VASP calculation practice

Pengfei Suo

November 6, 2021

1 Abstract

VASP[1] code is the most popular ab-initio calculation code. VASP has many pre-processing and post-processing tools, such as Vaspkit[2], Pymatgen, Atomic Simulation environment (ASE). Now we take graphite and graphene examples to show the calculation process of VASP with the help of them. Vaspkit is the most convenient and suggested one.

2 Graphite case

Graphite is a typical layed structure material and can be exfoliated to only one-layer two-dimensional material. The strong covalance interaction dominates the intralayer interaction, while the week vdW interaction dominates the interlayer interaction. Therefore, this natural 3D material shows nearly 2D electronic structure.

2.1 vdW interaction

Generally, DFT calcuation can decribe strong interaction quite well, but fail in weak interaction, such as vdW interaction. To overcome this, many methods have been developed, and they can be classified two types roughly, DFT-D and vdW-DF.

2.2 DFT-D series

DFT-D3BJ or DFT-D4 is suggested

2.2.1 DFT-D

2.2.2 DFT-D2

2.2.3 DFT-D3

2.2.4 DFT-D4

If we want to use DFT-D4 method, DFT-D4 code has to be installed and the VASP code has to be recompiled with DFT-D4. We can get DFTD4 from website https://github.com/dftd4/dftd4. After downloading:

```
mctc-lib mstore multicharge
```

into subprojects directory (decompress and rename them), we can setup a build with:

```
export FC=mpiifort
export CC=mpiicc
export CXX=mpiicpc
meson setup _build
```

To compile and run the projects testsuite use:

```
meson test -C _build --print-errorlogs
```

If the testsuite passes we can install with:

```
meson configure _build --prefix=/opt/dftd4-3.3.0
meson install -C _build
```

The chosen install prefix (/opt/dftd4-3.3.0) requires administrator access.

To include the DFTD4 van-der-Waals functional in VASP, we need to add:

```
CPP_OPTIONS += -DDFTD4
LLIBS      += -L/opt/dftd4-3.3.0/lib64 -ldftd4
INCS       += -I/opt/dftd4-3.3.0/include/dftd4/intel-19.1.2.254
```

in the make.include file, and the compile the VASP code.

2.3 vdW-DF series

2.4 Pure VASP calculation

For convenience, we choose LDA exchange-correlation functional and no vdW correction in this calculation example because of no DFT-D parameters for LDA. (IVDW = 13 for DFTD4). For any VASP calculation, 4 input files are needed, INCAR, POSCAR, POTCAR, KPOINTS.

2.4.1 POSCAR

There are mainly two kinds of methods to get POSCAR file, downloading cif file from website, or building model from software, such as Materials Studio (MS), VESTA, ASE, Pymatgen. Here we download the graphite cif file from Materials Project via a Pymatgen script (`get_graphite_struct.py`):

```
from pymatgen.ext.matproj import MPRester
mpr = MPRester()
struct = mpr.get_structure_by_material_id("mp-48")
struct.to(filename='POSCAR')
```

Using command `python get_graphite_struct.py`, we can get the POSCAR:

```
C4
1.0
1.233862 -2.137112 0.000000
1.233862 2.137112 0.000000
0.000000 0.000000 8.685038
C
4
direct
0.000000 0.000000 0.750000 C
0.000000 0.000000 0.250000 C
0.333333 0.666667 0.750000 C
0.666667 0.333333 0.250000 C
```

2.4.2 INCAR

An INCAR file for relaxation can be:

```
ISTART = 1
LREAL = .FALSE.
PREC = Normal
LWAVE = .FALSE.
LCHARG = .FALSE.
ADDGRID= .TRUE.
NCORE = 4
ENCUT = 520

Electronic Relaxation
ISMEAR = 0
SIGMA = 0.05
NELM = 90
NELMIN = 6
```

```
EDIFF = 1E-06

NSW = 100
IBRION = 2
ISIF = 3
EDIFFG = -1E-02
```

INCAR file for scf calculation can be:

```
ISTART = 1
LREAL = .FALSE.
PREC = Normal
ADDGRID= .TRUE.
NCORE = 4
ENCUT = 520

Electronic Relaxation
ISMEAR = 0
SIGMA = 0.05
NELM = 90
NELMIN = 6
EDIFF = 1E-06
```

2.4.3 KPOINTS

```
Auto
0
Gamma
  15  15   3
0.0  0.0  0.0
```

2.4.4 POTCAR

we choose the POTCAR of `PAW_LDA/C`.

