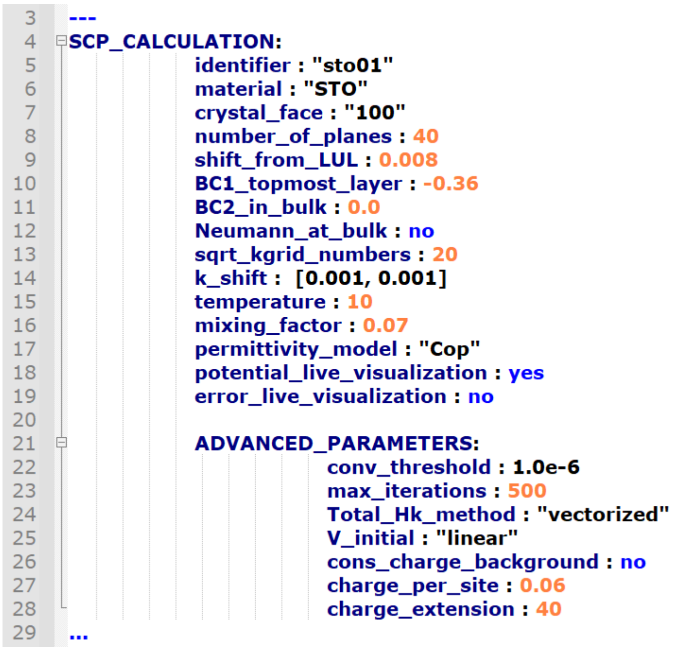
# Example: convergence of self-consistent solutions with the k-grid sampling

This file is intended to detail the calculation presented in the Supplementary Information of our *BinPo* manuscript. Importantly, we remind you that any *BinPo* component looks for the input parameters at the 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*. Those parameters updatable by command-line in any *BinPo* components (*BP-component.py*) can be checked by means of help command as:

```
$ python BP-component.py -h
```

Firstly, we show how the *scp.yaml* configuration file looks like:



```
---
SCP_CALCULATION:
                identifier : "sto01"
                material : "STO"
                crystal_face : "100"
                number_of_planes : 40
                shift_from_LUL : 0.008
                BC1_topmost_layer : -0.36
                BC2_in_bulk : 0.0
                Neumann_at_bulk : no
                sqrt_kgrid_numbers : 20
                k_shift :  [0.001, 0.001]
                temperature : 10
                mixing_factor : 0.07
                permittivity_model : "Cop"
                potential_live_visualization : yes
                error_live_visualization : no

                ADVANCED_PARAMETERS:
                                conv_threshold : 1.0e-6
                                max_iterations : 500
                                Total_Hk_method : "vectorized"
                                V_initial : "linear"
                                cons_charge_background : no
                                charge_per_site : 0.06
                                charge_extension : 40
...
```

Because we want to compare the different SC-solutions according to how dense is the k-grid, we need to run the *BP-scp.py* component just changing the identifier (*id*) and the square root of the total k-points (*nk*). In consequence, the following lines should be typed (does not matter to preserve the order).

```
$ python BP-scp.py -id stoNk16 -nk 16

$ python BP-scp.py -id stoNk26 -nk 26

$ python BP-scp.py -id stoNk36 -nk 36

$ python BP-scp.py -id stoNk46 -nk 46
$ python BP-scp.py -id stoNk56 -nk 56
$ python BP-scp.py -id stoNk76 -nk 76
$ python BP-scp.py -id stoNk106 -nk 106
```

After running these seven times the *BP-scp.py* component, we will have the same number of folders, named with the respective identifiers, where the SC-solutions are hold. Then, you can compare and analyze the SC-solutions according to some criteria by processing and plotting different quantities altogether.

NOTE: It is important to comment that the greater the *nk* parameter the more expensive is the calculation, in terms of computational cost. This is because the size of the total Hamiltonian scales as $nk^2$. As a consequence, at some point you will need to change the parameter “Total_Hk_method” in the ADVANCED_PARAMETERS block of *scp.yaml* file, from “vectorized” to “iterable”. This way, the memory consumption is drastically reduced at the cost of lowering the calculation speed. The value of *nk* at which you will need to make this modification depends completely on the available random memory of your system.