 纵坐标
15     xlim=None, # 设置x轴的显示范围
16     ylim=None, # 设置y轴的显示范围
17     figname="dspawpy_proj/dspawpy_tests/outputs/us/10MSD.png", # 输出的图像文件名
18     show=False, # 是否弹出图像窗口
19     # ax = None, # 可指定子图
20 )
```

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 :

```
1 # coding:utf-8
2 from dspawpy.analysis.aimdtools import get_lagtime_rmsd, plot_rmsd
3
4 # 如果AIMD非一次性完成，可以将多个h5文件路径以列表形式赋值给datafile参数
5 lagtime, rmsd = get_lagtime_rmsd(
6     datafile="dspawpy_proj/dspawpy_tests/inputs/2.18/aimd.h5",
7     timestep=None, # 数据文件路径 # 默认从datafile文件中读取时间步长
8 )
```

(续下页)

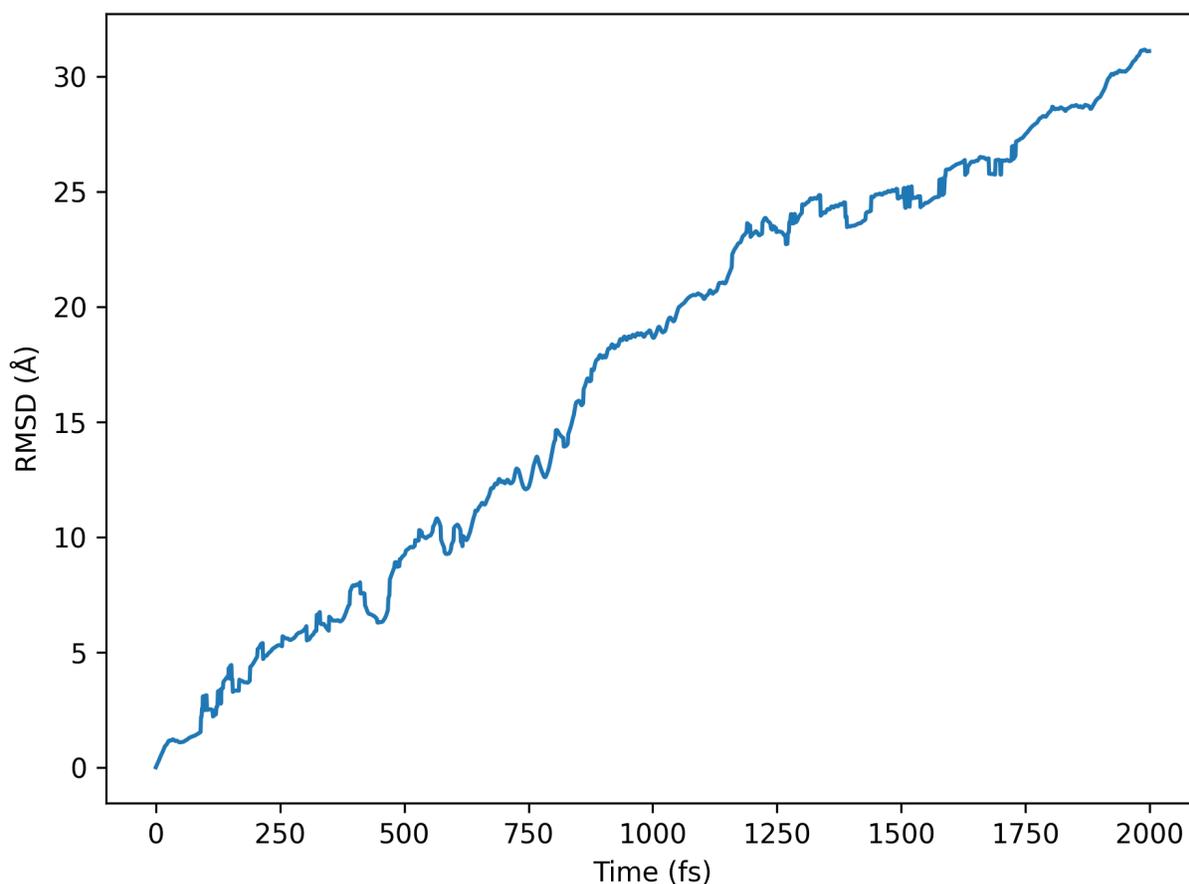
(接上页)