2.4.5 nscf calculation

We can get DOS and band structure via non-self-consistent calculation. In the nscf calculation, we need CHG, CHGCAR and WAVECAR from self-consistent field calculation. For DOS, INCAR file changes to:

```
ISTART = 1
ICHARG = 11
LREAL = .FALSE.
PREC = Normal
LWAVE = .FALSE.
LCHARG = .FALSE.
ADDGRID= .TRUE.
NCORE = 4
ENCUT = 520

Electronic Relaxation
ISMEAR = 0
SIGMA = 0.05
NELM = 90
NELMIN = 6
EDIFF = 1E-06
NEDOS = 3001
LORBIT = 11
```

KPOITNS changes to:

```
K-Spacing Value to Generate K-Mesh: 0.020
0
Gamma
  24  24   6
0.0  0.0  0.0
```

For band structure calculation, INCAR changes to:

```
ISTART = 1
ICHARG = 11
LREAL = .FALSE.
PREC = Normal
LWAVE = .FALSE.
LCHARG = .FALSE.
ADDGRID= .TRUE.
NCORE = 4
ENCUT = 520

Electronic Relaxation
ISMEAR = 0
SIGMA = 0.05
NELM = 90
NELMIN = 6
EDIFF = 1E-06
LORBIT = 11
```

KPOITNS changes to:

```
K-Path Generated by VASPKIT.
   20
Line-Mode
Reciprocal
   0.0000000000   0.0000000000   0.0000000000     GAMMA
   0.5000000000   0.0000000000   0.0000000000     M

   0.5000000000   0.0000000000   0.0000000000     M
   0.3333333333   0.3333333333   0.0000000000     K

   0.3333333333   0.3333333333   0.0000000000     K
   0.0000000000   0.0000000000   0.0000000000     GAMMA

   0.0000000000   0.0000000000   0.0000000000     GAMMA
   0.0000000000   0.0000000000   0.5000000000     A

   0.0000000000   0.0000000000   0.5000000000     A
   0.5000000000   0.0000000000   0.5000000000     L

   0.5000000000   0.0000000000   0.5000000000     L
   0.3333333333   0.3333333333   0.5000000000     H

   0.3333333333   0.3333333333   0.5000000000     H
   0.0000000000   0.0000000000   0.5000000000     A

   0.5000000000   0.0000000000   0.5000000000     L
   0.5000000000   0.0000000000   0.0000000000     M

   0.3333333333   0.3333333333   0.5000000000     H
   0.3333333333   0.3333333333   0.0000000000     K
```

2.5 VASP + Vaspkit

When POSCAR file in the directory, we can use:

```
vaspkit -task 102
```

to get INCAR, KPOINTS, POTCAR simultaneously, and then edit them to run.

2.6 VASP + ASE

In order to run VASP by ASE, we need install ASE via command `pip install ase` at first, and then the `VASP_ASE` setup should be:

```
export ASE_VASP_COMMAND="mpirun -n 24 vasp_std"
export VASP_PP_PATH=/opt/POT
export ASE_VASP_VDW=/opt/POT
```

