# ONETEP Spring School 2010: Tutorial 1

# Setting up Simple ONETEP Calculations

Nicholas D.M. Hine

Imperial College London

## Input files

Setting up a ONETEP job involves creating a main input file with the suffix `.dat` which contains all the required information to describe both the system and the parameters of the job. The specification uses the "Electronic Structure Data Format" which some users may be familiar from the CASTEP code. This requires the user to provide input in the form of *keywords* and *blocks*. Keywords are written in the form

```
keyword:  value [unit].
```

For example, to specify that the task we wish the code to perform is a Single-Point energy calculation, we would add:

```
task :  SinglePoint
```

to our input file (note that capitalisation is irrelevant). If we wish to specify a cutoff energy of 500 eV for our standard grid, we would add:

```
cutoff_energy :  500 eV
```

The value in eV's will be converted internally to atomic units ($E_\mathrm{h}$ in this case).

If a keyword is not specified in the input file, it is given a default value which is intended to work across a broad range of systems.

A full list of keywords and blocks, giving their meaning, syntax and default values, can be found on the ONETEP wiki:

http://www2.tcm.phy.cam.ac.uk/onetep/Main/Documentation

Blocks are used to define the values of input parameters which need to contain multiple records, such as the definition of the unit cell. They take the form:

```
%block blockname
a1 a2 a3
b1 b2 b3
...
%endblock blockname
```

Most blocks tend not to have a meaningful default value, and must be specified if the related functionality is to be used.

Comments can be added to input files using the '#' or '!' characters. Anything after these characters on a given line will be ignored.

### Setting up the Input File

We will start by running a simple job on a silane molecule $SiH_4$. Log in to DARWIN Create a working directory in which to run ONETEP on the /scratch/ drive

```
> cd ~/scratch
> mkdir silane
> cd silane
```

Create a new input file called silane.dat in your favourite text editor e.g.

```
> nedit silane.dat &
```

If using nedit, it will say that the file does not exist — click on "New file" to create it.

You might like to put a comment at the top explaining what this input file is for e.g.

```
# Simple ONETEP input file for a silane molecule
```

The first thing is to specify the simulation cell. The simplest choice is a cubic box with sides of about 40 bohr. Enter the 3-component cell vectors, one per line, between the `%block lattice_cart` and `%endblock lattice_cart` keywords.

Second, the atomic species need to be specified, in this case silicon and hydrogen. This information needs to be provided between `%block species` and `%endblock species` keywords. In this

block, we need to specify five pieces of information per species, separated by spaces:

(i) your symbol for the atomic species (this can be the same as the element symbol)

(ii) the element symbol itself

(iii) the atomic number Z

(iv) the number of NGWFs per atom

(v) the NGWF radius.

The number of NGWFs required can usually be judged from the symmetry of the atomic orbitals involved: In this case four for silicon and one for hydrogen will be adequate (can you think why?). For this molecule, 6 bohr should be a reasonable starting point for the NGWF radii.

Each atomic species in our calculation needs a pseudopotential file. The pseudopotential files are specified between `%block species_pot` and `%endblock species_pot` keywords. You can use the `hydrogen.recpot` and `silicon.recpot` files from the ONETEP pseudopotentials directory at `/usr/local/Cluster-Apps/onetep/2.4.13/pseudo/` — Copy them to your working directory now (or make a symbolic link).

Next, we need to specify the atomic positions, between `%block positions_abs` and `%endblock positions_abs` keywords. There is one line per atom. Remember to use your symbol for the atomic species as defined in the species block. The coordinates are assumed to be given in bohr unless specified otherwise. It is currently a requirement in ONETEP that all the atoms should lie within the simulation cell so it is best to start by placing the silicon atom at the centre of the cell. The Si-H bond length is about 2.76 bohr and silane is a tetrahedral molecule. The simplest way to work out the coordinates is to note that tetrahedral bonds can be chosen to lie along unit vectors $(a, b, 0)$, $(-a, b, 0)$, $(0, -b, a)$ and $(0, -b, -a)$ where $a = \sqrt{2/3}$ and $b = \sqrt{1/3}$. For example, the vector for the first Si-H bond is $(2.2535, 1.5935, 0.0000)$ bohr. Add these offsets to the position of your silicon atom to create the $SiH_4$ molecule.

