

Table of Contents

Table of Contents	1
Compute quantities from converged simulations	2



[Docs](#) » [Tutorials](#) » [Miscellaneous](#) » Compute quantities from converged simulations

Compute quantities from converged simulations

Version: 2016.0

Downloads & Links

[PDF version](#)
[Basic QuantumATK Tutorial](#)
[ATK Reference Manual](#)

In this tutorial, you will learn how to start a converged calculation to add additional functions. In this way a lot of simulation time can be saved.

What can you do if you forgot to add the Bandstructure analysis to your script when you set it up? Or you discover you probably need more k-points to get a converged transmission spectrum? Do not worry - with **ATK** you don't have to compute all quantities together with the self-consistent calculation. You can always compute them later on, by reading the converged state of the calculation from the HDF5 file.

Using the **Script Generator** is usually the easiest way to do post-processing analysis, especially for small calculations. The post-processing can take some time, however, especially to compute the transmission spectrum or DOS with many k-points. In those cases you might want to run the script in parallel on a cluster (the analysis calculations usually parallelize linearly up to very many nodes). If so, it is still very handy to set up the script using the **Script Generator**, and then save it. That is - just like you would do it for the main self-consistent calculation itself.

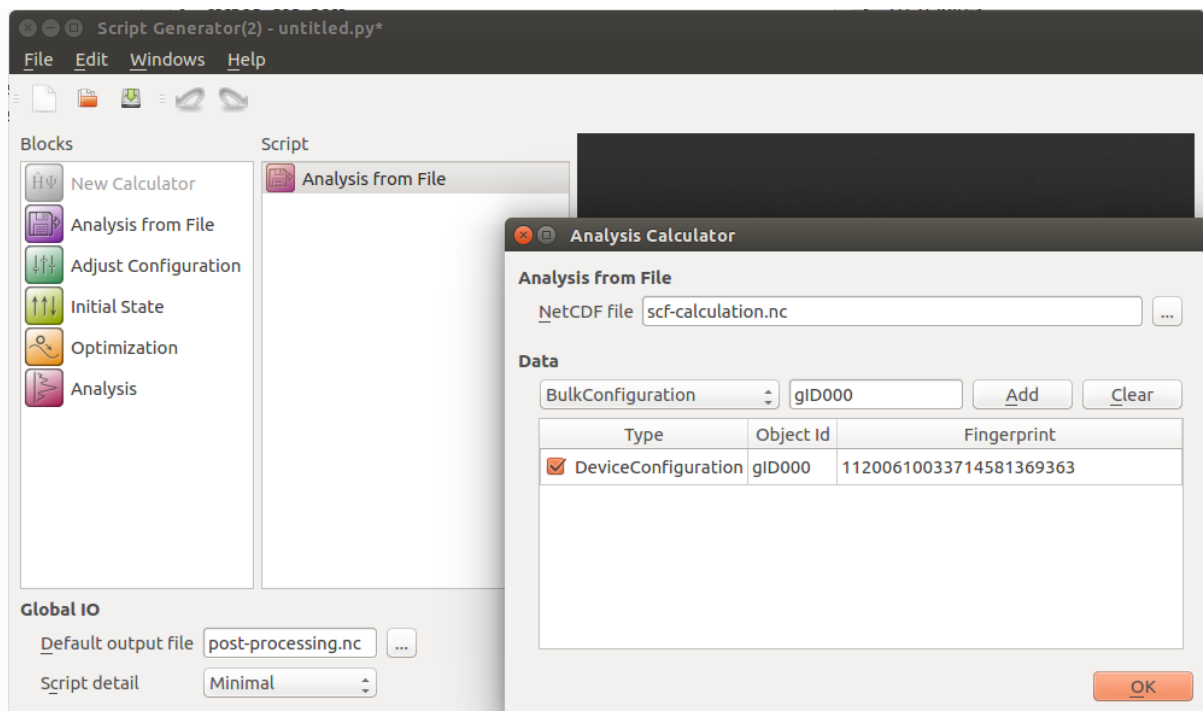


Start a new **Script Generator** from the **QuantumATK** toolbar. When you open it without dropping any configuration on it, the only available block is Analysis from File, which is precisely what we want - to perform analysis based on a HDF5 file. Double-click this block to insert it into the script.

Open it, then browse and select the relevant HDF5 file.

