CASTEP convergence automation tool 1.0 User Guide

Simone Sturniolo

August 19, 2014

1 Introduction

The CASTEP convergence automation tool (CASTEPconv) is a Python script designed to automate the process of calculating the convergence of energy and forces in DFT calculations with CASTEP, as a function of the cutoff energy and the number of kpoints used. It works with Python 2.6 or higher. Given a .cell file, the script is able to create a set of folders containing input files for the required simulations, run the jobs, and process the output to produce tabulated ASCII data files and graphs. A .param file can be introduced as well to define any additional options for the DFT calculation, while a .conv file can be used to provide further options for the automated convergence calculation itself.

2 Installation

To install the script on Linux or CygWin just open a terminal, navigate to the folder containing the files, and type:

```
user@machine:~$ sudo python setup.py install
```

Enter your password as prompted and the script will be installed and available as a command.

3 Usage

CASTEPconv requires only a .cell file to be present in the folder when it's used. Considering a file named "<*seedname*>.cell", where <*seedname*>represents the name of the job, the syntax to run the convergence job is simply

```
user@machine:~$ castepconv.py <seedname>
```

while in the folder where the file is kept. If one wants the convergence DFT simulations to have additional parameters (e.g. redefine the convergence criteria for SCF iterations, add dispersion correction etc.), a <*seedname*>.param file can be added in the same folder with these options. To control the convergence job itself, instead, a new file, <*seedname*>.conv, is needed.

4 Convergence parameters

The syntax of the .conv file is similar to the one of the CASTEP .param file:

```
<parameter name 1>:    <parameter value 1>
<parameter name 2>:    <parameter value 2>
...
```

The accepted parameter names are listed in the subsections below.

4.1 String parameters

convergence_task: describes the task that is required from the convergence script. This can be INPUT (creation of input files and folders), INPUTRUN (same as INPUT, plus actually runs the jobs), OUTPUT (processes some already finished calculations and creates output files) and ALL (does all of the previous things, waiting for the jobs to finish before processing the output). Default is INPUT.
Legal values: INPUT, INPUTRUN, OUTPUT, ALL

running_mode: describes the mode in which the calculations should be ran. If PARALLEL, all calculations will be launched at the same time (ideal for job submission on a cluster). If SERIAL, the program will wait for one calculation to finish before the next one begins (and reuse the .check file from the previous calculation as a starting point). In the latter case, all files will be created in a single folder. Default is SERIAL.
Legal values: PARALLEL, SERIAL

output_type: plotting program for which an output script should be created. Both GNUPLOT and GRACE are supported. Default is GNUPLOT.
Legal values: GNUPLOT, GRACE

running_command: command line that should be used to run jobs. This should be expressed by replacing the name of the job with the generic token *<seedname>*. Default is *castep <seedname>-dryrun*.
Legal values: any string. If it contains the token *<seedname>*, it will be appropriately replaced.

submission_script: only in version 0.9.2 and higher. The filename of a script that needs to be copied inside all folders before execution. This has been included to accomodate the issues with some large supercomputers that might require job submission to take place by means of such a script rather than a single command. In this case, the script will be renamed as the job, without any extensions, and any instances of *<seedname>*inside it will be replaced by the job name as well. So for example by adding such a script (with the proper job configuration details) it would be possible to submit one's job on a system

that supports *qsub* by using `qsub < <seedname>` as a *running_command*.
Legal values: any file name without spaces in it.

4.2 Float parameters

cutoff_min: Minimum value for the cutoff range explored in eV. Default is 400.0 eV.
Legal values: Any positive float.

cutoff_max: Maximum value for the cutoff range explored in eV. Default is 800.0 eV.
Legal values: Any positive float greater than cutoff_min.

cutoff_step: Step between the values of the cutoff range explored in eV. Default is 100.0 eV.
Legal values: Any positive float.

displace_atoms: Displacement in Angstroms to introduce in atom positions - necessary when the cell is equilibrated and it is not possible to converge forces because they are zero. Default is 0.0 Ang.
Legal values: Any float

final_energy_delta: Tolerance on final energy for the estimate of convergence. Default is 0.00001 eV/atom.
Legal values: Any positive float

forces_delta: Tolerance on maximum force for the estimate of convergence. See above. Default is 0.05 eV/Ang.
Legal values: Any positive float

stresses_delta: Tolerance on maximum stress for the estimate of convergence. See above. Default is 0.1 GPa.
Legal values: Any positive float

4.3 Integer parameters

kpoint_n_min: Minimum value for the k-point range explored. This applies to the shortest side of the kpoint_mp_grid: depending on the size of the other cell parameters, there might be proportionally more k-points along other sides. Default is 1.
Legal values: Any positive integer.

kpoint_n_max: Maximum value for the k-point range explored. Default is 4.

Legal values: Any positive integer greater than kpoint_n_min.

kpoint_n_step: Step between the values of the k-point range explored. Default is 1.
Legal values: Any positive integer.

max_parallel_jobs: Maximum number of parallel jobs to run when in "parallel" mode. Ignored in "serial" mode. Zero means that there is no limit. Default is 0.
Legal values: Any non negative integer (negative values will be ignored).

