GPAW calculation practice

Pengfei Suo

November 9, 2021

GPAW[1] is an excellent ab-initio code developed by Technical University of Denmark, using projector augmented-wave (PAW) method with a uniform real-space grid represetation of the electronic wavefunctions. GPAW can be used as a python module, working with ASE.

Here we show a bulk C_6Li example.

1 Relaxation

There are two relaxation methods in this script, vc-relax and equation of state (EOS) fitting. However, two methods give different result. Experimentally, EOS fitting gives more accurate geometry structure, while vc-relax is more efficient.

We use this script `C6Li_bulk.py` to relax the structure:

```
from ase import Atoms
from gpaw import GPAW, PW, FermiDirac, MethfesselPaxton
from ase.eos import EquationOfState as EOS
from ase.build import graphene, add_adsorbate
from ase.build.supercells import make_supercell
from ase.constraints import ExpCellFilter as ECF
from ase.optimize import BFGS
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
mpl.use('Agg')
import gpaw.mpi as mpi

def my_model(a,c):
    C2 = graphene(a=a)
    slab = make_supercell(C2,[[2,1,0],[-1,1,0],[0,0,1]])
    x, y = np.dot([1/3, 1/3],slab.cell[:2,:2])
    add_adsorbate(slab,'Li',position=(x,y),height=c/2)
    slab.cell[2] = [0, 0, c]
    slab.pbc = True
    return slab

def calc_scf(encut,k,width):
    calc = GPAW(
              mode=PW(encut),
              xc='LDA',
              kpts={'density':k,'gamma':True},
              random=True,
              occupations=FermiDirac(width),
              txt='gs.out'
                )
    return calc

def calc_gpaw(encut,width):
    calc = GPAW(
```

```
            mode=PW(encut),
            xc='LDA',
            kpts=(11,11,12),
            occupations=MethfesselPaxton(width),
            convergence={'energy':1e-6},
            txt='gs.out'
              )
    return calc

def encut_conv():
    atoms = my_model(2.47,3.6)
    natom = atoms.get_global_number_of_atoms()
    encuts = np.arange(300, 660, 50)
    energies = []
    for encut in encuts:
        atoms.calc = calc_scf(encut,10,0.01)
        energies.append(atoms.get_potential_energy()/natom)
    plt.plot(encuts, energies,'o-', lw=2)
    plt.xlabel('wavefunction cutoff (eV)', size = 15)
    plt.ylabel('total energy per atom (eV)', size = 15)
    plt.tight_layout()
    plt.savefig('Encut.png')
    plt.close()

def k_conv():
    atoms = my_model(2.47,3.6)
    natom = atoms.get_global_number_of_atoms()
    ks = np.arange(5.5, 12.1, 0.5)
    energies = []
    for k in ks:
        atoms.calc = calc_scf(520,k,0.01)
        energies.append(atoms.get_potential_energy()/natom)
    plt.plot(ks, energies,'o-', lw=2)
    plt.xlabel(r'Kmesh density (1/$\mathrm{\AA}$)', size = 15)
    plt.ylabel('total energy per atom (eV)', size = 15)
    plt.tight_layout()
    plt.savefig('k.png')
