Pivotpy Tutorial

Vasp input/output Processing Tool

Abdul Saboor

Talk Layout

- Installation and guidelines
- Input (POSCAR, KPATH)
  - POSCAR downloading from Materials Project
  - Brillouin Zone Plots
  - Lattice Plots
  - HSK path tracing with KPathApp
  - K-mesh for Fermi Surface
- Output (vasprun.xml, LOCPOT, EIGENVAL)
  - Extracting Data
  - Bandstructure
  - Density of States
  - VasprunApp
  - Splitting large vasprun files
  - Local Electrostatic Potential/Magnetization from LOCPOT
  - Parsing text files like EIGENVAL

Installation and guidelines

Run in a system terminal (Preferred in a virtual environment) Demo on venv

```
> pip install pivotpy
> python
```

> For doing everything possible, use Jupyter Lab >= 3

```
import pivotpy as pp
print(pp.__all__)
help(pp.download_structure)
```

['docs', 'example_notebook', 'create_colormap', 'get_kpath', 'str2kpath', 'fancy_quiver3d', 'rotation', 'generate_summary', 'VasprunApp', 'KPathApp', 'set_dir', 'get_child_items', 'transform_color', 'interpolate_data', 'split_vasprun', 'xml2dict', 'iplot2html', 'plt2html', 'plt2text', 'show', 'savefig', 'append_axes', 'download_structure', 'parse_text', 'POSCAR', 'LOCPOT', 'get_axes', 'Vasprun', '_show', '_savefig']

In IPython terminal or jupyter notebook

```
pp.download_structure?
```

gives you short help and

```
pp.download_structure??
```

gives you detailed help on function like one on right side.

```
def download_structure(formula, mp_id=None, max_sites=No
  """Download structure data from Materials project web
    - **Parameters**
        - formula: chemical formula of the material.
        - mp_id: Materials project id of material.
        - max_sites: maximum number of sites in structu
        - min_sites: minimum number of sites in structu
    > max_sites and min_sites are used to filter the nu
    - **One Time API Key**
        - api_key: API key from Materials project websi
        - save_key: Save API key to file. You can save
    - **Return**
        - structure: Structure data containing attribut
                    Each item in list has attributes `
    """
  mp = sio.InvokeMaterialsProject(api_key= api_key)
  mp.request(formula=formula,mp_id=mp_id,max_sites=max_
  if save_key and isinstance(api_key,str):
    mp.save_api_key(api_key)
  if mp.success:
    return vp.Dict2Data({'poscars':mp.poscars,'cifs':mp
  else:
    return print('Connection was not sccessful. Try aga
```


POSCAR from Materials Project

["Cmcm: mp-1097832", "P6/mmm: mp-568806", "P6/mmm: mp-1040425", "Fmmm: mp-937760", "P6/mmm: mp-990448", "Fd-3m: m

C # [P6/mmm] Generated by PivotPy using Materials Project Database.
2.46841603
1.0000000000000000 0.0000000000000000 0.0000000000000000
-0.5000000129008653 0.8660253963361205 0.0000000000000000
0.0000003438991 0.0000000005956507 4.0507993361019539
C
2
Direct
0.33333300 0.66666700 0.00000000 C
0.66666700 0.33333300 0.00000000 C

```
st = pp.download_structure('C',max_sites=2,min_s
s_p = st.poscars[4]
s_p.write('POSCAR') # Save to file
global poscar
poscar = pp.POSCAR(content=s_p.content) # Initial
```

POSCAR class

POSACR class to contain data and related methods

POSCAR.**bring_in_bz**(kpoints)

POSCAR.**bring_in_cell**(points)

POSCAR.**fix_sites**(tol=0.01, eqv_sites=True, translate=None)

POSCAR.**get_bz**(loop=True, digits=8, primitive=False)

POSCAR.**get_cell**(loop=True, digits=8)

POSCAR.**get_kmesh**(n_xyz=[5, 5, 5], weight=None, ibzkpt=None, outfile=None)

POSCAR.**iplot_bz**(fill=True, color='rgba(168,204,216,0.4)', background='rgb(255,255,255)', vname='b', alpha=0.4, ortho3d=True, fig=None)

POSCAR.**iplot_cell**(fill=True, color='rgba(168,204,216,0.4)', background='rgb(255,255,255)', vname='a', alpha=0.4, ortho3d=True, fig=None)

POSCAR.**iplot_lat**(sizes=10, colors='blue', bond_length=None, tol=0.1, eps=0.01, eqv_sites=True, translate=None, line_width=4, edge_color='black', fill=False, alpha=0.4, ortho3d=True, fig=None)

POSCAR.**join**(other, direction='z', tol=0.01)

POSCAR.**rotate**(angle_deg, axis_vec)

POSCAR.**scale**(scale=(1, 1, 1), tol=0.01)

