Interfacing abinit with external libraries or packages: electronic structure analysis and molecular dynamics

Prof. Aldo H Romero
West Virginia University

Pedram Tavazohi, Uthpala Herath, Matthieu Verstraete, Eric Bousquet and Hu Xe

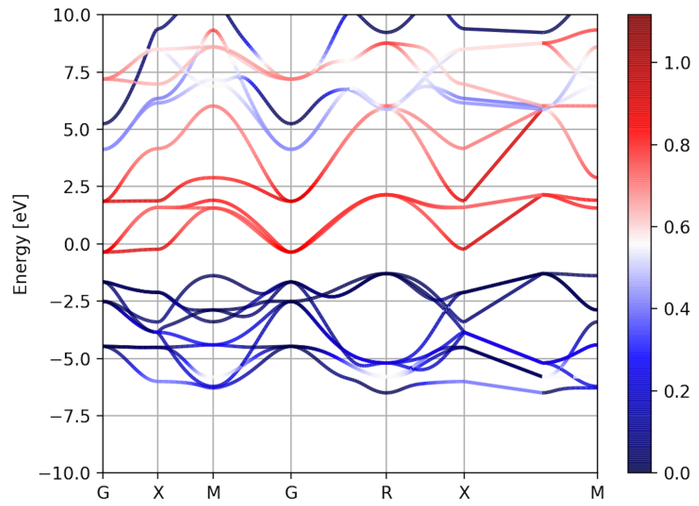
PyProcar: A Python library for DFT pre and post-processing

Available @: github.com/romerogroup/pyprocar

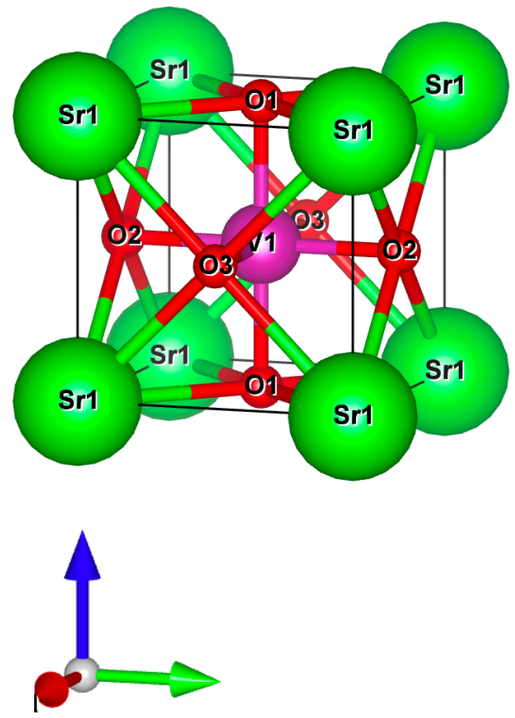
Forum: groups.google.com/d/forum/pyprocar

DFT output:
Numbers

Graphical
representation




```
 k-point     1 :    0.00000000 0.00000000 0.00000000     weight = 0.00625000

band     1 # energy  -29.06876982 # occ.  2.00000000

ion       s      py      pz      px      dxy     dyz     dz2     dxz   x2-y2     tot
    1   0.972   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.972
    2   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000
    3   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000
    4   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000
    5   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000
tot     0.974   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.974

band     2 # energy  -14.13374261 # occ.  2.00000000

ion       s      py      pz      px      dxy     dyz     dz2     dxz   x2-y2     tot
    1   0.011   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.011
    2   0.139   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.139
    3   0.233   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.233
    4   0.233   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.233
    5   0.233   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.233
tot     0.850   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.850
```



Question: Can we use the PROCAR format from other codes?

PyProcar

pre-processing

pyprocar.kpath()

pyprocar.generate2Dkmesh()

PyProcar core classes

- Procarparser()
- Utilsprocar()
- Procarselect()

pyprocar.repair()

post-processing

pyprocar.bandsplot()

pyprocar.cat()

pyprocar.fermi2D()

pyprocar.bandscompare()

pyprocar.fermi3D()

pyprocar. filter()

pyprocar. unfold()

DFT

PROCAR

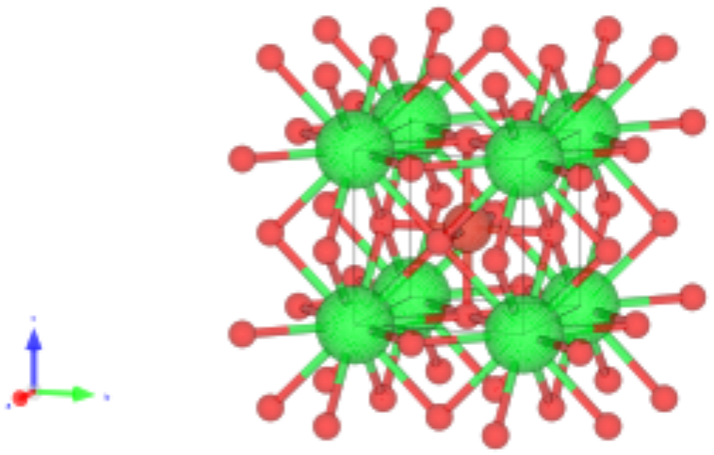
OUTCAR

K path generator

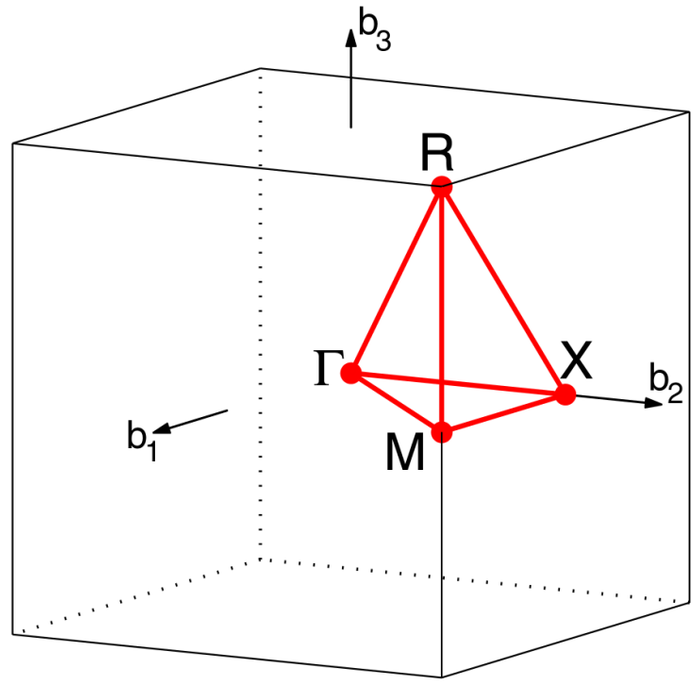
pyprocar.kpath()

- Used to define a path in the BZ for band structure calculations

E.g. : $SrVO_3$



Fourier Transform



- Consider only points and line segments with high crystallographic symmetries are generally more important than those with low symmetries