We can take the relaxation, scf calculation, DOS calculation, band calculation and plot the band structure in an ase script `graphite_ase.py`:

```
from ase.lattice.hexagonal import Graphite
from ase.calculators.vasp import Vasp
from ase.io import read
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
mpl.use('Agg')

def calc_scf():
    calc = Vasp(
            command = 'mpirun -n 24 vasp_std',
            xc = 'LDA',
            setups='recommended',
            kpts=(15,15,3),
            istart=0,
            icharg=2,
            encut=520,
            ncore=4,
            ismear=0,
            sigma=0.1,
            prec='Accurate',
            ediff=1e-6)
    return calc

## DOS calculation
def DOS_calc(atoms):
    calc = calc_scf()
    atoms.calc = calc
    calc.set(
            directory='DOS')
    atoms.get_potential_energy()
    calc.set(kpts=(24,24,6),
            istart=1,
            icharg=11,
            ismear=0,
            sigma=0.1,
            lorbit=10,
            nedos=3001,
            lwave=False,
            lcharg=False)
    atoms.get_potential_energy()
```

```
## bandstructure calculation
def band_calc(atoms):
    calc = calc_scf()
    atoms.calc = calc
    calc.set(
            directory='band')
    atoms.get_potential_energy()
    efermi = float([line for line in open('DOS/DOSCAR') if line.strip()][5].split()[-2])
    calc.set(isym=0,
            kpts={'path':'GKMGAHLA','npoints':200},
            istart=1,
            icharg=11,
            ismear=0,
            sigma=0.1,
            lorbit=10,
            lwave=False,
            lcharg=False)
    atoms.get_potential_energy()
    e_nk = calc.band_structure().energies[0].T - efermi     # get band data with reference to efermi
    path = atoms.cell.bandpath('GKMGAHLA',npoints=200)
    x, X, _ = path.get_linear_kpoint_axis()
    np.savetxt('band/e_nk.dat',e_nk)           # save the band data
    with open('band/kpath.dat','w') as f:
        for k in x:
            print(k,file=f)          # save the kpath axis data
    with open('band/highk.dat','w') as f:
        for k in X:
            print(k,file=f)         # save the high K data

## plot bandstructure
def plot_band(figsize=(6,5)):
    plt.figure(figsize=figsize)
    e_nk = np.loadtxt('band/e_nk.dat')
    x = np.loadtxt('band/kpath.dat')
    X = np.loadtxt('band/highk.dat')
    for e_n in e_nk:
        plt.plot(x, e_n, c='r', lw=2)
    plt.axhline(y=0,c='k',alpha=0.5,lw=1,ls='--')
    for i in X:
        plt.axvline(x=i,c='k',alpha=0.5,lw=1,ls='--')
    plt.axis([x[0],x[-1],-5,5])
    plt.xticks(X,[r'$\Gamma$','K','M',r'$\Gamma$','A','H','L','A'],size=15)
    plt.yticks(size=14)
    plt.ylabel(r'$\varepsilon_n(k) - \varepsilon_{F}$ (eV)', size=20)
    plt.title('band structure of graphite',size=20)
    plt.savefig('band/band_graphite.png',dpi=600)
    plt.close()

if __name__=='__main__':
    atoms = Graphite(symbol='C',latticeconstant={'a':2.46,'c':6.7})
    calc = calc_scf()
    atoms.calc = calc
    calc = calc.set(ediffg=-0.01,isif=3,ibrion=2,nsw=100)    # vc-relax calculator
    atoms.get_potential_energy()
    atoms = read('CONTCAR')
    DOS_calc(atoms)
    band_calc(atoms)
```

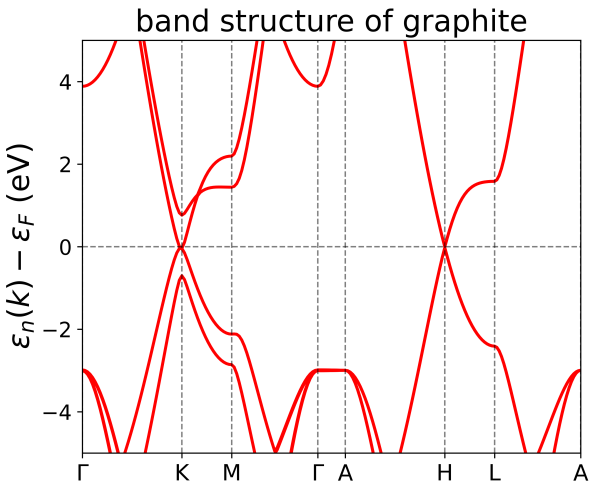

Figure 1: band structure of graphite plotted by the ASE script

```
    plot_band()
```