    plt.close()

def width_conv():
    atoms = my_model(2.47,3.6)
    natom = atoms.get_global_number_of_atoms()
    widths = np.arange(0.001, 0.052, 0.005)
    energies = []
    for width in widths:
        atoms.calc = calc_scf(520,10,width)
        atoms.get_potential_energy()
        gs_contents = [line for line in open('gs.out') if line.strip()]
        for line in gs_contents:
            if 'Entropy ' in line:
                energy = float(line.split()[-1])/natom
        energies.append(energy)
    plt.plot(widths, energies,'o-', lw=2)
    plt.xlabel('MP smearing width (eV)', size = 15)
    plt.ylabel('Entropy (-ST) per atom (eV)', size = 15)
    plt.tight_layout()
    plt.savefig('width.png')
    plt.close()
```

```
def vcrelax(atoms):
    atoms.calc = calc_gpaw(520,0.1)
    ecf = ECF(atoms)
    relax = BFGS(ecf,logfile='vcrelax.log')
    relax.run(fmax=0.01)
    return atoms

def c_fit(a):
    c_s = np.arange(3.4, 3.71, 0.02)
    Es = []
    Vs = []
    for c in c_s:
        atoms = my_model(a,c)
        atoms.calc = calc_gpaw(520,0.1)
        relax = BFGS(atoms,logfile='c_relax.log')
        relax.run(fmax=0.01)
        Vs.append(atoms.get_volume())
        Es.append(atoms.get_potential_energy())
    eos = EOS(Vs, Es, eos='birchmurnaghan')
    v0, e0, B = eos.fit()
    eos.plot(filename='c_fit.png')
    plt.close()
    return v0/3*2/np.sqrt(3)/a**2

def a_fit(c):
    a_s = np.arange(2.40,2.51,0.01)
    Es = []
    Vs = []
    for a in a_s:
        atoms = my_model(a,c)
        atoms.calc = calc_gpaw(520,0.1)
        relax = BFGS(atoms,logfile='a_relax.log')
        relax.run(fmax=0.01)
        Vs.append(atoms.get_volume())
        Es.append(atoms.get_potential_energy())
    eos = EOS(Vs, Es, eos='birchmurnaghan')
    v0, e0, B = eos.fit()
    eos.plot(filename='a_fit.png')
    plt.close()
    return np.sqrt(v0/3*2/np.sqrt(3)/c)

def relax_fitting():
    c_old = 3.55
    a_old = a_fit(c_old)
    c_new = c_fit(a_old)
    a_new = a_fit(c_new)
    if mpi.world.rank == 0:
        print('a_old = %10.6f, a_new = %10.6f' % (a_old, a_new))
        print('c_old = %10.6f, c_new = %10.6f' % (c_old, c_new))
    count = 0
    while abs(1-c_new/c_old)>1e-4 or abs(1-a_new/a_old)>1e-4:
        c_old = c_new
        a_old = a_new
        c_new = c_fit(a_old)
        a_new = a_fit(c_new)
        count += 1
        if mpi.world.rank == 0:
```

```
            print(count)
            print('a_old = %10.6f, a_new = %10.6f' % (a_old, a_new))
            print('c_old = %10.6f, c_new = %10.6f' % (c_old, c_new))
    atoms = my_model(a_new,c_new)
    atoms.calc = calc_gpaw(520,0.1)
    relax = BFGS(atoms,logfile='relax.log')
    relax.run(fmax=0.01)
    return atoms

if __name__=="__main__":
    atoms = relax_fitting()
    atoms.write('POSCAR1')
    atoms = vcrelax(atoms)
    atoms.write('POSCAR2')
```