FORMAT: `pyprocar.kpath(infile,grid_size,with_time_reversal,recipe,threshhold,symprec,angle_tolerence)`

Eg. `pyprocar.kpath('POSCAR',40,True,'hpkot',1e-07,1e-05,-1.0)`

- Given a crystal structure, PyProcar can do this automatically.

prtprocar 1

or

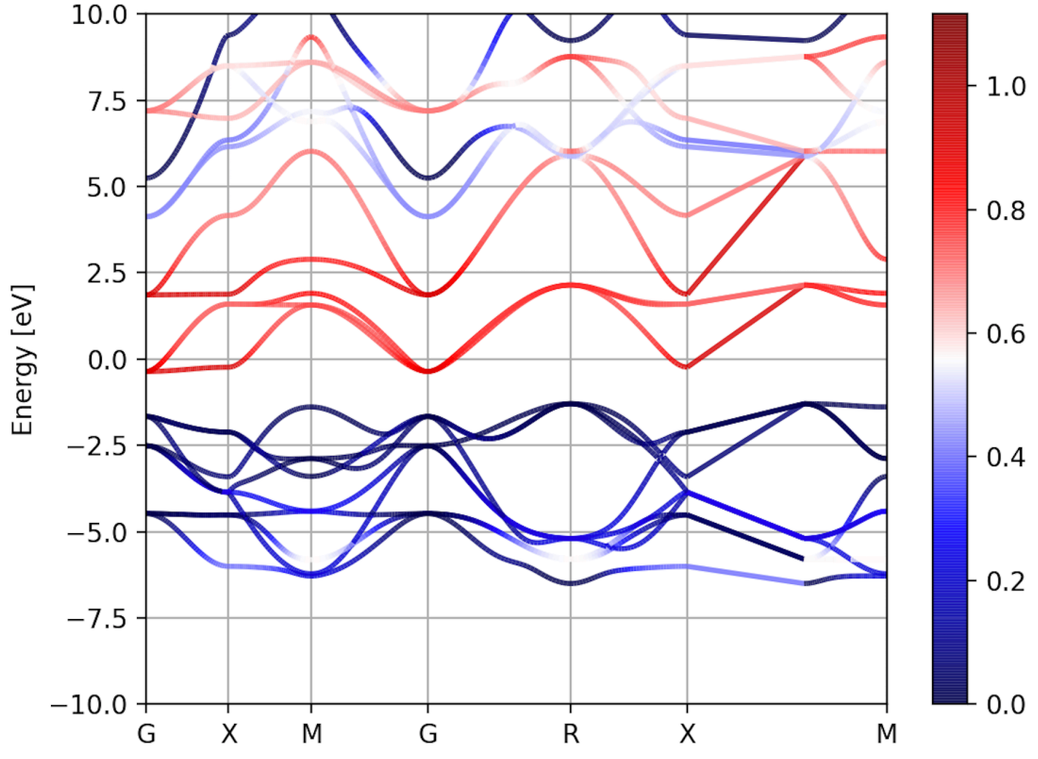
prtprocar 2

+

A normal band structure calculation in Abinit

DFT

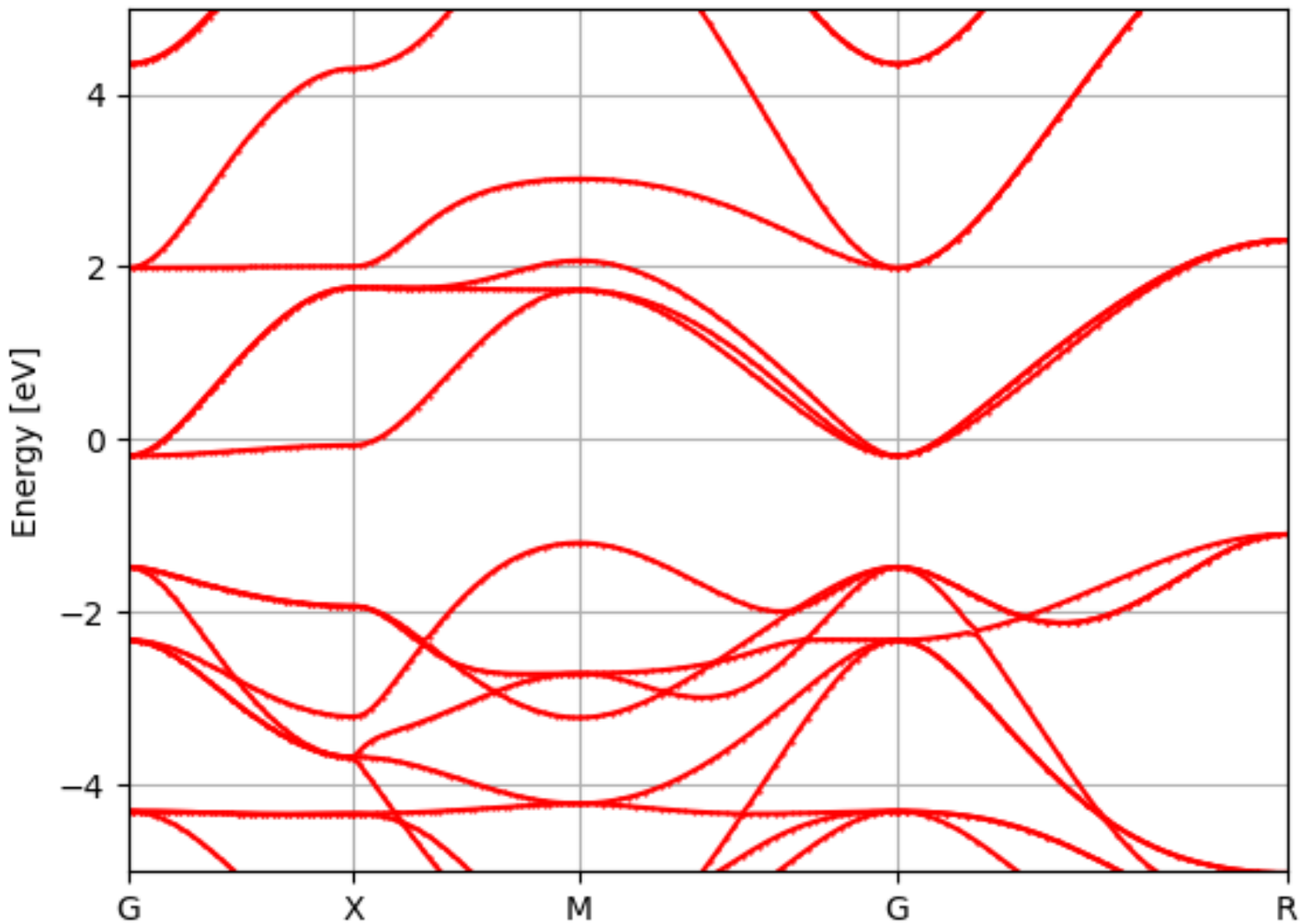
pyprocar.bandsplot()