The band structure is shown in Figure. 1

2.7 VASP + Pymatgen

Pymatgen can also be used as a powerful pre-processing and post-processing tools for VASP. Here we show the band and DOS script `banddos_pymatgen.py`:

```
import matplotlib.pyplot as plt
from pymatgen.io.vasp.outputs import Vasprun
from pymatgen.electronic_structure.plotter import BSDOSPlotter,\
BSPlotter,BSPlotterProjected,DosPlotter

emin = -4
emax = 3
# read vasprun.xml, get band and dos information
dos_vasprun=Vasprun('../DOS/vasprun.xml')
efermi=dos_vasprun.efermi
dos_data = dos_vasprun.complete_dos

bs_vasprun = Vasprun('./vasprun.xml',parse_projected_eigen=True)
bs_data = bs_vasprun.get_band_structure(efermi=efermi,line_mode=True)

# set figure parameters, draw banddos figure
banddos_fig = BSDOSPlotter(bs_projection=None, dos_projection=None, \
    vb_energy_range=-emin, cb_energy_range=emax, fixed_cb_energy=True)
banddos_fig.get_plot(bs=bs_data,dos=dos_data)
plt.savefig('band_fig.png', dpi=600)
plt.close()

# set figure parameters, draw pbandpdos figure
banddos_fig = BSDOSPlotter(bs_projection='elements', \
    vb_energy_range=-emin, cb_energy_range=emax, fixed_cb_energy=True)
banddos_fig.get_plot(bs=bs_data,dos=dos_data)
plt.savefig('pband_fig.png', dpi=600)
plt.close()
```

If we want to use smearing method in the DOS figure, we can change `density[spin]` to `get_smeared_density(0.1)[spin]`

at the 2471 and 2490 rows in file `pymatgen/electronic_structure/plotter.py`, where the 0.1 is the sigma value of Gaussian smearing.

3 Graphene case

Graphene is the first 2D material, only one-layer atoms of graphite. The calculation processes have some difference.

For 2D material, we need to build a slab model, 2D materail + vacuum layer, in order to neglect the effect of image potential in 3D periodic boundary. In the relaxation process, we usually don't need to relax the thickness of vacuum layer. To achieve this, we can change the `constr_cell_relax.F` file in `vasp/src` directory to:

```
!-----------------------------------------------------------------------
!
! At present, VASP does not allow to relax the cellshape selectively
! i.e. for instance only cell relaxation in x direction.
! To be more precisse, this behaviour can not be achived via the INCAR
! or POSCAR file.
! However, it is possible to set selected components of the stress tensor
! to zero.
! The most conveninent position to do this is the routines
! CONSTR_CELL_RELAX  (constraint cell relaxation).
! FCELL contains the forces on the basis vectors.
! These forces are used to modify the basis vectors according
! to the following equations:
!
!      A_OLD(1:3,1:3)=A(1:3,1:3) ! F90 style
!      DO J=1,3
!      DO I=1,3
!      DO K=1,3
!        A(I,J)=A(I,J) + FCELL(I,K)*A_OLD(K,J)*STEP_SIZE
!      ENDDO
!      ENDDO
!      ENDDO
! where A holds the basis vectors (in cartesian coordinates).
!
!-----------------------------------------------------------------------

      SUBROUTINE CONSTR_CELL_RELAX(FCELL)
      USE prec
      REAL(q) FCELL(3,3)

!     just one simple example
!     relaxation in x directions only
!      SAVE=FCELL(1,1)
!      FCELL=0   ! F90 style: set the whole array to zero
!      FCELL(1,1)=SAVE
!     relaxation in z direction only
!      SAVE=FCELL(3,3)
!      FCELL=0   ! F90 style: set the whole array to zero
!      FCELL(3,3)=SAVE

      LOGICAL FILFLG
      INTEGER ICELL(3,3)
      INQUIRE(FILE='OPTCELL',EXIST=FILFLG)
      IF (FILFLG) THEN
         OPEN(67,FILE='OPTCELL',FORM='FORMATTED',STATUS='OLD')
```

```
         DO J=1,3
            READ(67,"(3I1)") (ICELL(I,J),I=1,3)
         ENDDO
         CLOSE(67)
         DO J=1,3
           DO I=1,3
             IF (ICELL(I,J)==0) FCELL(I,J)=0.0
           ENDDO
         ENDDO
      ENDIF

      RETURN
      END SUBROUTINE
```