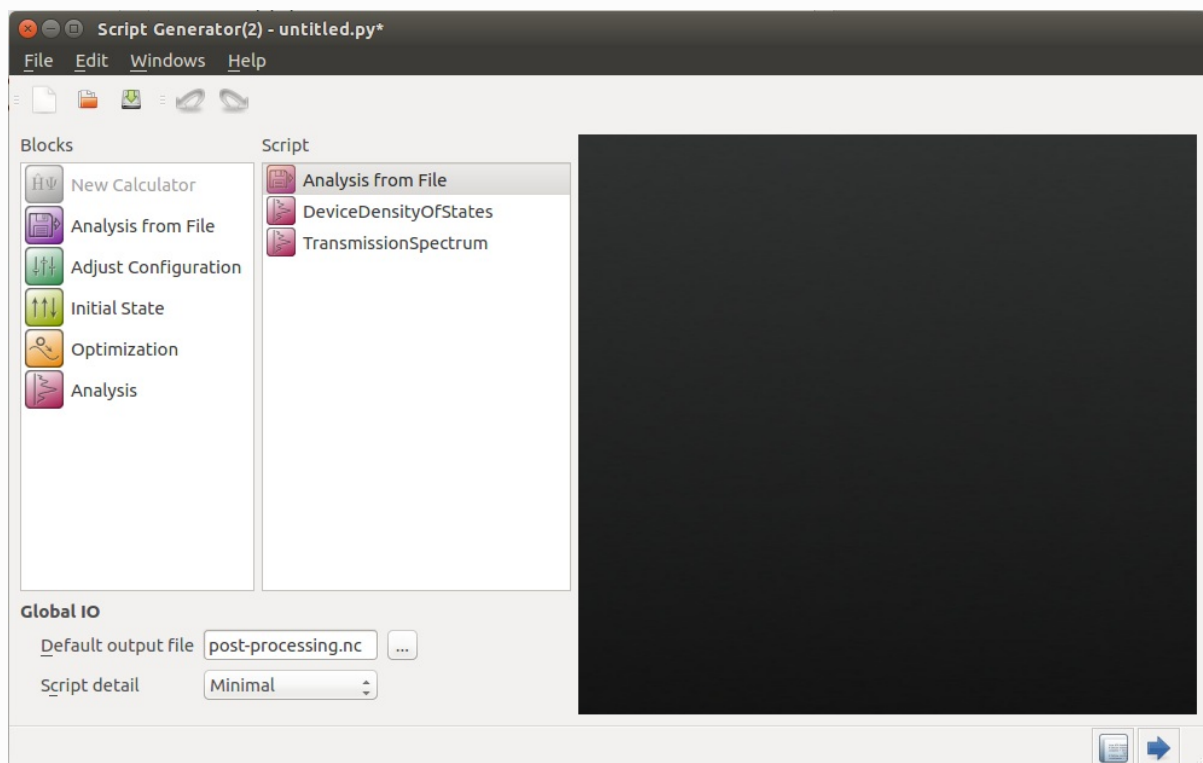
Note

If you intend to run the script on another computer, from the command line (e.g. in parallel) you must naturally copy both the script and the HDF5 file to it. Make sure to place the two files in the same directory, and only specify the name of the input HDF5 file, without any path.



Next, check that the Object id corresponds to the self-consistent calculation you want to base the analysis on. If your calculation was a “standard run”, then most likely the default *glD000* is the one you want. However, if you did a geometry optimization, *glD000* corresponds to the initial geometry, and the optimized geometry will be *glD001*. You can inspect the contents of the HDF5 file, and the object ids, in the *LabFloor* in the main **QuantumATK** window.

Click **OK**, and then add all the *Analysis* quantities you want to compute, and set the parameters as appropriate.





Finally, give a name and location at the blank *Default output file*.