POSCAR.**set_bz**(primitive=False, loop=True, digits=8)

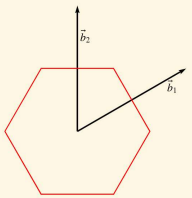
POSCAR.**splot_bz**(ax=None, plane=None, color='blue', fill=True, vectors=True, v3=False, vname='b', colormap='plasma', light_from=(1, 1, 1), alpha=0.4)

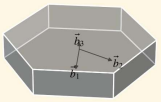
POSCAR.**splot_cell**(ax=None, plane=None, color='blue', fill=True, vectors=True, v3=False, vname='a', colormap='plasma', light_from=(1, 1, 1), alpha=0.4)

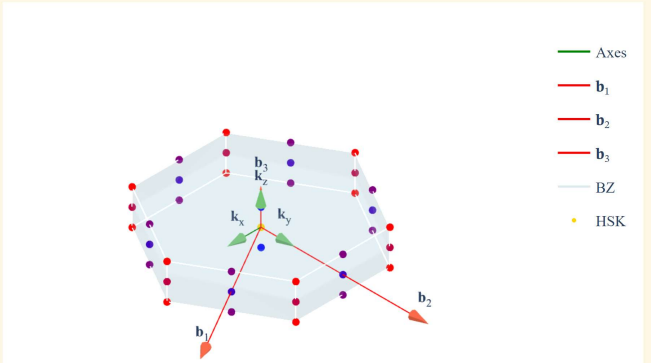
POSCAR.**splot_lat**(sizes=50, colors=[], colormap=None, bond_length=None, tol=0.1, eps=0.01, eqv_sites=True, translate=None, line_width=1, edge_color=(1, 0.5, 0, 0.4), vectors=True, v3=False, plane=None, light_from=(1, 1, 1), fill=False, alpha=0.4, ax=None)

POSCAR.**write**(sd_list=None, outfile=None, overwrite=False)

Brillouin Zone Plots

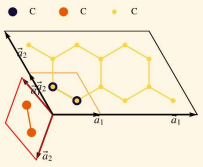


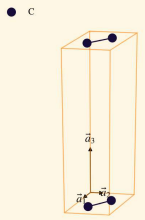


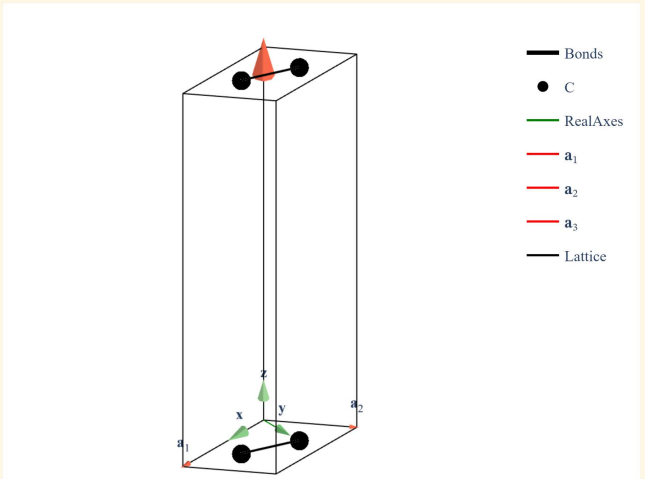



```
ax = poscar.splot_bz(colormap='bone',alpha=0.4,light_from=(5,0.5,10),
ax1 = poscar.splot_bz(colormap='RGB',plane='xy',color='red')
fig = poscar.iplot_bz()
```


Lattice Plots

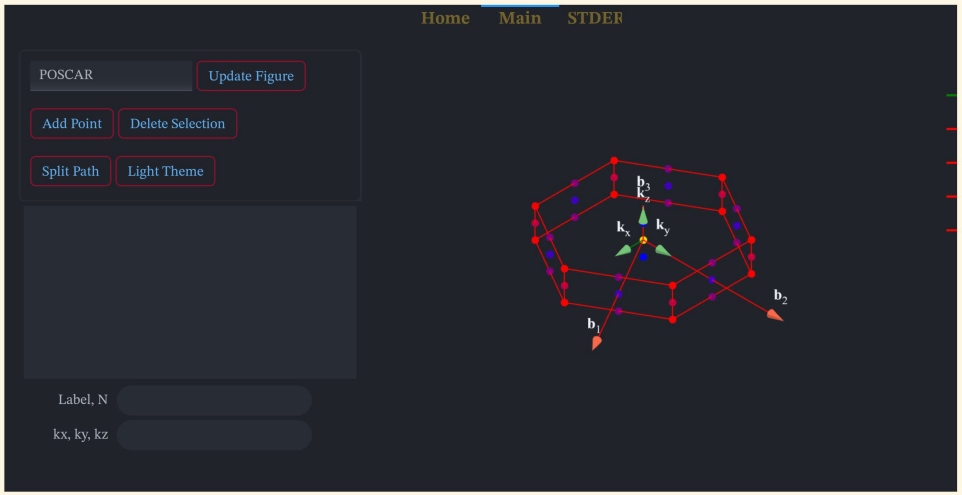







```
ax = poscar.splot_lat(colormap='inferno',alpha=0.4,light_from=(5,0.5,
ax1 = poscar.splot_lat(colormap='inferno',plane='xy')
poscar.rotate(130,[0,0,1]).splot_lat(colormap='RGB',plane='xy',ax=ax
poscar.scale([3,2,1]).splot_lat(sizes=[10],colormap='inferno_r',plan
fig = poscar.iplot_lat(colors=['black','orange'])
```


