Example: band structure calculation

This file is intended to detail the calculation presented in Sections 4.3.1 of our *BinPo* manuscript. Importantly, we remind you that any *BinPo* component looks for the input parameters at command-line at first, whereas the omitted and more advanced parameters are set by default from the configuration file. Keeping this in mind, we will show how the configuration files are set for each case and what to type from command-line. For a description of all the parameters in the configuration files, see *~/config_files/help_config.md*. Parameters modifiable through the command-line can be checked by means of help command as:

$ python BP-component.py -h

where *BP-component.py* is any *BinPo* component.

Total band structure calculation

First of all, we show the *bands.yaml* file. In order to make it more readable, we show at first the main block along with the total band structure calculation block.

```
BAND_STRUCTURE :
        identifier : "sto_01"
        path : "XGX"
        number_of_kpoints : 1200
        reference_kpoint : "G"
        Total_Hk_method : 'vectorized'
        num_bands : 50
        bands_task : 1
        initial_plane : 0
        final_plane : 5

        TOTAL_BANDS :
                PLOT_ADJUST :
                        plotstyle : "default"
                        xy_limits : &limxy [-0.46, 0.46, -0.49, 0.08]  #[x_min, x_max, y_min, y_max]
                        linecolor : "navy"
                        linewidth : 1.5
                        fig_size : [6,6]
                        axis_adjust : [0.2, 0.15, 0.9, 0.9] #[left, bottom, right, top]
                        title : ""
                        title_size : 18
                LABELS :
                        xlabel : 'K$_{//}$  [$\AA^{-1}$]'
                        xfontsize : 18
                        ylabel : 'E-E$_{F}$ [eV]'
                        yfontsize : 18
                        ticksize : 12
                SAVING :
                        save_bands : no
                        save_plot : yes
                        format : '.png'
                        dpi : 300
```

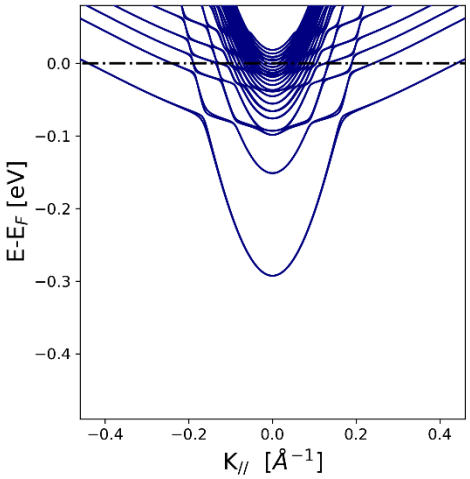
Following the first example, we type:

$ python BP-bands.py -id run2 -ph XGX -kp 600 -tk 0 -nb 50

The first parameter, *id*, is the identifier for the calculation that is being recalled. Then we have the path in reciprocal space (*ph*), the total number of points along the path (*kp*), the task to perform (*tk*) and the number of bands to compute (*nb*). In this case, details after running the component look like:

```
DETAILS:
        Identifier: run2
        Surface: STO(100)
        Number of planes: 40
        Path: XGX
        K-points: 600
        Number of bands: 50
        Temperature: 10 K
        Fermi level: 11.47650 eV
        Task: total_bandstructure
```

Note that, as we mentioned at the beginning of this text, the input parameters are those introduced by command-line, whereas the omitted ones are those set in the configuration file. It is worth mentioning that the data belonging to the SC calculation is automatically loaded from the *run2.yaml* file within the *run2* folder. As we can see, setting the task to 0 printed it as "total_bandstructure", this is just one of the possible modes in which the band structure could be computed. At the end of this calculation, the total band structure along the selected path will be interactively shown.

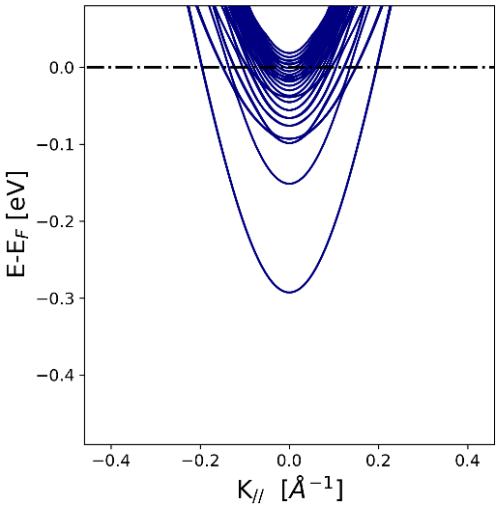


If the parameters "save_bands" and/or "save_plot" are setting to "yes", so the output *.dat* file with the bands and/or the plot will be saved to the *run2* folder.

We will now compute the band structure along another path:

$ python BP-bands.py -id run2 -ph MGM -kp 600 -tk 0 -nb 50

We will get the interactive plot showing the new computed bands.