Note

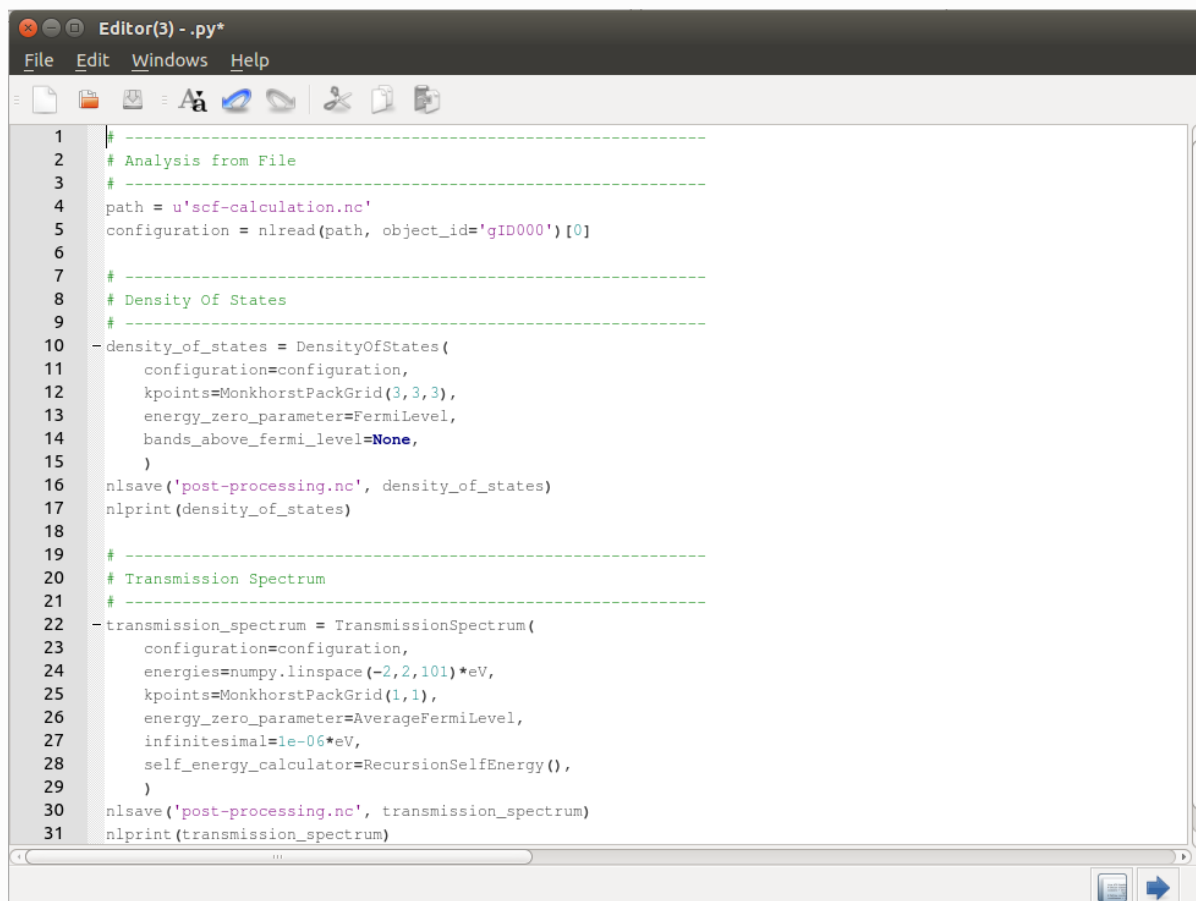
As above, if you will run the script from the command line on a different computer, just set a name without a path, in order to save the file in the running directory.

Tip

You might want to save the computed analysis quantities in a separate HDF5 file, in order not to make the original HDF5 file too big.

Now either send the script to the **Job Manager**  by using the *Send To* button  at the lower right corner, as usual, or save the script and copy it (with the HDF5 file!) to the cluster and run it from the command line.

If you haven't looked into the Python scripts used by **ATK**, this is an excellent type of script to start with. If you send the script to the **Editor** , you will see the following.



```
1  # -----
2  # Analysis from File
3  # -----
4  path = u'scf-calculation.nc'
5  configuration = nload(path, object_id='gID000')[0]
6
7  # -----
8  # Density Of States
9  # -----
10 - density_of_states = DensityOfStates(
11     configuration=configuration,
12     kpoints=MonkhorstPackGrid(3,3,3),
13     energy_zero_parameter=FermiLevel,
14     bands_above_fermi_level=None,
15 )
16 nlsave('post-processing.nc', density_of_states)
17 nprint(density_of_states)
18
19 # -----
20 # Transmission Spectrum
21 # -----
22 - transmission_spectrum = TransmissionSpectrum(
23     configuration=configuration,
24     energies=numpy.linspace(-2,2,101)*eV,
25     kpoints=MonkhorstPackGrid(1,1),
26     energy_zero_parameter=AverageFermiLevel,
27     infinitesimal=1e-06*eV,
28     self_energy_calculator=RecursionSelfEnergy(),
29 )
30 nlsave('post-processing.nc', transmission_spectrum)
31 nprint(transmission_spectrum)
```

You will see that each function of the script is separated into blocks and under each block, there are lines to setup the detail parameters of the simulation. The line that does the main trick is at the very top, using the function *nload()* to read the configuration from the HDF5 file.

Note

nload() can in principle read many objects at once from a HDF5 file. Object ids must however be unique, so by using the *object_id* keyword, you are guaranteed to get only one object back. However, *nload()* always returns a list of objects, even if there is only one match, so we need the *[0]* to pick out the first/only configuration matching the *object_id*.

 Previous

Next 