KPath Tracing App



ka = KPathApp()

After interaction with app, you can run following functions to save data/plots etc.

ka.**splot**(**kwargs)

Same as `pp.splot_bz`

except it also plots path on BZ. `kwargs`

are passed to `pp.splot_bz`

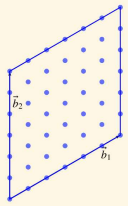
Demo on splot

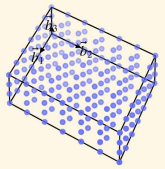
ka.**iplot**()

Returns disconnected current plotly figure

ka.**get_kpath**(n=5, weight=None, ibzkpt=None, outfile=None)

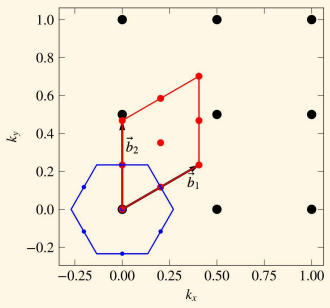
K-mesh for Fermi Surface





```
poscar.get_kmesh(n_xyz=[7,7,4],outfile='kmesh')
kpts = np.loadtxt('kmesh',skiprows=3,usecols=[0,1,2])
poscar.set_bz(primitive=True)
ax1 = poscar.splot_bz(color='k',colormap=None,fill=False)
ax2 = poscar.splot_bz(plane='xy',colormap='RGB')
k_in = poscar.bring_in_bz(kpts)
ax1.scatter(*k_in.T,s=10)
ax2.scatter(*k_in[:,:2].T,s=10)
```


Coordinates Translation

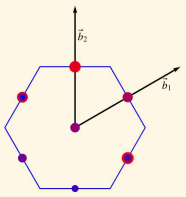


Code

```
poscar.set_bz(primitive=True) # for having a primitive B
ax = poscar.splot_bz(plane='xy',color='red')
to_p = poscar.bring_in_bz(points)

poscar.set_bz(primitive=False) # for having a regular BZ
poscar.splot_bz(plane='xy',ax=ax)
to_r = poscar.bring_in_bz(points)

ax.scatter(*points[:,:2].T,color='k',s=30)
ax.scatter(*to_p[:,:2].T,color='red',s=15)
ax.scatter(*to_r[:,:2].T,color='blue',s=4)
```