```

9 plot_rmsd(
10     lagtime, # 横坐标
11     rmsd, # 纵坐标
12     xlim=None, # 设置x轴的显示范围
13     ylim=None, # 设置y轴的显示范围
14     figname="dspawpy_proj/dspawpy_tests/outputs/us/10RMSD.png", # 输出的图像文件名
15     show=False, # 是否弹出图像窗口
16     ax=None, # 可指定子图
17 )

```

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')

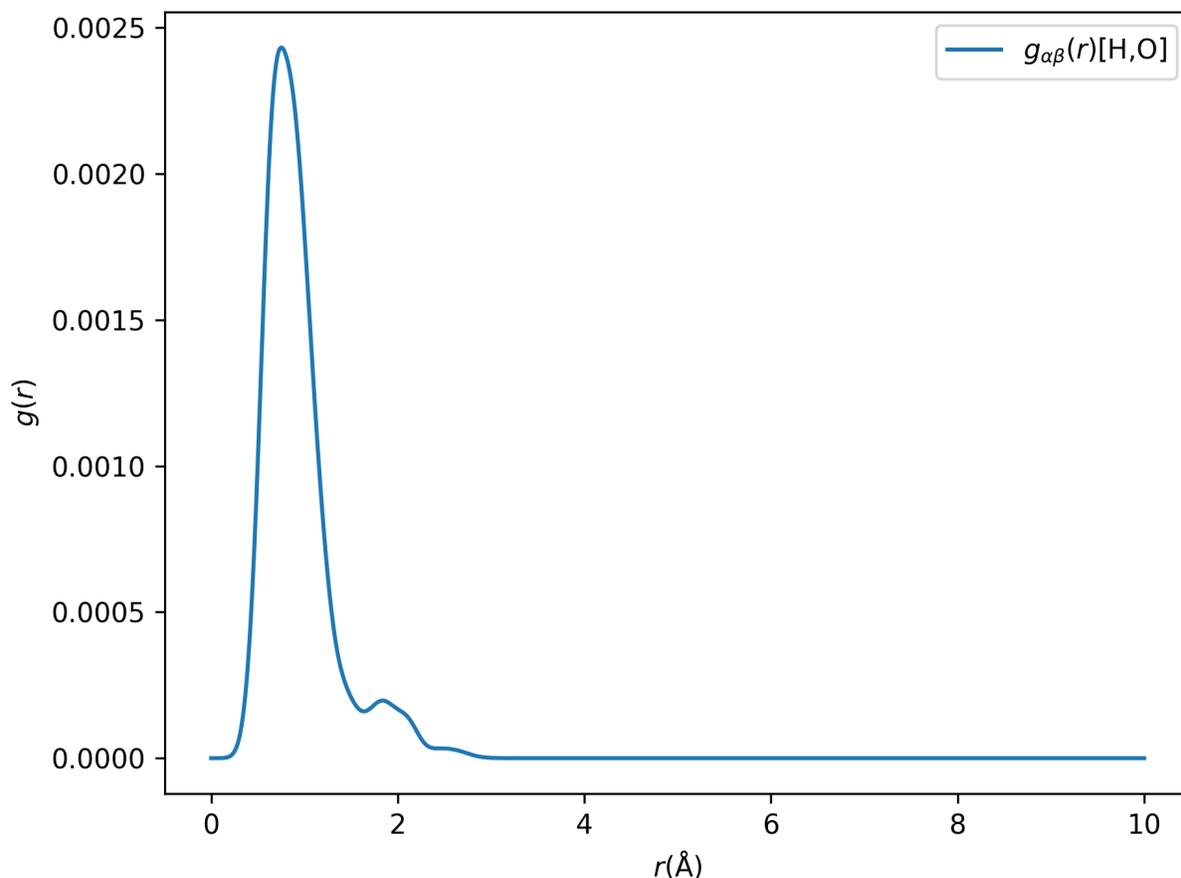
```

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 :

```
1 # coding:utf-8
2 from dspawpy.analysis.aimdtools import get_rs_rdfs, plot_rdf
3
4 # 如果AIMD非一次性完成, 可以将多个h5文件路径以列表形式赋值给datafile参数
5 rs, rdfs = get_rs_rdfs(
6     datafile="dspawpy_proj/dspawpy_tests/inputs/2.18/aimd.h5", # 数据文件路径
7     ele1="H", # 中心元素
8     ele2="O", # 对象元素
9     rmin=0.0, # 最小半径
10    rmax=10.0, # 最大半径
11    ngrid=1000, # 网格数
12    sigma=0.1, # sigma值
13 )
14 plot_rdf(
15     rs, # 横坐标
16     rdfs, # 纵坐标
17     "H", # 中心元素
18     "O", # 对象元素
19     figname="dspawpy_proj/dspawpy_tests/outputs/us/10RDF.png", # 图像保存路径
20     show=False, # 是否弹出图像窗口
21     ax=None, # 可指定子图
22 )
```

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, filename: str = 'aimd.png', flags_str:
                        str = '12345', raw: bool = False)
```

AIMD 任务完成后，绘制关键物理量的收敛过程图

aimd.h5 -> aimd.png

参数

- `datafile` -- h5 文件位置. 例如 'aimd.h5' 或 ['aimd.h5', 'aimd2.h5']
- `show` -- 是否展示交互界面. 默认 False
- `filename` -- 保存的图片路径. 默认 'aimd.h5'
- `flags_str` -- 子图编号. 1. 动能 2. 总能 3. 压力 4. 温度 5. 体积
- `raw` -- 是否输出绘图数据到 csv 文件

返回

图片路径，默认 'aimd.png'

返回类型

figname

示例