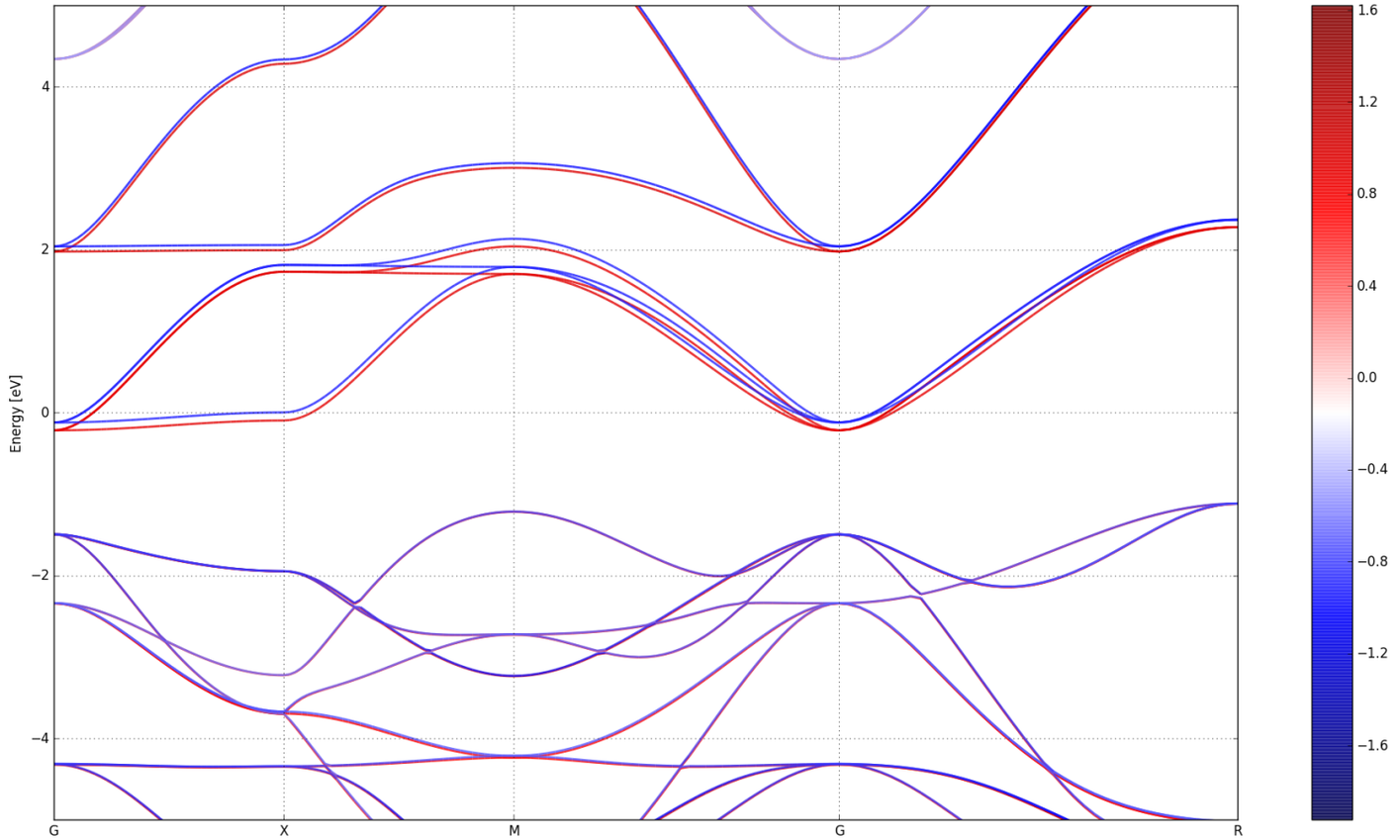
Plotting bands

pyprocar.bandsplot()

- Spin projected bands
  - No spin
  - colinear spin
  - non-colinear spin
- Atom projected bands
- Orbital projected bands
- Hybrids