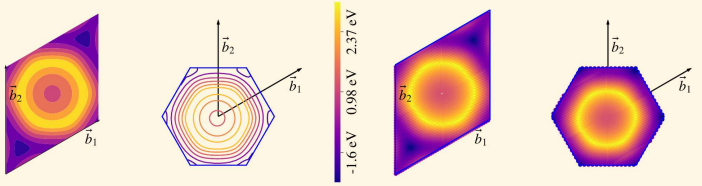
Inside Regular BZ

Points can overlap



Translation Example

Energy Contours in Graphene



▶ Show Code

▼Show Code

```
import matplotlib.pyplot as plt , pivotpy as pp, numpy as np
import matplotlib.tri as tri

axs = pp.get_axes(ncols=5,figsize=(9,2.3),widths=[5,5,0.2,5,5])
evr = pp.Vasprun('E:/Research/graphene_example/ISPIN_1/dos/vasprun.xml').data

kpoints = np.concatenate([evr.kpoints, -evr.kpoints]) # negative side
poscar = pp.POSCAR(_other_data=evr.poscar)
poscar.set_bz(primitive=True)
out1 = poscar.bring_in_bz(kpoints) + 0.002 #Just fun translate
poscar.splot_bz(plane='xy',ax = axs[0],color=(1,1,1,0)).set_axis_off()
poscar.splot_bz(plane='xy',ax = axs[3]).set_axis_off()

poscar.set_bz(primitive=False)
out2 = poscar.bring_in_bz(kpoints)
poscar.splot_bz(plane='xy',ax = axs[1]).set_axis_off()
poscar.splot_bz(plane='xy',ax = axs[4]).set_axis_off()

# Energy as color on BZ.
en = evr.bands.evals[:,8].ravel()
en = np.concatenate([en,en])
s_en = (en-np.min(en))/(np.max(en)-np.min(en)) # bring to range [0,1] for colors
c = plt.cm.get_cmap('plasma')(s_en) # get colors for scatter
labels = [str(l) + ' eV' for l in np.round(np.min(en)+(np.max(en)-np.min(en))*np.array([1/6,3/5,5/6]),2)]
tri1 = tri.Triangulation(out1.T[0], out1.T[1])
tri2 = tri.Triangulation(out2.T[0], out2.T[1])
axs[0].tricontourf(tri1, en, levels=np.linspace(-2.6, 3.4, 8), cmap='plasma')
axs[1].tricontour(tri2, en, levels=np.linspace(-2.6, 3.4, 8), cmap='plasma',linewidths=[0.8])
axs[3].scatter(out1[:,0],out1[:,1],s=4,c=c)
axs[4].scatter(out2[:,0],out2[:,1],s=4,c=c)

axs[2].add_colorbar(cax=axs[2],cmap_or_clist='plasma', ticklabels=labels,vertical=True)
```


High Symmerty K-Path

Automatically generated using PivotPy with HSK-INDS = [0, 5, -1], LABELS = ['A', 'B', 'C'], SEG-INDS = []

10

Reciprocal Lattice

0.0000000000	0.0000000000	0.0000000000	0.100000
0.2500000000	0.2500000000	0.2500000000	0.100000
0.5000000000	0.5000000000	0.5000000000	0.100000
0.7500000000	0.7500000000	0.7500000000	0.100000
1.0000000000	1.0000000000	1.0000000000	0.100000
1.0000000000	1.0000000000	1.0000000000	0.100000
1.2500000000	1.2500000000	1.2500000000	0.100000
1.5000000000	1.5000000000	1.5000000000	0.100000
1.7500000000	1.7500000000	1.7500000000	0.100000
2.0000000000	2.0000000000	2.0000000000	0.100000

```
pp.get_kpath([[0,0,0],[1,1,1],[2,2,2]],n=3,labels=['A','B','C'])#,outfile='KPATH')
```



```
ticks = pp.sio.read_ticks('KPATH')
```

⟶ {'ktick_inds': [0, 5, -1], 'ktick_vals': ['A', 'B', 'C'], 'kseg_inds': []}

Automatically generated using PivotPy with HSK-INDS = [0, -1], LABELS = ['G', 'M'], SEG-INDS = []

2

Reciprocal Lattice

0.0000000000	0.0000000000	0.0000000000	0.500000
1.0000000000	1.0000000000	1.0000000000	0.500000

```
pp.str2kpath('''
0 0 0 G 2
1 1 1 M
''')
```


Bandstructure/DOS Data Import/Export

pivotpy.api.**Vasprun**(path=None, skipk=None, elim=[], shift_kpath=0, try_pwsh=True, data_str=None)

- All plotting functions that depend on `export_vasprun` are joined under this class and renamed.
  - **Main Parameter**
    - path: str: path/to/vasprun.xml. Auto picks in CWD.
      `.json/.pickle` files are also accepted if they are saved previously using `to_json/to_pickle` methods.
      `.json` file is useful to load data in other languages.
  - **Optional Parameters** (only useful if path is `vasprun.xml` file)
    - skipk : int: Skip initial kpoints.
    - elim : list: Energy range e.g. [-5,5].
    - shift_kpath: float: Shift in kpath values for side by side plotting.
    - try_pwsh : bool: True by default, tries to load data exported using Powershell's `Vasp2Visual.Export-Vasprun` command.
    - data_str : json/pickle: Data in json/pickle format saved from `to_json/to_pickle` methods.
  - **Attributes and Methods**
    - data : Exported data from given file. This has it's own attributes as well to save as json/pickle etc.
    - to_json : Saves data in `.json` file. Useful for transport to other langauges.
    - to_pickle : Saves data in `.pickle` file. Useful for fast reload in python.
    - splot_[...] : Plots data using `sp.splot_[...]` functions.
    - iplot_[...] : Plots data using `sp.iplot_[...]` functions.