```
>>> from dspawpy.plot import plot_aimd
```

读取 aimd.h5 文件内容, 画出动能、总能、温度、体积的收敛过程图, 并保存相应数据到 rawaimd_*.csv 中

```
>>> plot_aimd(datafile='dspawpy_proj/dspawpy_tests/inputs/2.18/aimd.h5', flags_str='1 2 3 4 5', raw=True, show=False, figname="dspawpy_proj/dspawpy_tests/outputs/doctest/aimdconv.png")
>>> plot_aimd(datafile='dspawpy_proj/dspawpy_tests/inputs/2.18/aimd.json', flags_str='1 2 3 4 5', show=False, figname="dspawpy_proj/dspawpy_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)
```

计算不同时间步长下的均方位移

参数

- **datafile** --

- * aimd.h5/aimd.json 文件路径或包含这些文件的文件夹路径 (优先寻找 aimd.h5)
- * 写成列表将依次读取数据并合并到一起
- * 例如 ['aimd1.h5', 'aimd2.h5', '/data/home/my_aimd_task']

- **select** -- 选择原子序号或元素, 原子序号从 0 开始; 默认为'all', 计算所有原子

- **msd_type** -- 计算 MSD 的类型, 可选 xyz,xy,xz,yz,x,y,z, 默认为'xyz', 即计算所有分量

- **timestep** -- 相邻结构的时间间隔, 单位为 fs, 默认 None, 将从 datafile 中读取, 失败则设为 1.0fs; 若不为 None, 则将使用该值计算时间序列

返回

- **lagtime** (*np.ndarray*) -- 时间序列

- **result** (*np.ndarray*) -- 均方位移序列

示例

```
>>> from dspawpy.analysis.aimdtools import get_lagtime_msd
>>> lagtime, msd = get_lagtime_msd(datafile='dspawpy_proj/dspawpy_tests/inputs/2.18/aimd.json', timestep=0.1)
Calculating MSD...
>>> lagtime, msd = get_lagtime_msd(datafile='dspawpy_proj/dspawpy_tests/inputs/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
```

(续下页)

(接上页)

```

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='dspawpy_proj/dspawpy_tests/inputs/2.18/
↳aimd.h5', select='H')
Calculating MSD...
>>> lagtime, msd = get_lagtime_msd(datafile='dspawpy_proj/dspawpy_tests/inputs/2.18/
↳aimd.json', select=[0,1])
Calculating MSD...
>>> lagtime, msd = get_lagtime_msd(datafile='dspawpy_proj/dspawpy_tests/inputs/2.18/
↳aimd.h5', select=['H','O'])
Calculating MSD...
>>> lagtime, msd = get_lagtime_msd(datafile='dspawpy_proj/dspawpy_tests/inputs/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** --
 - * aimd.h5/aimd.json 文件路径或包含这些文件的文件夹路径（优先寻找 aimd.h5）
 - * 写成列表将依次读取数据并合并到一起
 - * 例如 ['aimd1.h5', 'aimd2.h5', '/data/home/my_aimd_task']
- **timestep** -- 相邻结构的时间间隔，单位为 fs，默认 None，将从 datafile 中读取，失败则设为 1.0fs；若不为 None，则将使用该值计算时间序列

返回

- **lagtime** (*numpy.ndarray*) -- 时间序列
- **rmsd** (*numpy.ndarray*) -- 均方根偏差序列

示例

```

>>> from dspawpy.analysis.aimdtools import get_lagtime_rmsd
>>> lagtime, rmsd = get_lagtime_rmsd(datafile='dspawpy_proj/dspawpy_tests/inputs/2.18/
↳aimd.json')
Calculating RMSD...
>>> lagtime, rmsd = get_lagtime_rmsd(datafile='dspawpy_proj/dspawpy_tests/inputs/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,
       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*)

计算 rdf 分布函数

参数

- **datafile** --

- **lagtime** (*np.ndarray*) -- 时间序列
- **result** (*np.ndarray*) -- 均方位移序列
- **xlim** -- x 轴的范围, 默认为 None, 自动设置
- **ylim** -- y 轴的范围, 默认为 None, 自动设置
- **figname** -- 图片名称, 默认为 None, 不保存图片
- **show** -- 是否显示图片, 默认为 True
- **ax** -- 用于将图片绘制到 matplotlib 的子图上
- ****kwargs** (*dict*) -- 其他参数, 如线条宽度、颜色等, 传递给 `plt.plot` 函数

返回类型

MSD 分析后的图片

示例

```
>>> from dspawpy.analysis.aimdtools import get_lagtime_msd, plot_msd
```

指定 h5 文件位置, 用 `get_lagtime_msd` 函数获取数据, `select` 参数选择第 `n` 个原子 (不是元素)

```
>>> lagtime, msd = get_lagtime_msd('dspawpy_proj/dspawpy_tests/inputs/2.18/aimd.h5',
↳select=[0])
Calculating MSD...
```

用获取的数据画图并保存