the structure from EOS fitting is

```
 C Li
 1.0000000000000000
     3.7125195983168071    2.1434241894599695    0.0000000000000000
    -3.7125195983168071    2.1434241894599695    0.0000000000000000
     0.0000000000000000    0.0000000000000000    3.5960392228939706
 C   Li
   6   1
Cartesian
  0.0000000000000000 -0.0012780827711535 -0.0000000000000000
  1.2386133832942898  0.7138356942937556 -0.0000000000000000
 -1.2386133832941797  2.1440632249858926 -0.0000000000000000
 -0.0000000000000000  2.8591770020502802  0.0000000000000000
  1.2386133832941799  2.1440632249858931 -0.0000000000000000
 -1.2386133832942892  0.7138356942937564  0.0000000000000000
  0.0000000000000000  1.4289494596401655  1.7980196114469853
```

while the structure from vc-relaxation is

```
 C Li
 1.0000000000000000
     3.7104477390308652    2.1422503665714689    0.0000000000000000
    -3.7104477390308679    2.1422503665714689    0.0000000000000000
     0.0000000000000000   -0.0000000000000000    3.5697976562653215
 C   Li
   6   1
Cartesian
  0.0000000000000000 -0.0013289261674183 -0.0000000000000000
  1.2379670935020670  0.7134182435321790 -0.0000000000000000
 -1.2379670935020923  2.1429155785626057  0.0000000000000000
 -0.0000000000000000  2.8576627482622619  0.0000000000000000
  1.2379670935020928  2.1429155785626062  0.0000000000000000
 -1.2379670935020661  0.7134182435321791  0.0000000000000000
  0.0000000000000000  1.4281669110477948  1.7848988281326608
```

we can see that vc-relaxation gets shrinking structure. In addition, VASP calculation also shows that vc-relaxation gets shrinking structure, but not as much as the GPAW. The structure from EOS fitting in VASP is:

```
 C Li
   1.00000000000000
     3.7141432007030621    2.1443615767347310    0.0000000000000000
    -3.7141432007030621    2.1443615767347310    0.0000000000000000
     0.0000000000000000    0.0000000000000000    3.5490371122003963
   C    Li
```

```
     6     1
Direct
 -0.0002566231571666 -0.0002566231571666 -0.0000000000000000
  0.3333333333333357 -0.0002566231571666 -0.0000000000000000
  0.3333333333333357  0.6669232898238310  0.0000000000000000
  0.6669232898238310  0.6669232898238310  0.0000000000000000
  0.6669232898238310  0.3333333333333357  0.0000000000000000
 -0.0002566231571666  0.3333333333333357  0.0000000000000000
  0.3333333333333357  0.3333333333333357  0.5000000000000000

  0.00000000E+00  0.00000000E+00  0.00000000E+00
  0.00000000E+00  0.00000000E+00  0.00000000E+00
  0.00000000E+00  0.00000000E+00  0.00000000E+00
  0.00000000E+00  0.00000000E+00  0.00000000E+00
  0.00000000E+00  0.00000000E+00  0.00000000E+00
  0.00000000E+00  0.00000000E+00  0.00000000E+00
  0.00000000E+00  0.00000000E+00  0.00000000E+00
```

while from vc-relaxation in VASP is

```
 C Li
   1.00000000000000
     3.7133313616753387    2.1438928612535362    0.0000000000000000
    -3.7133313616753387    2.1438928612535362    0.0000000000000000
    -0.0000000000000000   -0.0000000000000000    3.5421417909327939
   C    Li
     6     1
Direct
 -0.0002695086052436 -0.0002695086052436 -0.0000000000000000
  0.3333333333333357 -0.0002695086052436 -0.0000000000000000
  0.3333333333333357  0.6669361752719080  0.0000000000000000
  0.6669361752719080  0.6669361752719080  0.0000000000000000
  0.6669361752719080  0.3333333333333357 -0.0000000000000000
 -0.0002695086052436  0.3333333333333357  0.0000000000000000
  0.3333333333333357  0.3333333333333357  0.5000000000000000

  0.00000000E+00  0.00000000E+00  0.00000000E+00
  0.00000000E+00  0.00000000E+00  0.00000000E+00
  0.00000000E+00  0.00000000E+00  0.00000000E+00
  0.00000000E+00  0.00000000E+00  0.00000000E+00
  0.00000000E+00  0.00000000E+00  0.00000000E+00
  0.00000000E+00  0.00000000E+00  0.00000000E+00
  0.00000000E+00  0.00000000E+00  0.00000000E+00
```

2 Scf and bandstructure calculation

we can take scf calculation and band structrue calculation in this script `band.py`:

```
from gpaw import GPAW, PW, FermiDirac
from ase.io import read
from ase.parallel import paropen

## groud state calculation
atoms = read('../scf/POSCAR')
calc = GPAW(mode=PW(500),
            xc='LDA',
            parallel={'domain':1},
            kpts={'density':11,'gamma':True},
            occupations=FermiDirac(0.01),
```