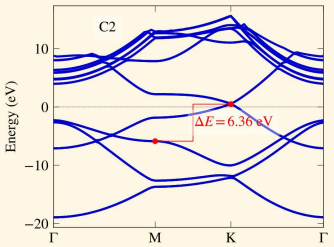
> Tip: If KPOINTS file is generated by this module, ticks on kpath are auto-picked.

```
import pivotpy as pp
vr = pp.Vasprun(path='path/to/vasprun.xml')
print(list(vr.data.keys()),end='\n\n')
print('Fields: ',vr.data.sys_info.fields)
print('INCAR: ',vr.data.sys_info.incar)
```

```
Loading from PowerShell Exported Data...
['sys_info', 'dim_info', 'kpoints', 'kpath', 'bands', 'tdos', 'pro_bands', 'pro_dos', 'poscar']

Fields:  ['s', 'py', 'pz', 'px', 'dxy', 'dyz', 'dz2', 'dxz', 'x2-y2']
INCAR:  Data(
  SYSTEM = C2
  PREC = high
  ALGO = N
  LSORBIT = T
  NELMIN = 7
  ISMEAR = 0
  SIGMA = 0.10000000
  LORBIT = 11
  GGA = PS
)
```


Plotting Functions

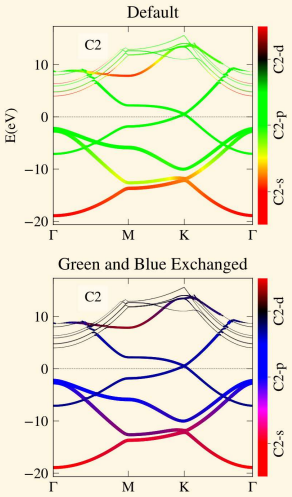
vr.**splot_bands**(ax=None, **kwargs)



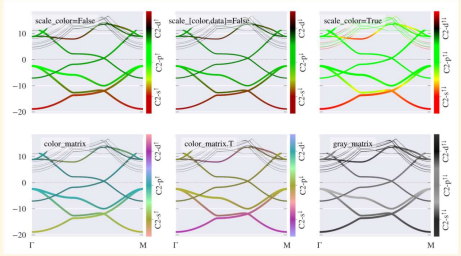

```
ax = vr.splot_bands()
delta = vr.get_en_diff(b1_i=4,b2_i=6,k1_i=30,k2_i=60)
vr.splot_en_diff(delta.coords,ax,color='r',lw=0.5)
ax.add_text(delta.coords[:,0].mean()+0.2, delta.coords[:,1].mean(),f'$\Delta E = {delta.de:8.2f}$ eV',tran
```


Plotting Functions

vr.**splot_rgb_lines**(elements=[[], [], []], orbs=[[], [], []], labels=['', '', ''], ax=None, **kwargs)



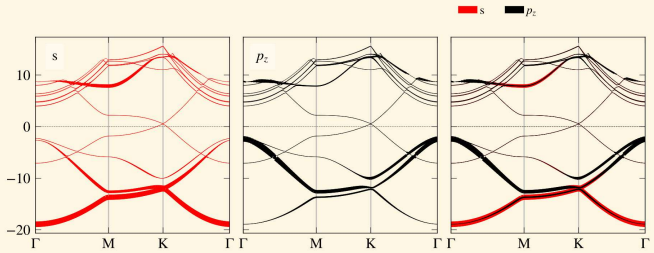

```
ax1, ax2 = pp.get_axes((3,5),nrows=2, sharey=True
vr.splot_rgb_lines(ax=ax1)
ax1.set_ylabel('E(eV)')
ax1.set_title('Default')
vr.splot_rgb_lines(ax=ax2, color_matrix = pp.rbg_
ax2.set_title('Green and Blue Exchanged')
```



link: splot_rgb_lines

Plotting Functions

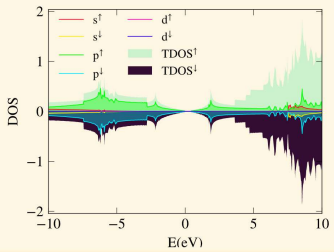
vr.**splot_color_lines**(elements=[[0]], orbs=[[0]], labels=['s'], axes=None, **kwargs)




```
axs = pp.get_axes(nrows=1,ncols=3,figsize=(7,2.5),sharey=True,sharex=True,hspace=0.1,wspace=0.07)
args_dict=dict(elements=[[0,1],[0,1]],orbs=[0,1],labels=['s','$p_z$'],max_width=10)
vr.splot_color_lines(axes=axs[0:2],**args_dict,left=0.06,colormap='flag',showlegend=False,xytxt=(0.1,0.9))
vr.splot_color_lines(axes=axs[2],**args_dict,left=0.06,colormap='flag',showlegend=True);
for ax in axs:
  ax.grid(which='major',axis='x')
```