```
>>> plot_msd(lagtime, msd, xlim=[0,800], ylim=[0,1000], figname='dspawpy_proj/dspawpy_
↳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)
```

AIMD 计算后分析 rdf 并画图

参数

- **rs** (*numpy.ndarray*) -- 径向分布网格点
- **rdfs** (*numpy.ndarray*) -- 径向分布函数
- **ele1** -- 中心元素
- **ele2** -- 相邻元素
- **xlim** -- x 轴范围, 默认为 None, 即自动设置
- **ylim** -- y 轴范围, 默认为 None, 即自动设置
- **figname** -- 图片名称, 默认为 None, 即不保存图片
- **show** -- 是否显示图片, 默认为 True
- **ax** (*matplotlib.axes.Axes*) -- 画图的坐标轴, 默认为 None, 即新建坐标轴
- ****kwargs** (*dict*) -- 其他参数, 如线条宽度、颜色等, 传递给 `plt.plot` 函数

返回类型

rdf 分析后的图片

示例

```
>>> from dspawpy.analysis.aimdtools import get_rs_rdfs, plot_rdf
```

先获取 rs 和 rdfs 数据作为 xy 轴数据

```
>>> rs, rdfs = get_rs_rdfs('dspawpy_proj/dspawpy_tests/inputs/2.18/aimd.h5', 'H', 'O',
↳ rmax=6)
Calculating RDF...
```

将 xy 轴数据传入 plot_rdf 函数绘图

```
>>> plot_rdf(rs, rdfs, 'H','O', xlim=[0, 6], ylim=[0, 0.015], figname='dspawpy_proj/
↳ dspawpy_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**)

AIMD 计算后分析 rmsd 并画图

参数

- **lagtime** -- 时间序列
- **result** -- 均方根偏差序列
- **xlim** -- x 轴范围
- **ylim** -- y 轴范围
- **figname** -- 图片保存路径
- **show** -- 是否显示图片
- **ax** (`matplotlib.axes._subplots.AxesSubplot`) -- 画子图的话, 传入子图对象
- ****kwargs** (`dict`) -- 传入 `plt.plot` 的参数

返回类型

rmsd 分析结构的图片

示例

```
>>> from dspawpy.analysis.aimdtools import get_lagtime_rmsd, plot_rmsd
```

timestep 表示时间步长

```
>>> lagtime, rmsd = get_lagtime_rmsd(datafile='dspawpy_proj/dspawpy_tests/inputs/2.18/
↳ aimd.h5', timestep=0.1)
Calculating RMSD...
>>> lagtime, rmsd = get_lagtime_rmsd(datafile='dspawpy_proj/dspawpy_tests/inputs/2.18/
↳ aimd.json', timestep=0.1)
Calculating RMSD...
```

直接保存为 RMSD.png 图片

```
>>> plot_rmsd(lagtime, rmsd, xlim=[0,200], ylim=[0, 30], figname='dspawpy_proj/dspawpy_
↳ tests/outputs/doctest/RMSD.png', show=False)
==> ...RMSD.png
...
```