4.4 Boolean parameters

converge_stress: Apply calculation of stresses to the simulations and then estimate convergence on stresses as well as energy and forces. Default is False.
Legal values: Anything. The word "true", regardless of the case, means the stresses are calculated. Anything else will be interpreted as False.

reuse_calcs: If results from a previous convergence are present, recycle them when possible. This requires the .conv_tab file from the previous calculation to be unaltered. Default is False.
Legal values: Anything. The word "true", regardless of the case, means the stresses are calculated. Anything else will be interpreted as False.

serial_reuse: When running a serial calculation, reuse the .check file from previous calculations to start your new simulations instead of starting from scratch to speed up SCF convergence. Default is True.
Legal values: Anything. The word "true", regardless of the case, means the serial reuse is employed. Anything else will be interpreted as False.

5 Output

When the calculations are over, CASTEPconv will have produced the following files:

<*seedname*>.conv_tab: this file is created during the INPUT phase of the run (when folders and input files for the calculations are created) and sums up the values used for cutoff and the kpoint grids employed in the various files. This is just useful as a memo, and mostly used to resume analysis of previously ran calculations (by using the OUTPUT convergence_task) - if it's not present, the ranges will be recalculated from the .conv file.

<*seedname*>_cut_conv.dat: generated in the OUTPUT phase of the run, will contain a tabulated ASCII of cutoff values (in eV), final energies (in eV) and maximum forces (in eV/Ang) for the various calculations ran. The maximum

force is the highest modulus $|\mathbf{f}_i|$ between the forces acting on all atoms i in the system. If **converge_stress** has been set to True, a fourth column will contain the value of the maximum stress component acting on the system.

where σ_{ij} is the generic element of the stress tensor.

<*seedname*>_kpn_conv.dat: generated in the OUTPUT phase of the run, it is similar to the above, except that it expresses its quantities as a function of the total number of kpoints in the grid.

<*seedname*>_conv.<variable extension>: this file is the script meant for generation of graphic output. By default it will be a Gnuplot script (extension .gp). When other forms of output will be supported, choosing the appropriate value for the output_type option will replace it with a different format.

Besides this, CASTEPconv produces a bit of textual output to suggest which minimum values of cutoff and k-point grid might be the best. This is done by taking the difference between final energy, maximum force, and if required maximum stress, for successive values and comparing it with a tolerance which, by default, is equal to the tolerance assumed in CASTEP calculations on the same quantities. It should be kept to mind that this is only a rule of thumb method, and that it needs to be considered only as an indication - by no means this is meant to be always the correct answer. Different simulations will also require different convergence criteria. Whenever no convergence is found, or the values examined are too small to be sure of their reliability, a warning message is issued.

6 Getting Started

In this section we'll go through a quick tutorial to getting started with CASTEPconv, by using the silicon example provided with the code. Even when the example includes running calculations, they should be fairly simple even for a desktop computer. If you are using an especially old or slow machine, please skip those parts.
In order to run the example, just enter the console, move to the *castepconv/example* folder and edit the *Si.conv* file with a text editor. Replace the line

```
running_command: castep.serial <seedname>
```

with whatever the name/path of castep on your system is (remember to keep the <*seedname*>part untouched though). After doing that, you can just use the command

```
user@machine:~$ castepconv.py Si
```

to run the convergence. Just sit back and wait. At the end of the process, after a few minutes, some output files should have been produced. The ones of interest are *Si_cut_conv.dat, Si_cut_conv.gp, Si_kpn_conv.dat* and *Si_kpn_conv.gp*. The two .dat files contain tabulated results for energy and forces by varying, respectively, cutoff and k-point grid; the two .gp files are scripts for Gnuplot plotting of the same data sets. If you have gnuplot installed they can be visualized for example with:

```
user@machine:~$ gnuplot Si_cut_conv.gp
```

If you plot the data, you will notice that the line for the "force" is completely flat. This happens because the *Si.cell* file used describes a perfectly equilibrated structure, and in a system in equilibrium, forces are always zero. In order to change this we need to upset this equilibrium a bit. To do so go back to editing the *Si.conv* file and uncomment (remove the hash, #, from the beginning) the line

```
displace_atoms: 0.05
```

which will shift all atoms by 0.05Å in a random direction, enough to give rise to non zero forces. Now rerun the calculation, and when prompted, answer to overwrite the old files. At the end you will be able to see the convergence of forces as well.
If instead you'd like to run everything by hand and let CASTEPconv handle only the input generation and the output postprocessing, that is possible too. First let's clean up the current folder structure by issuing the command

```
user@machine:~$ castepconv.py -t c Si
```

which replaces the current task (*all*) with *clean*, namely, delete all generated files and folders. Then edit the *Si.conv* file and uncomment the following lines:

```
running_mode: parallel
```

and either replace *convergence_task* with *input* and run normally, or run with the option *-t i*. In this way, a structure of folders will be generated, containing the starting files for the various points of the convergence calculations. This could be done with the *serial* running mode as well, but in that case everything would be placed in a single folder and you would need to run the tasks in a specific order for the calculations to succeed (as every calculation in that case reuses the results from the previous one as a starting point in order to improve its performance). After generating the folders, visit each one of them and run the CASTEP calculation normally. When all calculations are finished, go back in the parent folder and either change *convergence_task* with *output* and run normally or run with the option *-t o* in order to get the output postprocessed - as an outcome, you will get the same data files and plots as in the first run.