Band structure projection onto the manifold orbitals

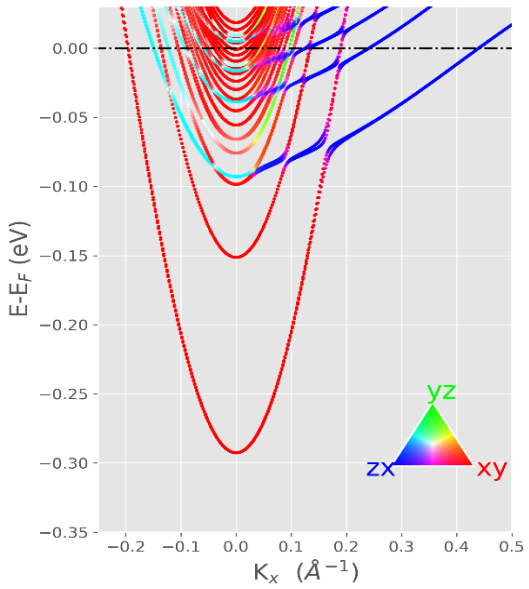
This is how the part of the *bands.yaml* file for this projection looks like:

```
ORBITAL_CHARACTER :
        PLOT_ADJUST :
                plotstyle : "ggplot"
                xy_limits : *limxy
                color_seq :  r,lime,b
                point_size : 4
                fig_size : [6,8]
                axis_adjust : [0.2, 0.15, 0.9, 0.9] #[left, bottom, right, top]
                title : ""
                title_size : 18
        COLOR_TRIANGLE:
                proportion : '35%'
                location : 4
                padding : 0.5
                fontsize : 20
        LABELS :
                xlabel : 'K$_{x}$  ($\AA^{-1}$)'
                xfontsize : 18
                ylabel : 'E-E$_{F}$ (eV)'
                yfontsize : 18
                ticksize : 12
        SAVING :
                save_bands : no
                save_plot : yes
                format : '.png'
                dpi : 300
```

We are going now to compute the bands with projections onto the different orbitals in the t_{2g} manifold by typing:

$ python BP-bands.py -id run2 -ph MGX -kp 1000 -tk 1 -nb 50 -xy -0.25 0.5 -0.35 0.03

In this case, you are selecting the path MGX with a denser number of points. The task parameter equal to 1 indicates that you are projecting the bands onto the different orbitals.



Band structure projection onto a set of planes

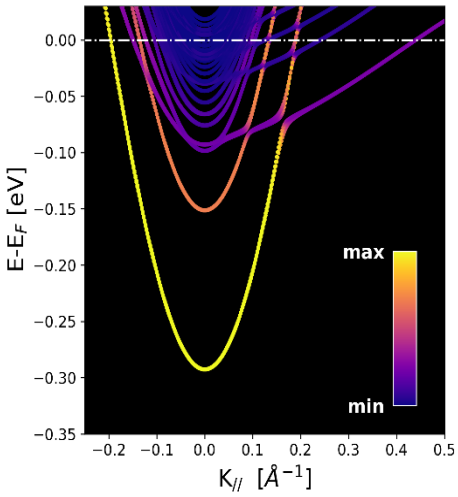
This is how the part of the *bands.yaml* file for this projection looks like:

```
PLANE_PROJECTION :
    PLOT_ADJUST :
        plotstyle : "default"
        xy_limits : *limxy
        colormap : "plasma"
        background_color : "k"
        point_size : 5
        fig_size : [6,6]
        axis_adjust : [0.2, 0.15, 0.9, 0.9] #[left, bottom, right, top]
        title : ""
        title_size : 18
    COLORBAR :
        location : [0.8, 0.2, 0.045, 0.27] #[x, y, width, height]
        textbar : ['min','max']
        fontsize : 15
        fontcolor : 'w'
    LABELS :
        xlabel : 'K$_{//}$  [$\AA^{-1}$]'
        xfontsize : 18
        ylabel : 'E-E$_{F}$ [eV]'
        yfontsize : 18
        ticksize : 12
    SAVING :
        save_bands : no
        save_plot : yes
        format : '.png'
        dpi : 300
...
```

In this last example, we will project the band structure onto a set of planes. Consider the following line:

$ python BP-bands.py -id run2 -ph MGX -kp 1000 -tk 2 -nb 50 -xy -0.25 0.5 -0.35 0.03 -pi 0 -pf 2

As we can note, the line is almost identical to the last example except for the task parameter, which is equal to 2 for the projection onto planes, and the two additional parameters: initial plane (*pi*) and final plane (*pf*). These new parameters allow for projecting the band structure onto the set given by [*pi*, *pf*-1] planes. In consequence, the projection will be done onto the two first planes. The output plot is:


0.00
−0.05
−0.10
−0.15
−0.20
−0.25
−0.30
−0.35
E-E$_F$ [eV]
−0.2
−0.1
0.0
0.1
0.2
0.3
0.4
0.5
K$_{//}$ [Å^{-1}]
max
min