Plotting Functions

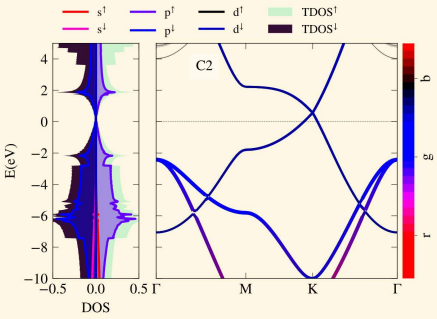
vr.**splot_dos_lines**(elements=[[0]], orbs=[[0]], labels=['s'], ax=None, **kwargs)




```
vr1 = pp.Vasprun('E:/Research/graphene_example/ISPIN_2/dos/vasprun.xml')
ax = vr1.splot_dos_lines(elements = [range(2),range(2),range(2)], orbs = [0,range(1,4),range(4,9)],labels=['s
                         query_data = {'s':[range(2),[0]],'p':[range(2),range(1,4)],'d':[range(2),range(4,9)]},sl
ax.add_legend(anchor=(0,0.6),ncol=2)
ax.set(xlim=[-10,10],xlabel='E(eV)',ylabel='DOS')
```


Plotting Functions

Bandstructure & DOS together



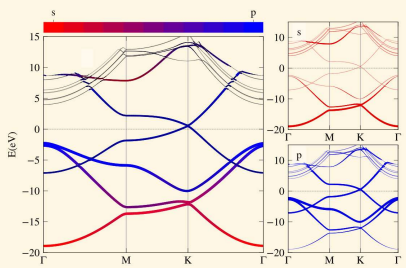

```
vr2 = pp.Vasprun('E:/Research/graphene_example/ISPIN_2/bands/vasprun.xml')
ax1, ax2 = pp.get_axes((4.5,3),ncols=2,widths=[1,3],sharey=True,wspace=0.1)
ax1.set(ylim=[-10,5],ylabel='E(eV)',xlim=[-0.5,0.5],xlabel='DOS')
vr2.splot_rgb_lines(ax=ax2,color_matrix=[[1,0,0],[0,0,1],[0,1,0]],colorbar=False)
ax2.add_colorbar(N=30)
vr1.splot_dos_lines([range(2),range(2),range(2)],[0,range(1,4),range(4,9)],['s','p','d'],
  ax=ax1,vertical=True,colormap='RGB_m',showlegend=False,linewidth=1.5)
ax1.add_legend(ncol=4,anchor=(0,0.98))
```


Plotting Functions

A comprehensive plot

Two ways to collect projections data: **dimensions = KPTS, BANDS, IONS, ORBS**

```
plot_command(elements=[range(2),range(2)],orbs=[0,[1,2,3]],labels=['s','p'])
plot_command(query_data = {'s': (range(2),0),'p': (range(2),[1,2,3])}) # version >= 1.1.1
```

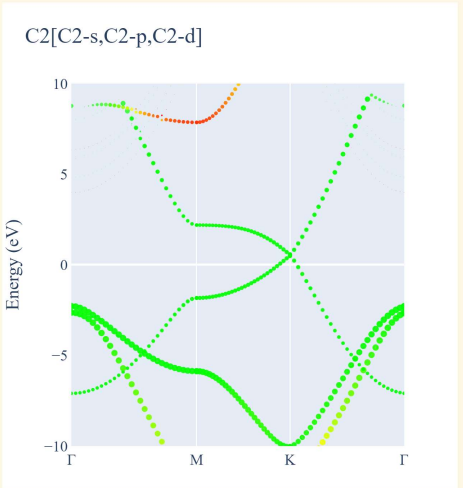


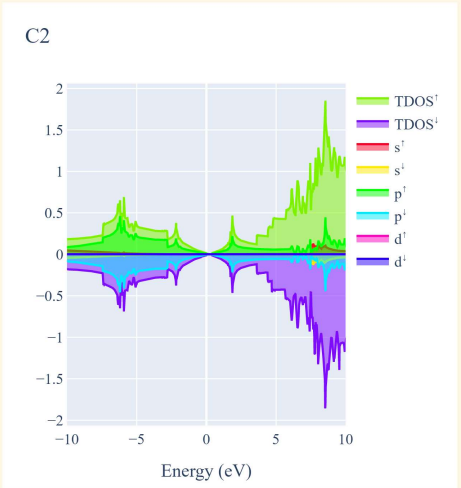

```
ax1, ax2 = pp.get_axes((6,4),ncols=2,widths=[2,1]
ax3 = pp.append_axes(ax2,'bottom',size='100%',pa

query_data = {'s': (range(2),0),'p': (range(2),[1
vr.splot_rgb_lines(ax=ax1,colorbar=False,txt='
ax1.add_colorbar(cmap_or_clist=plt.cm.get_cmap(

kws = dict(elements=[range(2),range(2)],orbs=[0,
vr.splot_color_lines(axes=[ax2,ax3],colormap='R

for ax in (ax1,ax2,ax3):
  ax.set_ylim([-20,15])
  ax.grid(axis='x')

ax1.set_ylabel('E(eV)')
```


Plotting Functions

vr.**iplot_rgb_lines**(elements=[[], [], []], orbs=[[], [], []], labels=['', '', ''], **kwargs)