and recompile the VASP code. Then in the relaxation, put a file named OPTCELL:

```
110
110
000
```

the c-axis will not be changed.

3.1 Pure VASP

3.1.1 Relaxation and scf calculation

The INCAR and POTCAR file can be the same as graphite case, the POSCAR changes to

```
 C
 1.0000000000000000
     2.4600000000000000    0.0000000000000000    0.0000000000000000
    -1.2300000000000000    2.1304224933097191    0.0000000000000000
     0.0000000000000000   -0.0000000000000000   20.0000000000000000
 C
   2
Cartesian
  0.0000000000000000  0.0000000000000000 10.0000000000000000
  1.2300000000000000  0.7101408311032397 10.0000000000000000
```

the KPOINTS changes to

```
Auto
0
Gamma
  15  15   1
0.0  0.0  0.0
```

Combined with OPTCELL, we can run it.

3.1.2 Band structure calculation

the KPOITNS changes to

```
K-Path Generated by VASPKIT.
   20
Line-Mode
Reciprocal
   0.0000000000   0.0000000000   0.0000000000     GAMMA
   0.5000000000   0.0000000000   0.0000000000     M

   0.5000000000   0.0000000000   0.0000000000     M
```

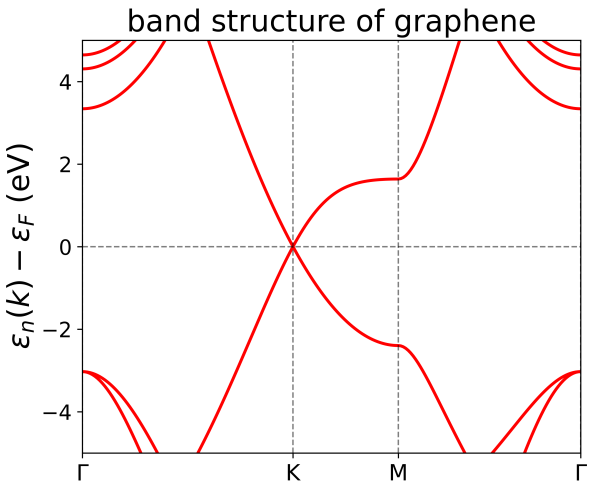

Figure 2: band structure of graphene plotted by the ASE script

```
  0.3333333333   0.3333333333   0.0000000000     K

  0.3333333333   0.3333333333   0.0000000000     K
  0.0000000000   0.0000000000   0.0000000000     GAMMA
```

3.2 VASP + Vaspkit

This is the same as graphite case. When POSCAR file in the directory, we can use:

```
vaspkit -task 102
```

to get INCAR, KPOINTS, POTCAR simultaneously, and then edit them to run.