Colinear spin projected bands

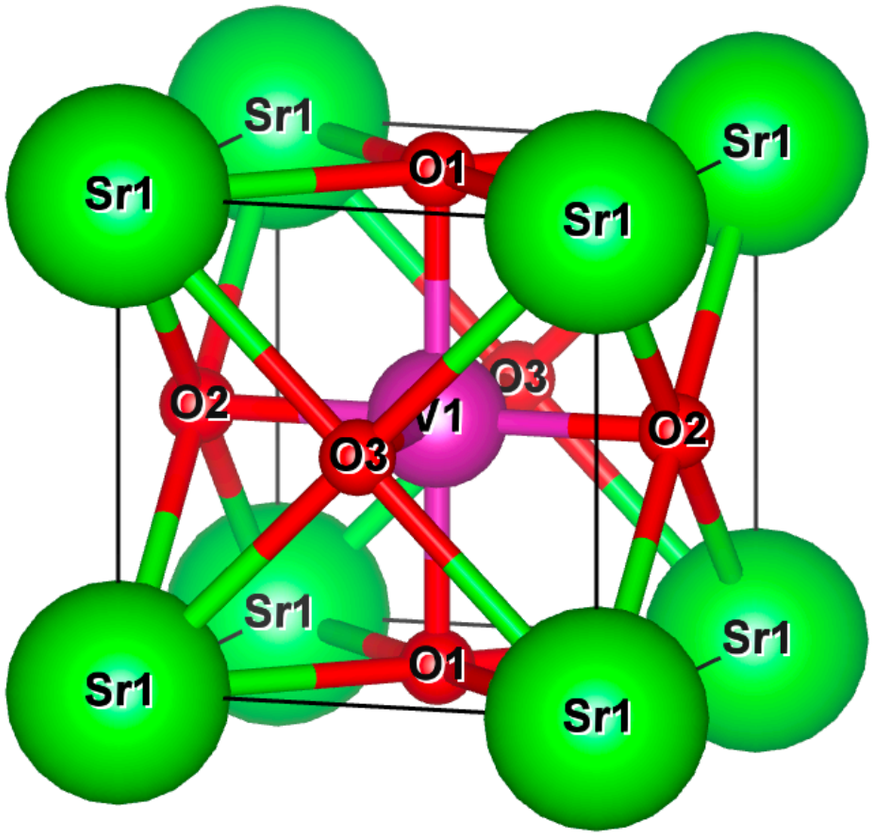


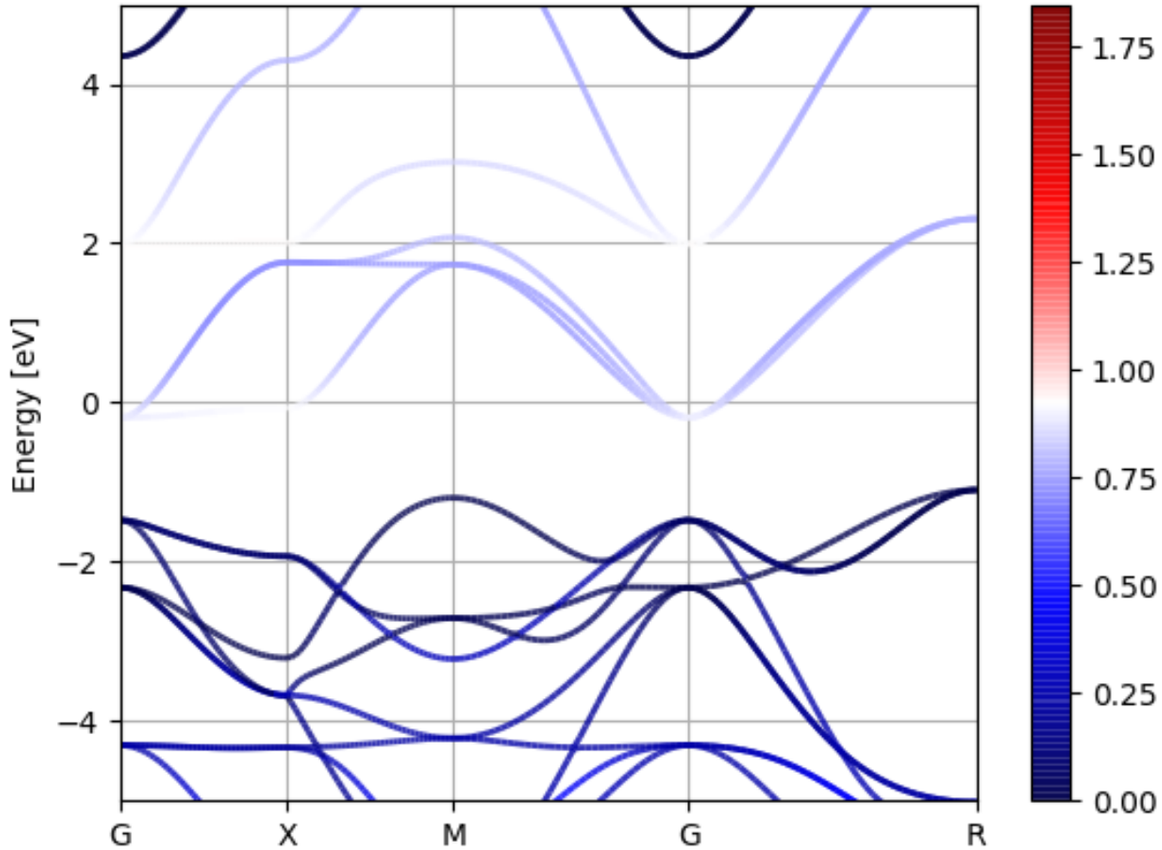
Spin ↑ or spin ↓

Band structure of $SrVO_3$ for a colinear spin projection (Red-spin up, Blue-spin down)

```
pyprocar.bandsplot('PROCAR',abinit_output='SrVO3.out', cmap='seismic',mode='parametric',spin='1' )
```

Atom projected bands

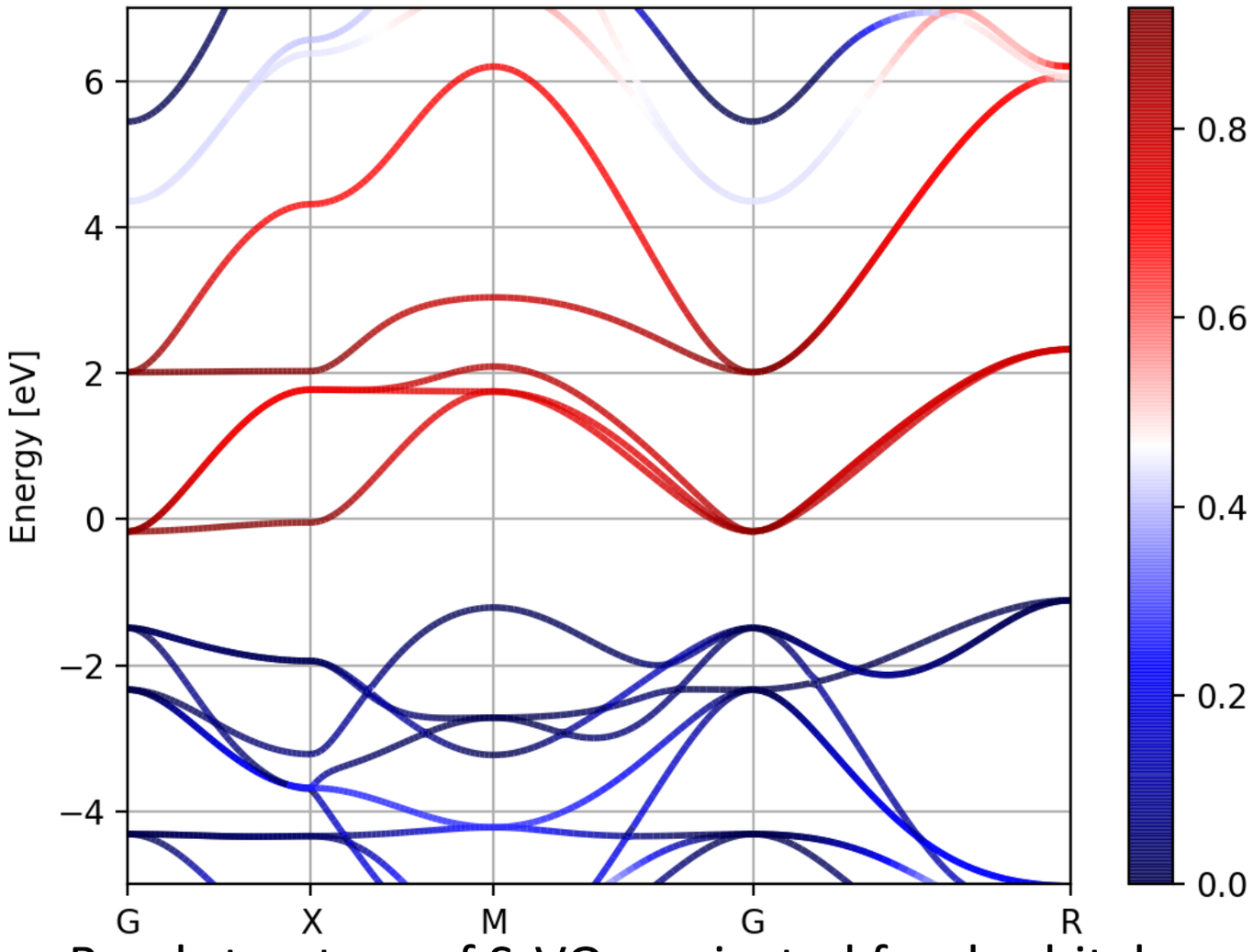




Band structure of $SrVO_3$ projected for V atom

```
pyprocar.bandsplot('PROCAR', abinit_output='SrVO3.out',cmap='seismic',mode='parametric',atoms=[1])
```