- 提取数据自行处理请参考：

```

1 from dspawpy.io.read import get_sinfo
2 from dspawpy.io.structure import read
3
4 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
5
6 # 一次性读取多个aimd.h5文件中的数据并创建pymatgen的Structures列表
7 pymatgen_structures = read(datafile=aimd_h5_files)
8
9 # 或者将数组提取出来
10 for i, df in enumerate(aimd_h5_files): # 依次获取每个aimd.h5文件的数据
11     Nstep, elements, positions, lattices, D_mag_fix = get_sinfo(df)
12     # Nstep 表示总离子步数 (int)
13     # elements 表示元素列表, (Natom x 1)
14     # positions 表示离子坐标, (Nstep x Natom x 3)
15     # lattices 表示晶胞矩阵, (Nstep x 3 x 3)
16     # 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：

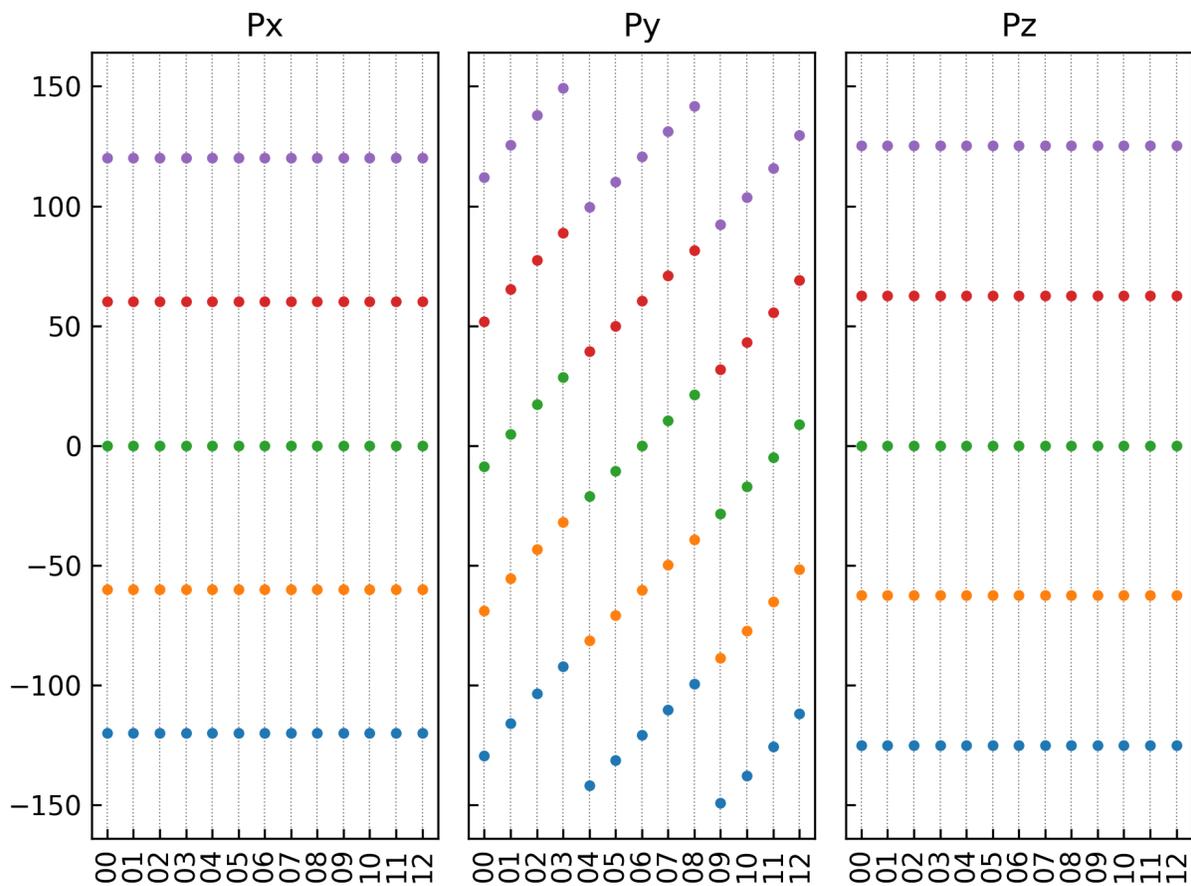
- 参考 11Ferri.py :

```

1 # coding:utf-8
2 from dspawpy.plot import plot_polarization_figure
3
4 plot_polarization_figure(
5     directory="dspawpy_proj/dspawpy_tests/inputs/2.20", # 铁电极化计算路径
6     repetition=2, # 数据点绘图时重复次数
7     figname="dspawpy_proj/dspawpy_tests/outputs/us/11pol.png", # 输出的极化图文件名
8     show=False, # 是否显示极化图
9 ) # --> pol.png

```

执行代码将生成如下组合图：



12 组结构对应极化数值

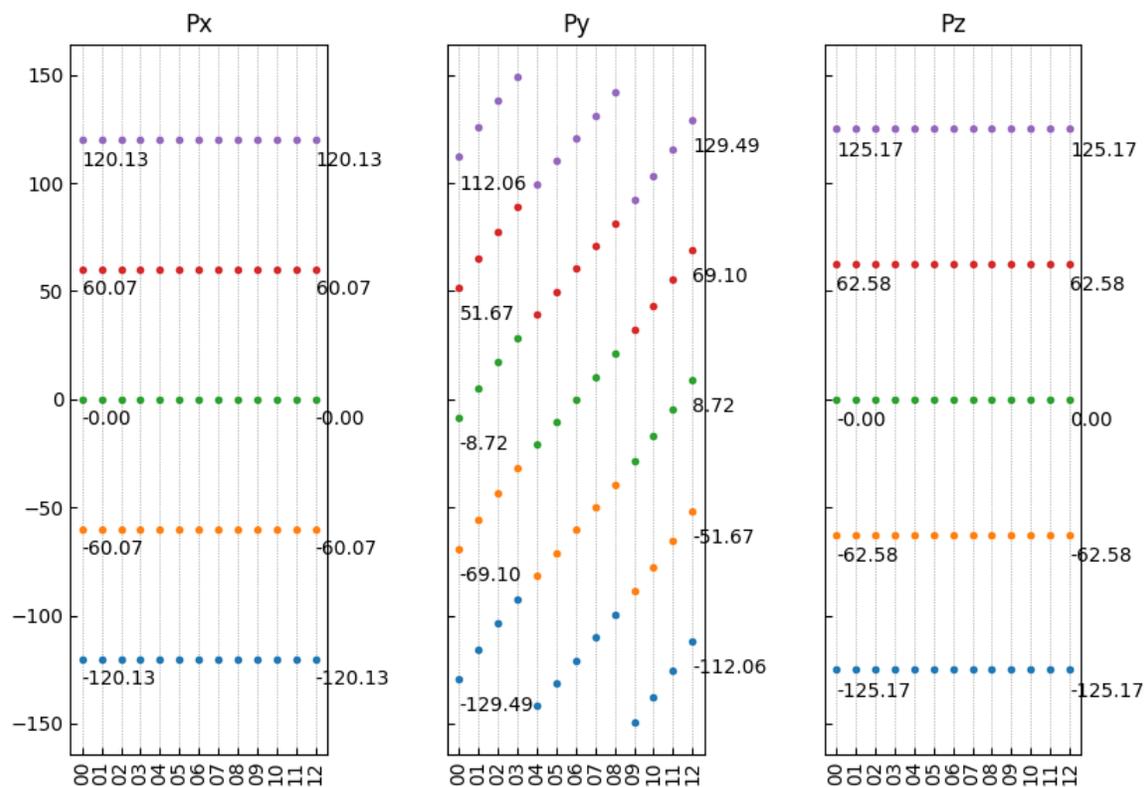
查看首尾构型的铁电极化数值，可以参考如下：