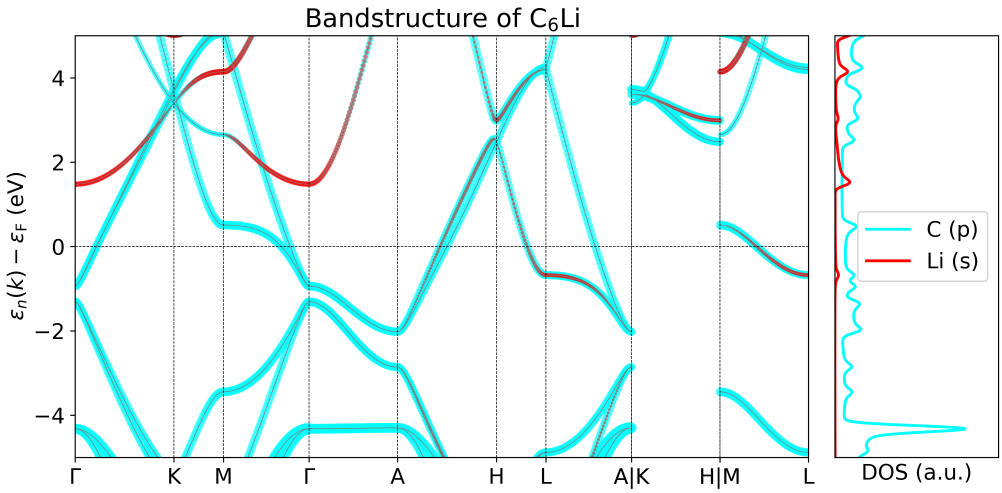

Figure 1: pbanddos of C_6Li plotted via the python script `plot_band.py`

```
            txt='gs.out')

atoms.calc = calc
atoms.get_potential_energy()
calc.write('gs.gpw','all')

## band structure calculation
path = atoms.cell.bandpath('GKMGAHLA,KH,ML',npoints=600)
calc = calc.fixed_density(symmetry='off',
                          kpts=path.kpts,
                          txt='band.out')
calc.write('band.gpw','all')

x, X, _ = path.get_linear_kpoint_axis()
with paropen('kpath.dat','w') as f:
    for k in x:
        print(k,file=f)

with paropen('highk.dat','w') as f:
    for k in X:
        print(k,file=f)
```

and plot the band structure via this script `plot_band.py`:

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
mpl.use('Agg')
from gpaw import GPAW

title = 'Bandstructure of C$_6$Li'
calc = GPAW('band.gpw')
ef = calc.get_fermi_level()
nbnd = calc.get_number_of_bands()
natom = calc.atoms.get_global_number_of_atoms()
NEDOS = 3001
kx = np.loadtxt('kpath.dat')
X = np.loadtxt('highk.dat')
klabels = [r'$\Gamma$','K','M',r'$\Gamma$','A','H','L','','A|K','','H|M','L']
e_kn = np.array([calc.get_eigenvalues(k) for k in range(len(kx))]) - ef
```

```
w_kni = abs(calc.get_projections(locfun='projectors'))
energy, dos = calc.get_dos(npts=NEDOS,width=0.1)
e_nk = e_kn.T

def plotband(figsize=(5,5)):
    plt.figure(figsize=figsize)
    for e_k in e_nk:
        plt.plot(kx,e_k,c='r',lw=2)
    for k in X[1:-1]:
        plt.axvline(x=k,c='k',lw=0.5,ls='--')
    plt.axhline(y=0,c='k',lw=0.5,ls='--')
    plt.xticks(X,klabels,size=15)
    plt.yticks(size=15)
    plt.ylabel(r'$\varepsilon_n(k) - \varepsilon_{\mathrm{F}}$ (eV)',size=15)
    plt.title(title, size=18)
    plt.axis([0,kx[-1],-5,5])
    plt.tight_layout()
    plt.savefig('band.png',dpi=600 )
    plt.close()