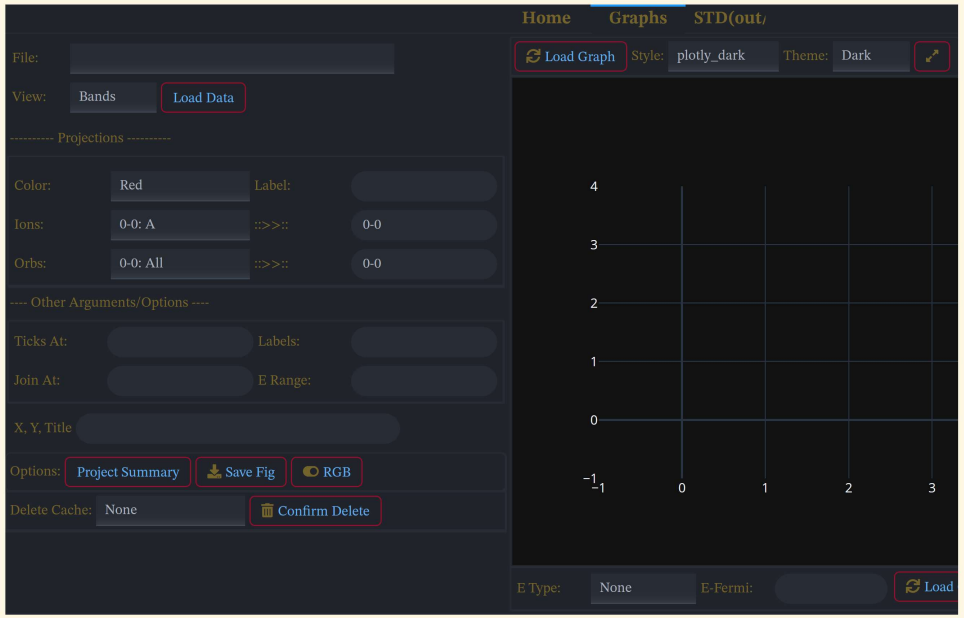
vr.**iplot_dos_lines**(elements=[[0]], orbs=[[0]], labels=['s'], **kwargs)






```
dos_fig = vr1.iplot_dos_lines([[0,1],[0,1],[0,1]],[0,[1,2,3],[4,5,6,7,8]],labels=['s','p','d'],elim=[-10,1
bands_fig = vr.iplot_rgb_lines()
```

Vasprun App



Vasprun App

va = VasprunApp()
After interaction with app, you can run following functions to save plots etc.

va.**splot**(**kwargs) Demo on splot
Returns matplotlib Axes. kwargs
are passed to splot_rgb_lines or splot_dos_lines
based on current figure. kwargs
should exclude whatever inside self.input and path_evr

va.**iplot**(**kwargs)
Returns a detached interactive Figure. kwargs are passed to iplot_rgb_lines or iplot_dos_lines
based on current figure. kwargs should exclude whatever inside self.input and path_evr

Extending Functionality

See https://massgh.github.io/pivotpy/Widgets.html example to build your own ways to interact with this app.

Splitting Large Vasprun Files

This is blazing fast and memeory efficient process

```
Writing 'E:\\Research\\graphene_example\\ISPIN_2\\bands\\_vasprun.xml' ... Done
Writing 'E:\\Research\\graphene_example\\ISPIN_2\\bands\\_set1.txt' ... Done
Writing 'E:\\Research\\graphene_example\\ISPIN_2\\bands\\_set2.txt' ... Done
```

Python

```python
pp.split_vasprun() #auto picks vasprun.xml from pwd
```

Shell

```
> cd E:/Research/graphene_example/ISPIN_2/bands
> ls | grep _
_set1.txt  _set2.txt  _vasprun.xml
```

Other useful functions

pivotpy.**create_colormap**(name='RB', colors=[(0.9, 0, 0), (0, 0, 0.9)])

pivotpy.**docs**()

pivotpy.api.**plt2text**(plt_fig=None, width=144, vscale=0.96, colorful=True, invert=False, crop=False, outfile=None)

pivotpy.api.**plt2html**(plt_fig=None, transparent=True, dash_html=None)

pivotpy.api.**iplot2html**(fig, filename=None, out_string=False, modebar=True)

pivotpy.api.**savefig**(filename, dpi=600, **kwargs)

pivotpy.api.**rotation**(angle_deg, axis_vec)

pivotpy.api.**parse_text**(path, dtype=, delimiter='\\s+', include=None, exclude='#', raw=False, fix_format=True, start=0, nlines=None, count=-1, new_shape=None, cols=None, old_shape=None, slice_rows=None)

pivotpy.api.**interpolate_data**(x, y, n=10, k=3)