orbital projected bands



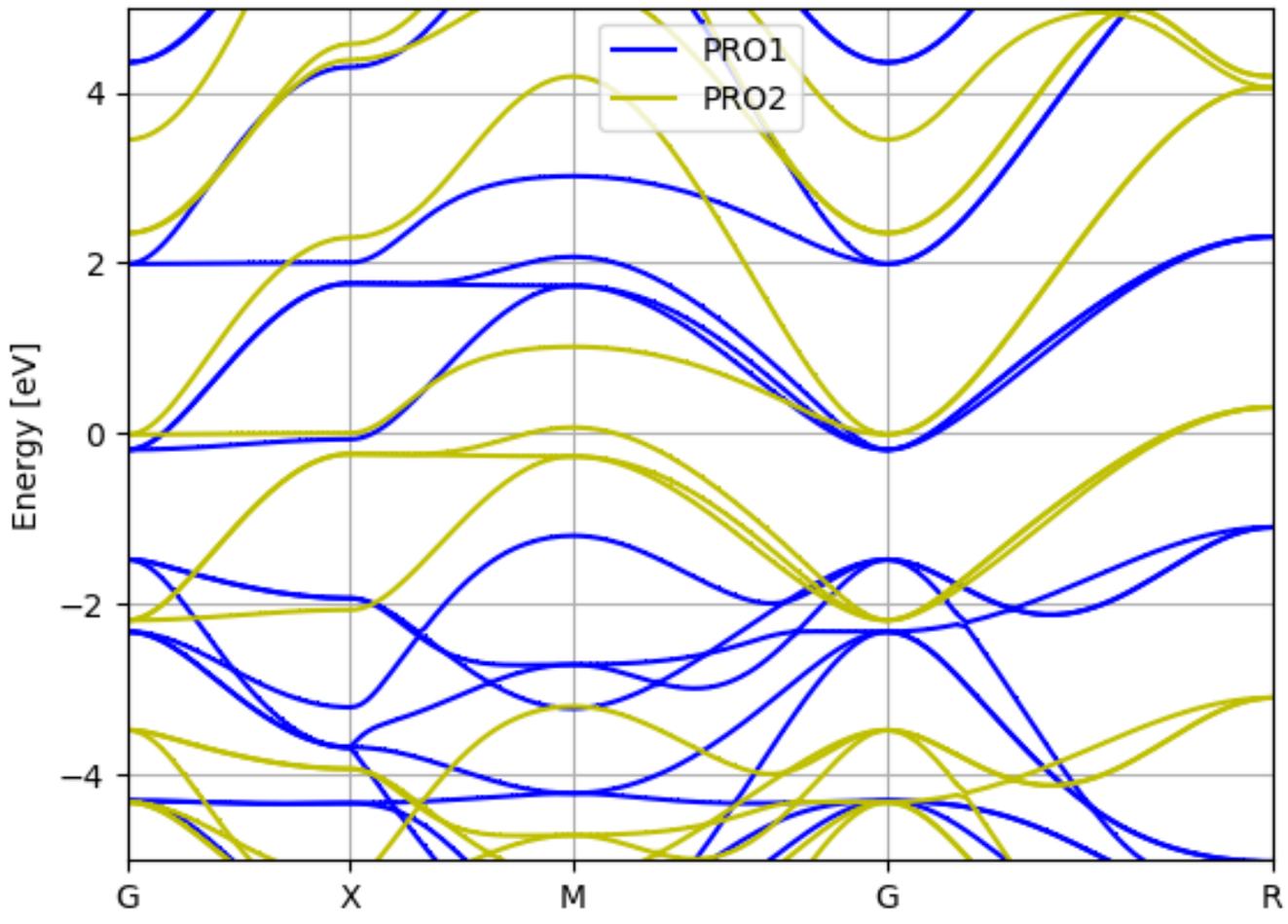
Band structure of $SrVO_3$ projected for d orbitals

Both e_g and t_{2g} contributions considered here.

```
pyprocar.bandsplot('PROCAR',abinit_output='SrVO3.out,cmap='seismic',mode='parametric',orbitals=[4,5,6,7,8])
```

Comparing bands

pyprocar.bandscompare()



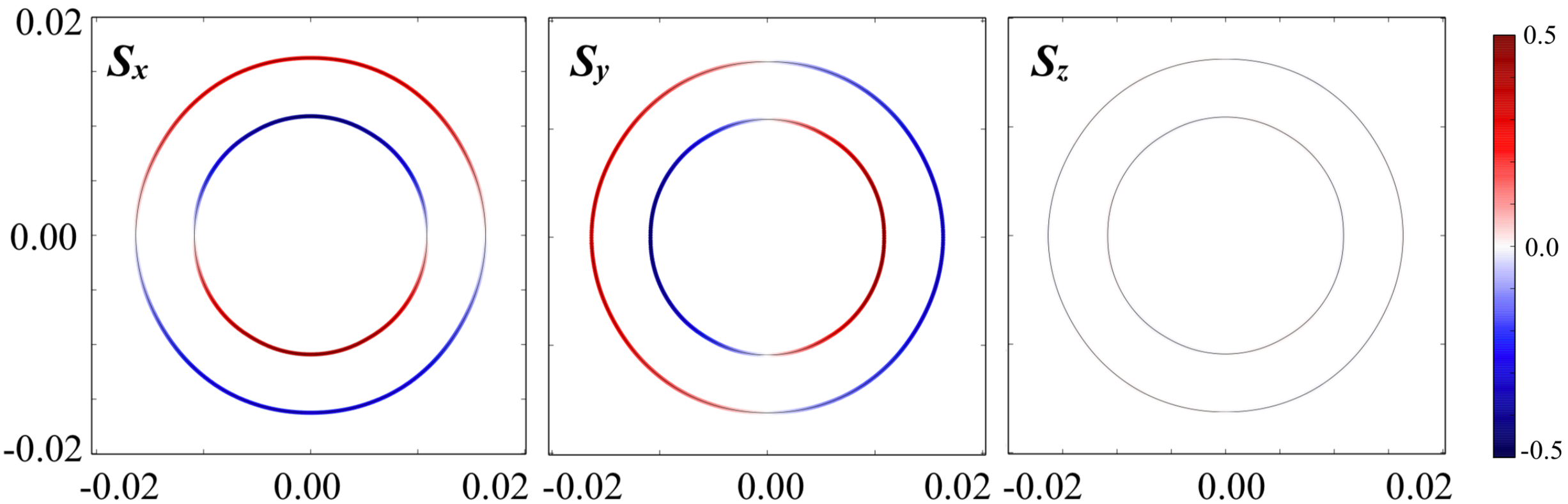
Plot different bands on the same plot

E.g- $SrVO_3$ with a shift in Fermi energy

```
pyprocar.bandscompare('PROCAR1','PROCAR2', abinit_output='SrVO3.1.out' abinit_output2='SrVO3.2.out',
cmap='seismic',mode='parametric')
```

spin texture/ 2d fermi surface

pyprocar.fermi2D()