```

1 from dspawpy.plot import plot_polarization_figure
2
3 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）

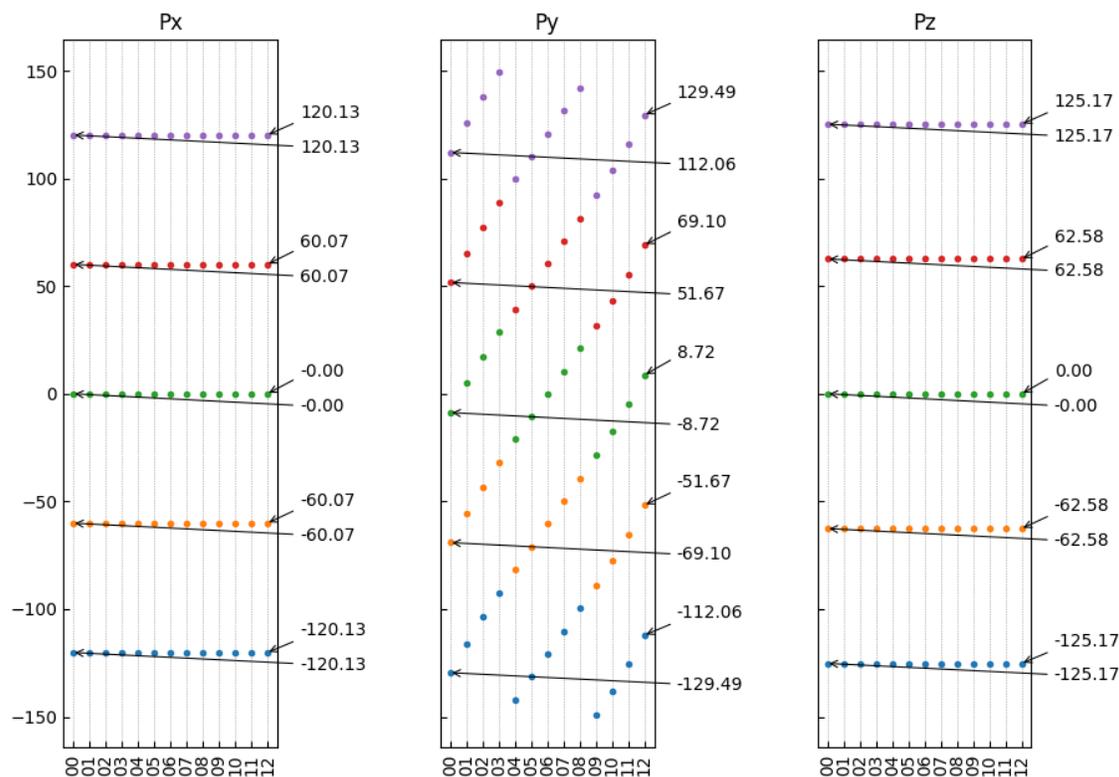
也可以使用第二种批注样式：

```

1 from dspawpy.plot import plot_polarization_figure
2
3 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)

绘制铁电极化结果图

参数

- **directory** -- 铁电极化计算任务主目录
- **repetition** -- 沿上（或下）方向重复绘制的次数, 默认 2
- **annotation** -- 是否显示首尾构型的铁电极化数值, 默认显示
- **show** -- 是否交互显示图片, 默认 True
- **figname** -- 图片保存路径, 默认 'pol.png'
- **raw** -- 是否将原始数据保存到 csv 文件

返回

axes -- 可传递给其他函数进行进一步处理

返回类型

matplotlib.axes._subplots.AxesSubplot

示例