def plotdos(figsize=(5,5)):
    plt.figure(figsize=figsize)
    plt.plot(energy - ef, dos, c = 'r', lw=2)
    plt.xlabel(r'$\varepsilon_n(k) - \varepsilon_{\mathrm{F}}$ (eV)',size=15)
    plt.ylabel('Density of States (1/eV)',size=15)
    plt.xlim([-8,8])
    plt.ylim(ymin=0)
    plt.tight_layout()
    plt.savefig('DOS.png',dpi=600 )
    plt.close()

def plotbanddos(figsize=(9,5)):
    plt.figure(figsize=figsize)
    grid = plt.GridSpec(1,5)
    p1 = plt.subplot(grid[0,0:4])
    for e_k in e_nk:
        plt.plot(kx,e_k,c='r',lw=2)
    for k in X[1:-1]:
        plt.axvline(x=k,c='k',lw=0.5,ls='--')
    plt.axhline(y=0,c='k',lw=0.5,ls='--')
    plt.xticks(X,klabels,size=15)
    plt.yticks(size=15)
    plt.ylabel(r'$\varepsilon_n(k) - \varepsilon_{\mathrm{F}}$ (eV)',size=15)
    plt.title(title, size=18)
    plt.axis([0,kx[-1],-5,5])
    p2 = plt.subplot(grid[0,4])
    plt.plot(dos, energy - ef, c='r',lw=2)
    plt.axhline(y=0, lw=0.5, c='k',ls='--')
    plt.axvline(x=0, lw=0.5, c='k',ls='--')
    plt.fill_between(dos,energy-ef,0,where=dos>=0,facecolor='silver',interpolate=True)
    plt.xlim(xmin=0)
    plt.ylim([-5,5])
    plt.xlabel('DOS (a.u.)',size=15)
    plt.xticks([])
    plt.ylabel('')
    plt.yticks([])
    plt.tight_layout()
    plt.savefig('banddos.png',dpi=600 )
```

```
    plt.close()

def plotpband(figsize=(8,5)):
    plt.figure(figsize=figsize)
    for e_k in e_nk:
        plt.plot(kx,e_k,c='0.5',lw=0.5)
    for k in X[1:-1]:
        plt.axvline(x=k,c='k',lw=0.5,ls='--')
    w = np.zeros([len(kx),nbnd,2])
    for i in range(2,23,4):
        w[:,:,0] += w_kni[:,:,i]
    w[:,:,1] = w_kni[:,:,24]
    scale = 60.0
    colors = ['cyan','r']
    labels = ['C (pz)', 'Li (s)']
    st = []
    for i in range(len(colors)):
        st.append(plt.scatter(-1,-1,20,c=colors[i],alpha=0.5,\
        label=labels[i],marker='.',edgecolor='none'))
        for n in range(nbnd):
            st.append(plt.scatter(kx,e_nk[n,],w[:,n,i].T*scale,\
            c=colors[i],alpha=0.5,marker='.',edgecolor='none'))
    plt.axhline(y=0,c='k',lw=0.5,ls='--')
    plt.xticks(X,klabels,size=15)
    plt.yticks(size=15)
    plt.ylabel(r'$\varepsilon_n(k) - \varepsilon_{\mathrm{F}}$ (eV)',size=15)
    plt.title(title, size=18)
    plt.legend(scatterpoints =1, numpoints=1,markerscale=2.0,fontsize=15)
    plt.xlim([0,kx[-1]])
    plt.ylim([-10,10])
    plt.tight_layout()
    plt.savefig('pband_whole.png',dpi=600 )
    plt.ylim([-5,5])
    plt.tight_layout()
    plt.savefig('pband.png',dpi=600 )
    plt.close()