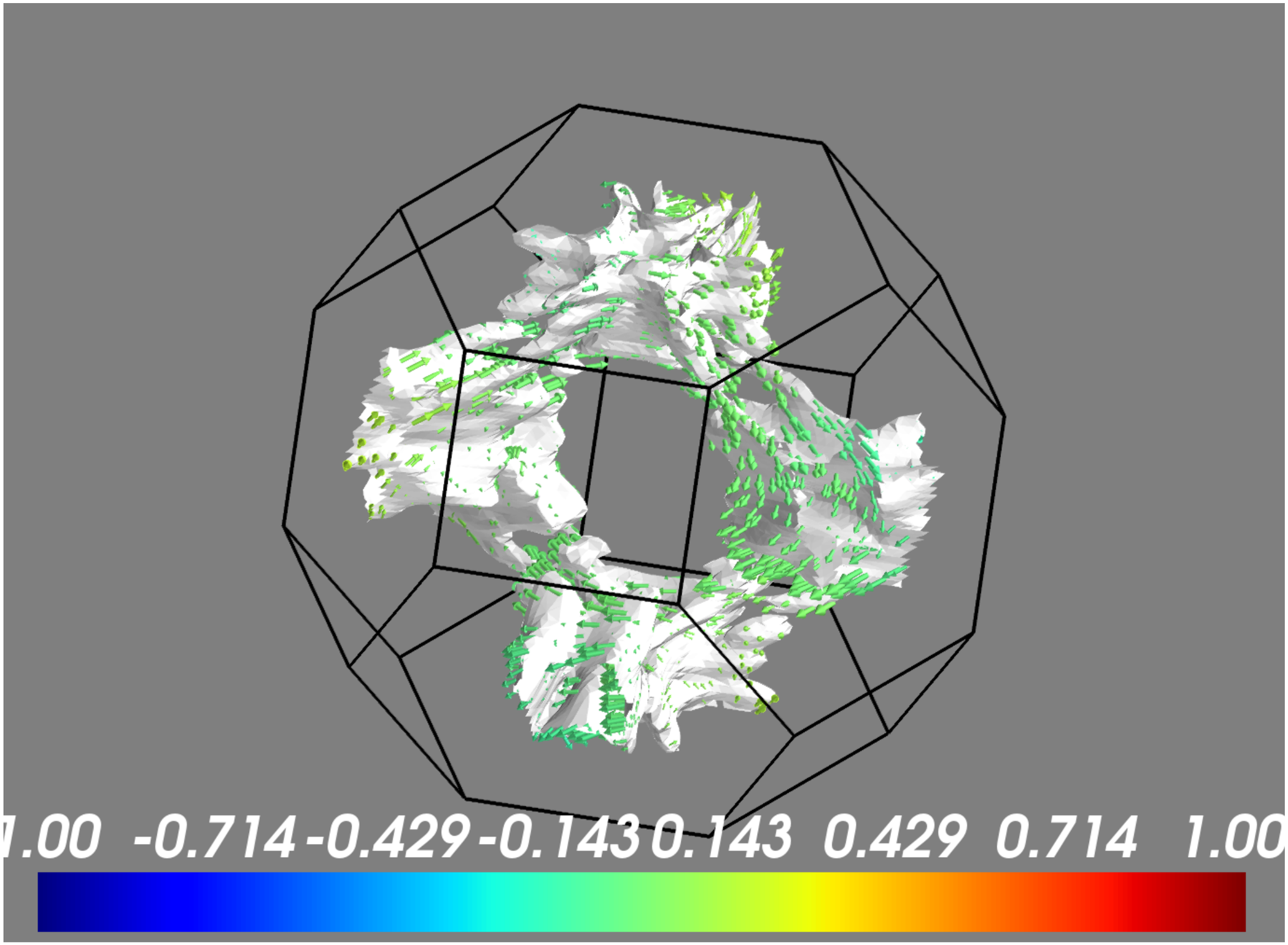


Spin texture in BiSb monolayer in a constant energy surface on the k_x-k_y plane for S_x, S_y and S_z spin projections. Useful to investigate Rashba spin splitting.

```
pyprocar.fermi2D('PROCAR',abinit_output='SrVO3.out', energy=-1.0,st=True,noarrow=True, spin=2 )
```

3d fermi surface

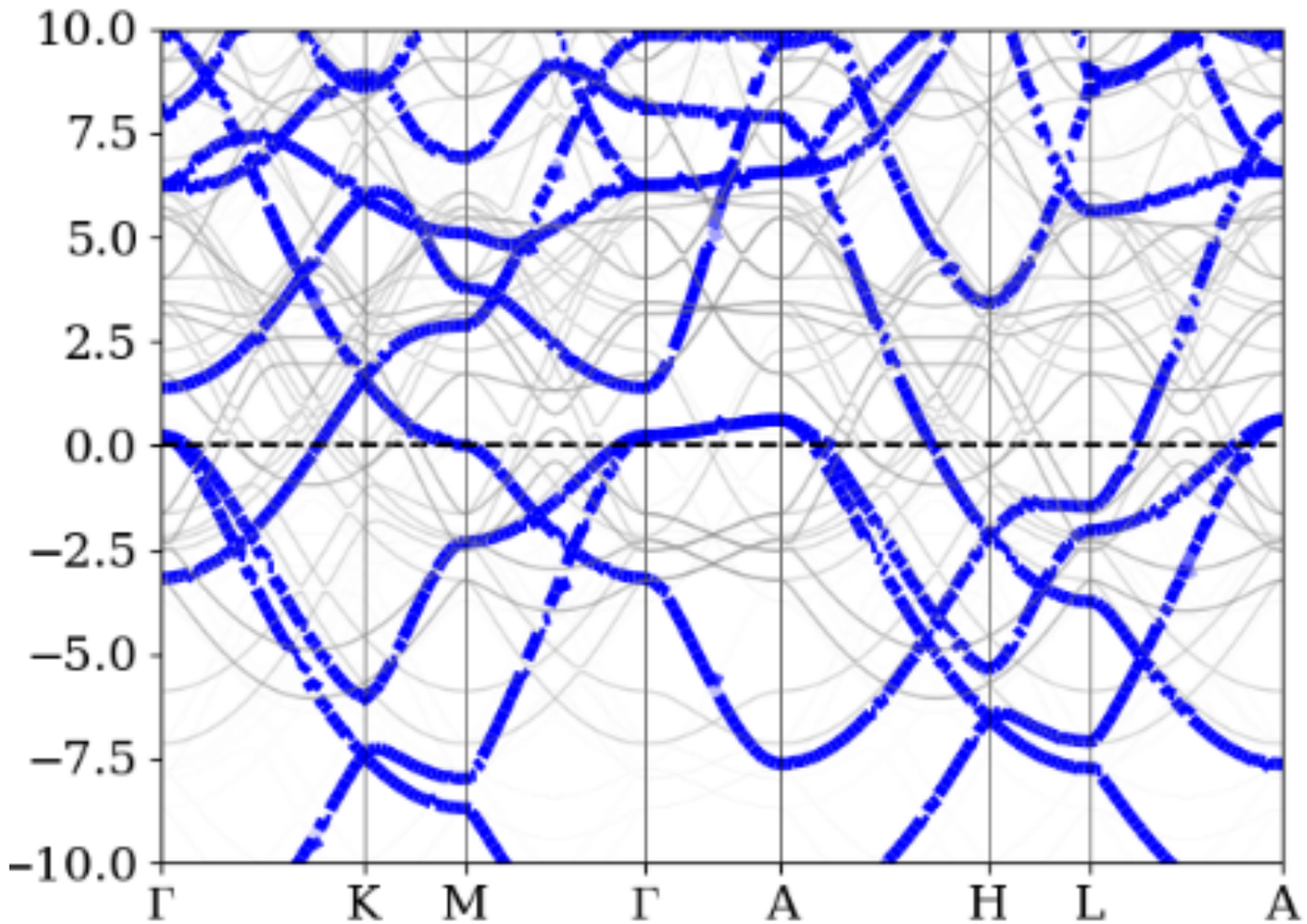
pyprocar.fermi3D()