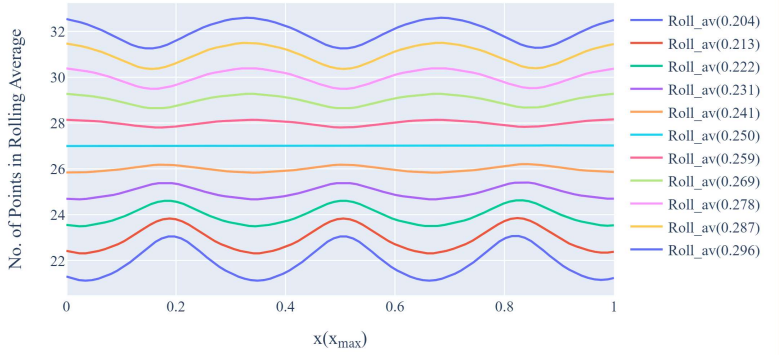
pivotpy.api.**transform_color**(arr, s=1, c=1, b=0, mixing_matrix=None)

pivotpy.api.**set_dir**(path)

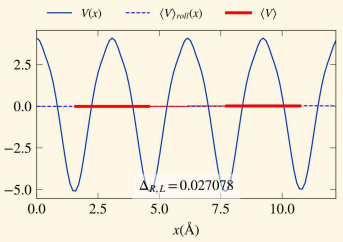
pivotpy.api.**fancy_quiver3d**(X, Y, Z, U, V, W, ax=None, C='r', L=0.7, mutation_scale=10, **kwargs)

LOCPOT/CHG Processing

pivotpy.api.**LOCPOT**(path=None, e=True, m=False) Process LOCPOT and similar files like CHG




```
locpot = pp.LOCPOT()
fig = locpot.view_period(operation='mean_x')
```

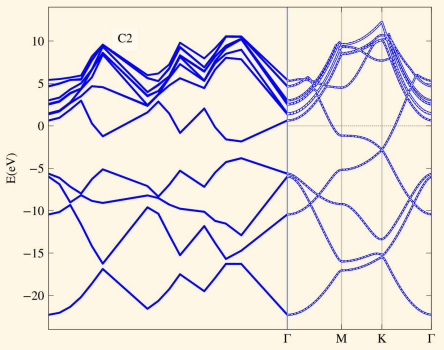

```
Data(
  direction = x
  left = Data(
    v = -0.013540937229764662
    av_range = [14, 40]
  )
  right = Data(
    v = 0.013537416419215777
    av_range = [68, 94]
  )
  deltav = 0.02707835364898044
)
```

```
ax, data = locpot.splot_e(operation='mean_x',period = 0.25,
        labels=('$V(x)$', '$\\langle V \\rangle _{roll}(x)$', '$\\langle V \\rangle $'))
ax.add_legend()
```


Parse Large Text Files

pivotpy.api.**parse_text**(path, dtype=, delimiter='\\s+', include=None, exclude='#', raw=False, fix_format=True, start=0, nlines=None, count=-1, new_shape=None, cols=None, old_shape=None, slice_rows=None)

LOCPOT is parsed using this technique
Let's parse EIGENVAL on the fly




```
  2   2   1   1
0.5274662E+02  0.2468031E-09  0.2468030E-09  0.1999829E-
1.0000000000000000E-004
CAR
C2
   8   104    21


0.0000000E+00  0.0000000E+00  0.0000000E+00  0.1000000
  1    -22.277256   1.000000
  2    -22.277085   1.000000
  3    -10.447433   1.000000
  4    -10.446945   1.000000
  5     -5.983367   1.000000
```

```
# Read KPOINTS
kpoints = pp.parse_text('EIGENVAL',include='E',start=2,new_shape=(-1,3),cols=[0,1,2])
kpoints = np.concatenate([kpoints[:1,:],kpoints]) # Add first point for next step
kpath = np.linalg.norm(kpoints[1:,:] - kpoints[:-1,:],axis=1)
kpath = np.cumsum(kpath)
kpath = kpath  - kpath[-90] # For path match with vasprun.xml excluded kpoints

# Read Energies
NBANDS = int(pp.parse_text('EIGENVAL',start=5,nlines=1,raw=True).split()[-1])
eigen = pp.parse_text('EIGENVAL',exclude='E',start=5,cols=[1]).reshape((-1,NBANDS))

# check if reading is correct by plotting it
ax = pp.get_axes((5,4))
ax.plot(kpath, eigen, color='b')
ax.axvline(x=kpath[-90],lw=0.7)

# Compare with vasprun.xml plot as zero fermi energy
vr.splot_bands(ax,E_Fermi=0,lw=0.5,color='w',ls='dashed')
ax.set(xlim=[kpath[0],kpath[-2]],ylabel='E(eV)') # 1 point less in line collections
ax.grid(axis='x')
file_content = pp.parse_text('EIGENVAL',nlines=30,raw=True) # Raw content
```


Thank You