```
>>> from dspawpy.plot import plot_polarization_figure
>>> result = plot_polarization_figure(directory='dspawpy_proj/dspawpy_tests/inputs/2.
↳20', figname='dspawpy_proj/dspawpy_tests/outputs/doctest/pol1.png', show=False,↳
↳annotation=True, annotation_style=1)
>>> result = plot_polarization_figure(directory='dspawpy_proj/dspawpy_tests/inputs/2.
↳20', figname='dspawpy_proj/dspawpy_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 ($6.626 \times 10^{-34} \text{ J} \cdot \text{s}$)， N 是原子数。

- 参考 12getZPE.py :

```
1 # coding:utf-8
2 from dspawpy.io.utils import getZPE
3
4 # 导入频率计算得到的 frequency.txt 文件
5 getZPE (
6     fretxt="dspawpy_proj/dspawpy_tests/inputs/2.13/frequency.txt",
7 )
```

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')
```

从 fretxt 中读取数据，计算 ZPE

将另外保存结果到 ZPE_TS.dat 中

参数

fretxt -- 记录频率信息的文件所在路径, 默认当前路径下的'frequency.txt'

返回

零点能

返回类型

ZPE

示例

```
>>> from dspawpy.io.utils import getZPE
>>> ZPE=getZPE(fretxt='dspawpy_proj/dspawpy_tests/inputs/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:

$$\Delta S_{ads}(0 \rightarrow T, P^0) = S_{vib} = \sum_{i=1}^{3N} \left[\frac{N_A h \nu_i}{T (e^{h\nu_i/k_B T} - 1)} - R \ln \left(1 - e^{-h\nu_i/k_B T} \right) \right]$$

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 ($6.022 \times 10^{23} \text{ mol}^{-1}$), h 是普朗克常数 ($6.626 \times 10^{-34} \text{ J} \cdot \text{s}$), k_B 是玻尔兹曼常数 ($1.38 \times 10^{-23} \text{ J} \cdot \text{K}^{-1}$), R 是理想气体常数 ($8.314 \text{ J} \cdot \text{mol}^{-1} \cdot \text{K}^{-1}$), T 是体系温度, 单位 K 。

- 参考 13getTSads.py :

```
1 # coding:utf-8
2 from dspawpy.io.utils import getTSads
3
4 # 导入频率计算得到的 frequency.txt 文件, 温度可自行修改
5 getTSads(
6     fretxt="dspawpy_proj/dspawpy_tests/inputs/2.13/frequency.txt",
7     T=298.15,
8 )
```

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:

$$\begin{aligned} S(T, P) &= S(T, P^0) - k_B \ln \frac{P}{P^0} \\ &= S_{\text{trans}} + S_{\text{rot}} + S_{\text{elec}} + S_{\text{vib}} - k_B \ln \frac{P}{P^0} \end{aligned}$$

其中:

$$S_{\text{trans}} = k_B \left\{ \ln \left[\left(\frac{2\pi M k_B T}{h^2} \right)^{3/2} \frac{k_B T}{P^\circ} \right] + \frac{5}{2} \right\}$$

$$S_{\text{rot}} = \begin{cases} 0 & , \text{monatomic} \\ k_B \left[\ln \left(\frac{8\pi^2 I k_B T}{\sigma h^2} \right) + 1 \right] & , \text{linear} \\ k_B \left\{ \ln \left[\frac{\sqrt{\pi I_A I_B I_C}}{\sigma} \left(\frac{8\pi^2 k_B T}{h^2} \right)^{3/2} \right] + \frac{3}{2} \right\} & , \text{nonlinear} \end{cases}$$

$$S_{\text{elec}} = k_B \ln[2 \times (\text{total spin}) + 1]$$

$$S_{\text{vib}} = k_B \sum_i^{\text{vib DOF}} \left[\frac{\epsilon_i}{k_B T (e^{\epsilon_i/k_B T} - 1)} - \ln(1 - e^{-\epsilon_i/k_B T}) \right]$$

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:

```

1 # coding:utf-8
2 from dspawpy.io.utils import getTSgas
3
4 # 从计算结果文件 json 或 h5 中读取元素和坐标
5 TSgas = getTSgas(
6     fretxt="dspawpy_proj/dspawpy_tests/inputs/2.13/frequency.txt",
7     datafile="dspawpy_proj/dspawpy_tests/inputs/2.13/frequency.h5",
8     potentialenergy=-0.0,
9     geometry="linear",
10    symmetrynumber=1,
11    spin=1,
12    temperature=298.15,
13    pressure=101325.0,
14 )
15 print("Entropy contribution, T*S (eV): ", TSgas)
16
17 # 如果只有频率 txt 文件, 可通过手动指定元素和坐标完成计算
18 # TSgas = getTSgas(fretxt='dspawpy_proj/dspawpy_tests/inputs/2.13/frequency.txt',
19 ↪ datafile=None, potentialenergy=-0.0, elements=['H', 'H'], geometry='linear',
20 ↪ positions=[[0.0, 0.0, 0.0], [0.0, 0.0, 1.0]], symmetrynumber=1, spin=1, temperature=298.
21 ↪ 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)`

从 fretxt 中读取数据, 计算 ZPE 和 TS

将另外保存结果到 TSads.dat 中

参数

- **fretxt** -- 记录频率信息的文件所在路径, 默认当前路径下的 'frequency.txt'
- **T** -- 温度, 单位 K, 默认 298.15

返回

熵校正

返回类型

TS

示例