def plotpbanddos(figsize=(7,5)):
    plt.figure(figsize=figsize)
    w = np.zeros([len(kx),nbnd,4])
    for i in range(0,21,4):
        w[:,:,0] += w_kni[:,:,i]
        w[:,:,1] += w_kni[:,:,i+1:i+4].sum(axis=2)
    w[:,:,2] = w_kni[:,:,24]
    w[:,:,3] = w_kni[:,:,25:28].sum(axis=2)
    dos_weight_ia = np.zeros([NEDOS,2*natom])
    count = 0
    for orbital in 'sp':
        for a in range(natom):
            _,dos_weight_ia[:,count]=calc.get_orbital_ldos(a=a,angular=orbital,npts=NEDOS)
            count += 1
    pdos = np.zeros([NEDOS,4])
    pdos[:,0] = dos_weight_ia[:,0:6].sum(axis=1)  # C(s)
    pdos[:,1] = dos_weight_ia[:,7:13].sum(axis=1) # C(p)
    pdos[:,2] = dos_weight_ia[:,6]                # Li(s)
    pdos[:,3] = dos_weight_ia[:,13]               # Li(p)
    scale = 60.0
    colors = ['cyan','r']
```

```
    labels = ['C (p)', 'Li (s)']
    st = []
    grid = plt.GridSpec(1,5)
    p1 = plt.subplot(grid[0,0:4])
    for e_k in e_nk:
        plt.plot(kx,e_k,c='0.5',lw=0.5)
    for k in X[1:-1]:
        plt.axvline(x=k,c='k',lw=0.5,ls='--')
    for i in range(len(colors)):
        st.append(plt.scatter(-1,-1,20,c=colors[i],alpha=0.5,\
        label=labels[i],marker='.',edgecolor='none'))
        for n in range(nbnd):
            st.append(plt.scatter(kx,e_nk[n,],w[:,n,i+1].T*scale,\
            c=colors[i],alpha=0.5,marker='.',edgecolor='none'))
    plt.axhline(y=0,c='k',lw=0.5,ls='--')
    plt.xticks(X,klabels,size=15)
    plt.yticks(size=15)
    plt.ylabel(r'$\varepsilon_n(k) - \varepsilon_{\mathrm{F}}$ (eV)',size=15)
    plt.title(title, size=18)
#    plt.legend(scatterpoints =1, numpoints=1,markerscale=2.0,fontsize=15)
    plt.axis([0,kx[-1],-5,5])

    p2 = plt.subplot(grid[0,4])
    for i in range(len(colors)):
        plt.plot(pdos[:,i+1],energy - ef, c=colors[i],lw=2, label=labels[i])
    plt.xlim(xmin=0)
    plt.ylim([-5,5])
    plt.xlabel('DOS (a.u.)',size = 15)
    plt.xticks([])
    plt.ylabel('')
    plt.yticks([])
    plt.legend(fontsize=15)
    plt.tight_layout()
    plt.savefig('pbanddos.png',dpi=600 )
    plt.close()

if __name__=='__main__':
    plotband((8,5))
    plotdos()
    plotbanddos((10,5))
    plotpband()
    plotpbanddos((10,5))
```

the band structure has been shown in Figure. 1.

References

[1] J Enkovaara, C Rostgaard, J J Mortensen, J Chen, M Dułak, L Ferrighi, J Gavnholt, C Glinsvad, V Haikola, H A Hansen, H H Kristoffersen, M Kuisma, A H Larsen, L Lehtovaara, M Ljungberg, O Lopez-Acevedo, P G Moses, J Ojanen, T Olsen, V Petzold, N A Romero, J Stausholm-Møller, M Strange, G A Tritsaris, M Vanin, M Walter, B Hammer, H Häkkinen, G K H Madsen, R M Nieminen, J K Nørskov, M Puska, T T Rantala, J Schiøtz, K S Thygesen, and K W Jacobsen. Electronic structure calculations with GPAW: a real-space implementation of the projector augmented-wave method. 22(25):253202, jun 2010.