3.3 VASP + ASE

We can take the relaxation, scf calculation, DOS calculation, band calculation and plot the band structure in an ase script `graphene_ase.py`:

```
from ase.build.surface import graphene
from ase.calculators.vasp import Vasp
from ase.io import read
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
mpl.use('Agg')

def calc_scf():
    calc = Vasp(
            command = 'mpirun -n 24 vasp_std',
            xc = 'LDA',
            setups='recommended',
            kpts=(15,15,1),
            istart=0,
            icharg=2,
            encut=520,
            ncore=4,
            ismear=0,
            sigma=0.1,
```

```
            prec='Accurate',
            ediff=1e-6)
    return calc

## DOS calculation
def DOS_calc(atoms):
    calc = calc_scf()
    atoms.calc = calc
    calc.set(
            directory='DOS')
    atoms.get_potential_energy()
    calc.set(kpts=(24,24,1),
            istart=1,
            icharg=11,
            ismear=0,
            sigma=0.1,
            lorbit=10,
            nedos=3001,
            lwave=False,
            lcharg=False)
    atoms.get_potential_energy()

## bandstructure calculation
def band_calc(atoms):
    calc = calc_scf()
    atoms.calc = calc
    calc.set(
            directory='band')
    atoms.get_potential_energy()
    efermi = float([line for line in open('DOS/DOSCAR') if line.strip()][5].split()[-2])
    calc.set(isym=0,
            kpts={'path':'GKMG','npoints':200},
            istart=1,
            icharg=11,
            ismear=0,
            sigma=0.1,
            lorbit=10,
            lwave=False,
            lcharg=False)
    atoms.get_potential_energy()
    e_nk = calc.band_structure().energies[0].T - efermi     # get band data with reference to efermi
    path = atoms.cell.bandpath('GKMG',npoints=200)
    x, X, _ = path.get_linear_kpoint_axis()
    np.savetxt('band/e_nk.dat',e_nk)           # save the band data
    with open('band/kpath.dat','w') as f:
        for k in x:
            print(k,file=f)          # save the kpath axis data
    with open('band/highk.dat','w') as f:
        for k in X:
            print(k,file=f)         # save the high K data

## plot bandstructure
def plot_band(figsize=(6,5)):
    plt.figure(figsize=figsize)
    e_nk = np.loadtxt('band/e_nk.dat')
    x = np.loadtxt('band/kpath.dat')
    X = np.loadtxt('band/highk.dat')
    for e_n in e_nk:
```

```
        plt.plot(x, e_n, c='r', lw=2)
    plt.axhline(y=0,c='k',alpha=0.5,lw=1,ls='--')
    for i in X:
        plt.axvline(x=i,c='k',alpha=0.5,lw=1,ls='--')
    plt.axis([x[0],x[-1],-5,5])
    plt.xticks(X,[r'$\Gamma$','K','M',r'$\Gamma$'],size=15)
    plt.yticks(size=14)
    plt.ylabel(r'$\varepsilon_n(k) - \varepsilon_{F}$ (eV)', size=20)
    plt.title('band structure of graphene',size=20)
    plt.savefig('band/band_graphene.png',dpi=600)
    plt.close()

if __name__=='__main__':
    atoms = graphene(a=2.46,vacuum=10)
    atoms.pbc = True
    calc = calc_scf()
    atoms.calc = calc
    calc = calc.set(ediffg=-0.01,isif=3,ibrion=2,nsw=100)    # vc-relax calculator
    with open('OPTCELL','w') as f:
        f.write('100\n110\n000')
    atoms.get_potential_energy()
    atoms = read('CONTCAR')
    DOS_calc(atoms)
    band_calc(atoms)
    plot_band()
```

The band structure is shown in Figure. 2

References

[1] Georg Kresse and Jürgen Hafner. Ab initio molecular dynamics for liquid metals. *Physical review B*, 47(1):558, 1993.

[2] Vei Wang, Nan Xu, Jin-Cheng Liu, Gang Tang, and Wen-Tong Geng. Vaspkit: A user-friendly interface facilitating high-throughput computing and analysis using vasp code. *Computer Physics Communications*, 267:108033, 2021.