Adding Spin Texture

Band unfolding

pyprocar. unfold()




```
pyprocar.unfold('PROCAR',
poscar='POSCAR',abinit_output='MgB2.out',supercell_matrix=np.diag([2,2,2]),shift_efermi=True,show_band=
True)
```

Band unfolding requires a special type of PROCAR file which includes the phase of the wave function

```
PROCAR lm decomposed + phase
# of k-points:  160        # of bands:   50        # of ions:    5

 k-point     1 :    0.00000000 0.00000000 0.00000000     weight = 0.00625000

band     1 # energy  -29.05624010 # occ.  2.00000000

ion      s     py     pz     px    dxy    dyz    dz2    dxz  x2-y2    tot
    1  0.972  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.972
    2  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
    3  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
    4  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
    5  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
tot    0.974  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.974
ion            s              py             pz             px             dxy            dyz            dz2            dxz            dx2-y2
    1 -0.437 -0.883   0.000  0.000   0.000 -0.000  -0.000 -0.000   0.000 -0.000  -0.000 -0.000  -0.000 -0.000  -0.000 -0.000   0.000 -0.000   0.971
    2 -0.007 -0.015   0.000 -0.000   0.000 -0.000   0.000  0.000   0.000 -0.000  -0.000  0.000   0.000  0.000  -0.000 -0.000  -0.000  0.000   0.000
    3 -0.009 -0.018  -0.000 -0.000   0.000  0.000  -0.000 -0.000   0.000  0.000   0.000  0.000   0.000  0.000   0.000  0.000   0.000  0.000   0.000
    4 -0.009 -0.018  -0.000  0.000   0.000 -0.000   0.000 -0.000   0.000  0.000   0.000  0.000   0.000  0.000   0.000  0.000   0.000  0.000   0.000
    5 -0.009 -0.018  -0.000 -0.000   0.000  0.000   0.000 -0.000   0.000  0.000   0.000  0.000   0.000  0.000   0.000  0.000   0.000  0.000   0.000
charge 0.973          0.000          0.000          0.000          0.000          0.000          0.000          0.000          0.000          0.973
```

Real and imaginary parts of projection

mdwc: Molecular Dynamics with Constraints

Available @: https://github.com/romerogroup/mdwc

•The molecular dynamics with constraints (mdwc) package is a command line open source python program.

•constraint molecular dynamics with:
•NPT (keeping pressure constant with the Parrinello Rahman lagrangian, and keeping the temperature constant with the Nose thermostat)

•NVT (keeping the temperature constant with the Nose thermostat).

Available constrains:
•Bond distances
•Bond angles
•Atomic positions
•Lattice parameters
•Angles between lattice vectors
•Volume of the unit cell.

The MD_suite Fortran library

URL: https://molecular-dynamics-with-constraints.github.io/text/50.MD_suite_fortran_lib.html

(a). Generating the library:

cd into mdwc/MD_suite_fortran_lib directory. Create the objects by compiling using gfortran

```
gfortran -c MD_suite.f90
```

Now create the library

```
ar crv libMD_suite.a MD_suite.o
```

This gives the library, libMD_suite.a which can be linked to an external fortran program.

(b). Linking the library to an external fortran program:

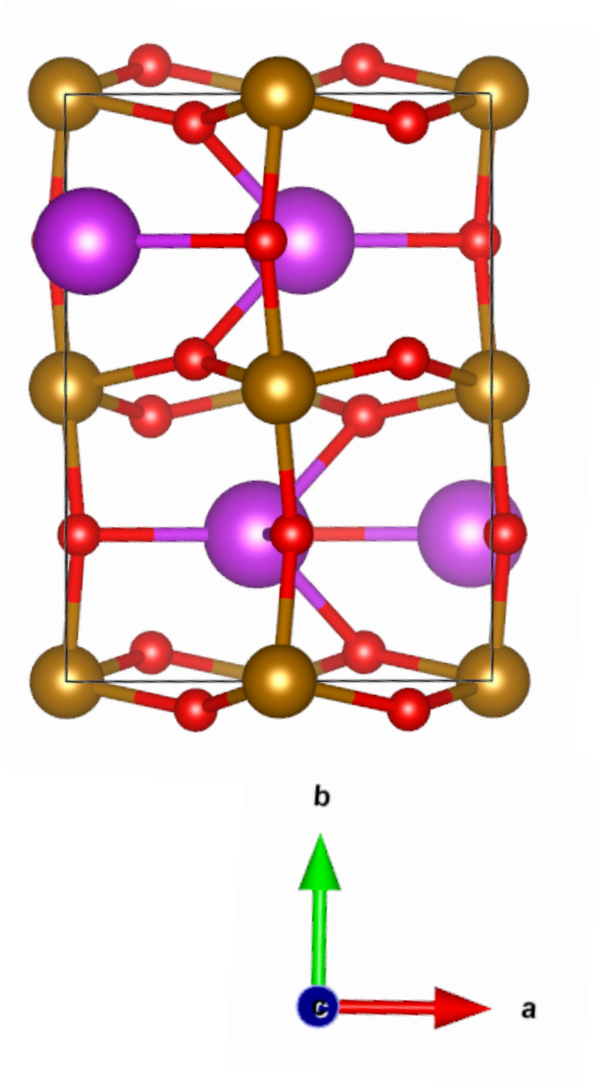
```
gfortran main.f90 -L<path to library> -lMD_suite -o main
```