```
>>> from dspawpy.io.utils import getTSads
>>> TSads=getTSads(fretxt='dspawpy_proj/dspawpy_tests/inputs/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)
```

理想气体近似下，计算熵的能量贡献

参数

- **fretxt** -- 记录频率信息的文件所在路径, 默认当前路径下的'frequency.txt'
- **datafile** -- 计算结果 json 或 h5 文件或包含它们的文件夹路径, 默认当前路径; 如果设置为 None, 则需要传入 **elements** 和 **positions** 参数
- **potentialenergy** -- 势能, 单位 eV
- **elements** -- 元素列表, 如果
- **geometry** -- 分子几何, monatomic, linear, nonlinear
- **positions** -- 原子坐标, 单位 Angstrom
- **symmetrynumber** -- 对称数
- **spin** -- 自旋数
- **temperature** -- 温度, 单位 K
- **pressure** -- 压强, 单位 Pa

返回

理想气体近似下，计算熵的能量贡献，单位 eV

返回类型

TSgas

示例

```
>>> from dspawpy.io.utils import getTSgas
>>> TSgas=getTSgas(fretxt='dspawpy_proj/dspawpy_tests/inputs/2.13/frequency.txt',
↳datafile='dspawpy_proj/dspawpy_tests/inputs/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 附录

- 快速下载所有脚本，点击 [用户脚本.zip](#)
- [dspawpy 更新日志](#)

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_zhegy_work* 函数发生错误
 - 解决方案: 调整 $cal.methods$

- 错误信息: E1064
 - 错误详情: 在对角化时, 执行 `LAPACKE_zhegv_work` 函数发生错误
 - 解决方案: 调整 `cal.methods`
-

- 错误信息: E1073
 - 错误详情: 并行加速运算发生错误
 - 解决方案: 在提交脚本中关闭 `-pob` 命令重新提交任务
-

- 错误信息: E1115
 - 错误详情: 晶格体积为 0
-

- 错误信息: E1186
 - 错误详情: 在求旋转矩阵的逆时发生错误
 - 解决方案: 关闭对称性 `sys.symmetry = false`
-

- 错误信息: E1187
 - 错误详情: 在求旋转矩阵的逆时发生错误
 - 解决方案: 尝试使用 `cal.ksampling = MP`
-

- 错误信息: E1226
 - 错误详情: 扩胞过程中发生错误
 - 解决方案: 检查并修改结构文件
-

- 错误信息: E1248
 - 错误详情: 在正交化波函数时, `LAPACKE_zpotrf_work` 函数发生错误
 - 解决方案: 将 `sys.symmetry = false` 同时减小 `relax.stepRange`
-

- 错误信息: E1249
 - 错误详情: 在正交化波函数时, `LAPACKE_ztrtri_work` 函数发生错误
-

- 错误信息: E2024/E2025
 - 错误详情: 在求旋转矩阵的逆时发生错误
 - 解决方案: 提高对称性判断的精度, 如设置 `sys.symmetryAccuracy = 1.0e-6`
-

- 错误信息: E3058
- 错误详情: 赝势读取错误
- 解决方案: 目前 *DS-PAW* 提供 72 种元素的赝势, 暂不支持赝势库以外的元素进行计算; 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 Dspawpy 相关问题

1. 来自 numpy 的导包错误

```
from .mtrand import RandomState  
$ ImportError: DLL load failed while importing mtrand
```

请尝试重装 numpy 和 dspawpy

```
$ pip uninstall numpy dspawpy  
$ pip install numpy dspawpy
```

如果还是一样的错误，建议使用 conda 安装 dspawpy

```
$ conda install dspawpy -c conda-forge
```

9.6 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

10.1 2023A 版本更新说明

10.1.1 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
-

10.1.2 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。

10.1.3 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
-

10.1.4 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

10.1.5 2023A 2024/04/03 更新

10.1.5.1 IO 调整

1. optical 计算时补充输出波函数对 k 的导

10.1.5.2 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

10.1.6 2023A 2024/03/15 更新

10.1.6.1 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

10.1.7 2023A 2024/01/12 更新

10.1.7.1 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

10.1.7.2 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

10.1.8 2023A 2023/10/07 更新

10.1.8.1 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

10.1.8.2 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

10.1.9 2023A 2023/6/21 更新

10.1.9.1 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

10.1.9.2 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

10.1.10 2023A 2023/5/9 更新

10.1.10.1 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

10.1.10.2 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

10.2 functions 功能概要



10.3 history 历史版本更新说明

10.3.1 2022A

10.3.1.1 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

10.3.1.2 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

10.3.2 2021B

10.3.2.1 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

10.3.3 2021beta

10.3.3.1 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. 自治场计算
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