Example: $FeBiO_3$

(a) No constraints

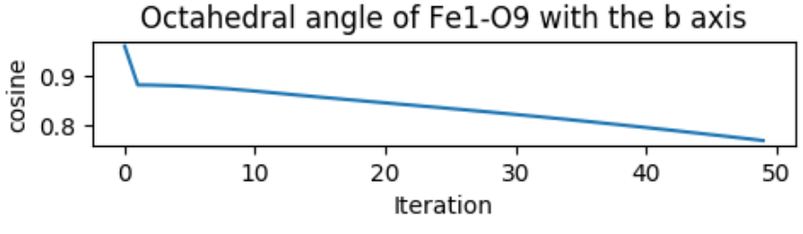
Input parameters: FeBiO3.md

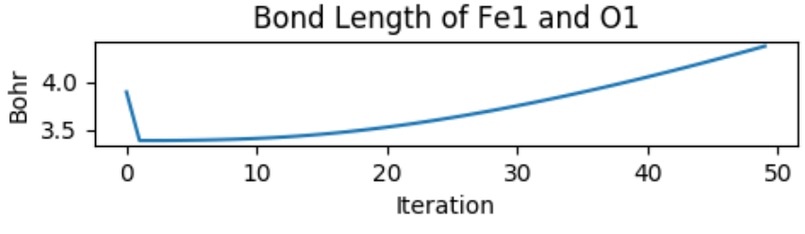
```
Qmass           0.5                     #mass of the thermostat
temp_cons       600                     #temp in K
#temp_line      250.6, 275.6,           #linear control temperature
#temp_plat      250.0, 280.0, 290.5,    #plateus control temperature
#temp_step      5,   2,   3,            #temperature steps
bmass           10.0                    # mass of the barostat
Pressure        0.00025                 # pressure in hatree/bohr^3
dt              0.1                     #time in femtoseconds
correct_spteps 8
md_steps       5
abinit_steps  100

number_atom_fix_constrains 0   #0 is off, 1 is on
atom_fixed_constrains 1, 1, #costrains over atoms 1 and 2
atom_fixed_position 14.96 4.57 0.00, 14.96 0.00 4.77,
atom_fix_coordinate 1 1 1, 1 1 1,

number_bond_constrains 0
bond_constrains 1 2, 1 3, #costrains between atoms 1 and 2, 1 and 3
bond_distance 6.76275, 6.89948

number_angle_constrains 0
angle_constrains 1 2 3, 4 5 6,#constrain of angle formed 3 4 5 with vertex at 3
value_cosine_angle 0.532, 0.915

number_cell_parameter_constrain 0
cell_parameter_constrain 1, 2,#constrain over the length of the first cell vector
cell_parameter_value 19.9400, 9.1400

number_cell_angle_constrain 0
cell_angle_constrain 1 2, 2 3, #constrain over the angle between cell vectors of 1,2 and 2,3
value_cosine_cell_angle 0.0000, 0.0000

volume_constrain 0
volume_value 248.464  # in Angstrom cube
```

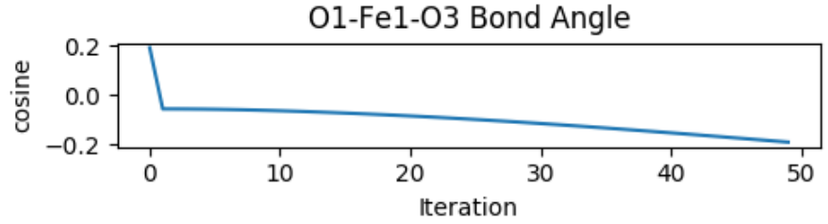


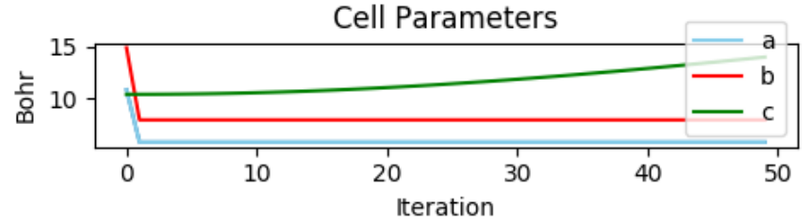
Example: Constraining (a,b) plane

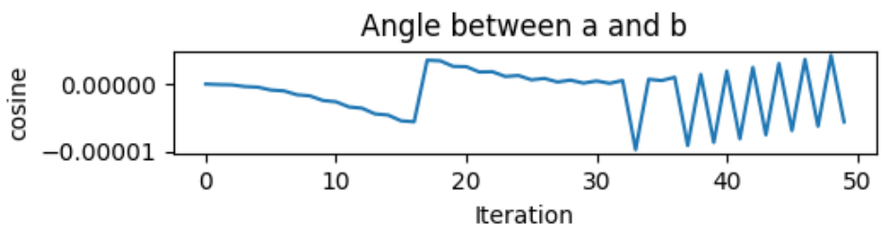
```
number_cell_parameter_constrain 2
cell_parameter_constrain 1, 2,#constrain over the length of the first cell vector
cell_parameter_value 5.72,7.89

number_cell_angle_constrain 1
cell_angle_constrain 1 2, #constrain over the angle between cell vectors of 1,2
value_cosine_cell_angle 0.0000
```

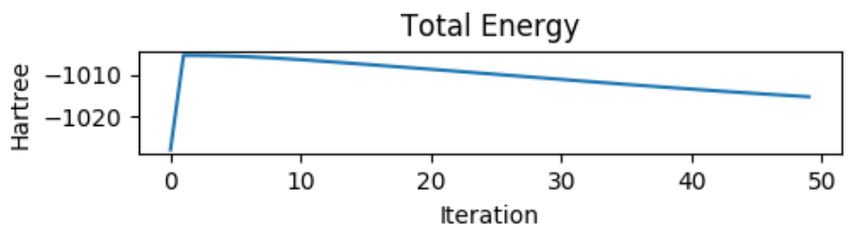


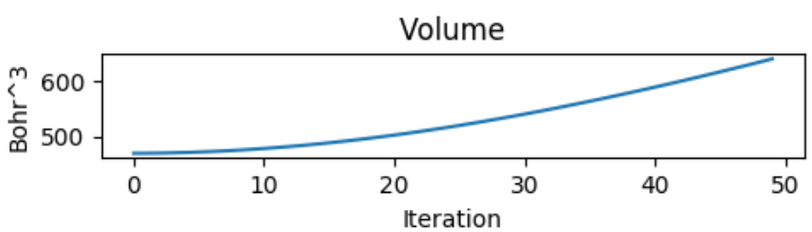


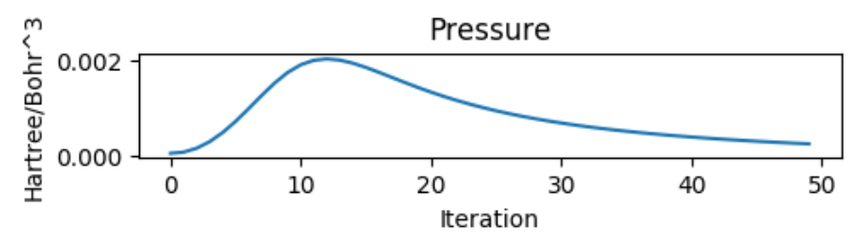












Merci

Danke

Gracias

Grazie