The last essential parameter to specify is the kinetic energy cutoff parameter for the psinc basis set. A reasonable value to start with is 300 eV. Use the `cutoff_energy` keyword and remember to specify the energy unit as well as the value.

### Running the Job

To submit this job to be run on the compute nodes of DARWIN, we will need a job script. The contents of this script will vary greatly between different supercomputers, so we have provided a working example suitable for this machine in the home directories of each of the tutorial accounts. Copy this script to your working directory and modify the appropriate variables as directed in the comments inside the script.

Submit this script to the queue with the 'qsub' command and wait for it to run — it should execute immediately as long as not too many others are trying to do the same thing at once. Examine the output: if you have followed these instructions precisely it should converge very quickly (8 iterations) to a total energy of around -6.1897$E_{\mathrm{h}}$.

### Convergence Convergence Convergence

Just as in any form of traditional DFT, we must ensure that our calculation results are converged with respect to the size of the basis. In ONETEP, convergence with basis size is controlled by a small number of parameters, with respect to which the total energy is variational. In this context, that means the total energy at a given value of the parameter will be an upper bound to the true, converged total energy, and increasing the parameter will monotonically decrease the total energy, which asymptotically tends to its converged value.

#### Cutoff Energy

The first parameter will be familiar to anyone who has carried out plane-wave DFT calculations — the cutoff energy. This specifies the kinetic energy of the maximum **G**-vector of the reciprocal-space grid, and therefore the spacing of the real-space grid. With a 40 bohr cell and a 300eV cutoff, ONETEP will have chosen a $48 \times 48 \times 48$ grid, hence a grid spacing of 0.833 bohr. This may be too coarse: move your old output file to a new name (eg `SiH4.out_Ec300`) and try changing the cutoff energy to 500eV, then re-run the job script. You may wish to add `output_detail: VERBOSE` to your input file, to see exactly what grids are being used at each cutoff.

Comparing the two outputs, you should see that the total energy has decreased by around 0.03$E_{\mathrm{h}}$ (nearly 1eV, or 0.2eV/atom). This suggests 300eV was too low initially. Try increasing the cutoff

in steps of 100eV (You may wish to automate this, by having a loop in your job script in which the input file is updated and the job run for each update, if you are sufficiently familiar with bash scripting).

Plot the total energy as a function of cutoff energy (we assume here that you are familiar with Linux plotting tools such as gnuplot or xmgrace — please speak to a demonstrator if this is not the case). You should see a monotonic decrease in $E_T$ as a function of $E_{cut}$: try to evaluate at what value you think the total energy is converged to about 0.1eV/atom of its asymptotic limit. Note that the calculation time increases rapidly with cutoff energy, because the number of grid points in each FFTbox is growing rapidly with cutoff energy, and thus each FFT takes longer, so do not try going beyond around 1200eV.

In few cases in reality do we require strict convergence of the total energy. It is more usual that we require convergence of some measurable quantity such as a binding energy. In that case, we do not require the total energy to be converged — only the difference between total energies of very similar systems. This may converge much faster than the total energy itself, presuming the same species are present in both systems. Always consider what it is that you need converging before you start running enormous calculations! For more advice on this topic, check out the CASTEP tutorials available at: `http://www.castep.org/`

**NGWF radius**

Next, we will investigate convergence with respect to the NGWF radius. Pick a value of cutoff energy for which you can perform reasonably fast calculations (say, 500eV) and try increasing the NGWF radius from 6.0 to 10.0 in 1.0 bohr steps. Plot the total energy against NGWF radius. Again, you should see a monotonic decrease. Note that above 6.0 bohr the FFT box is as large as the simulation cell — in a larger cell this would keep growing, and the calculation time would increase rapidly. Also, you should notice that the number of NGWF Conjugate Gradients iterations grows with the size of the localisation region — this is natural since with larger spheres there are more NGWF coefficients to simultaneously optimise.

You may also wish to try converging with respect to the number of NGWFs per atom (eg try 9 NGWFs on the Silicon). In some systems, notably crystalline solids, this can be crucial to achieving good convergence of the NGWFs themselves.

### Kernel Cutoff

This $SiH_4$ system is too small to investigate convergence with respect to the cutoff of the density kernel. In larger systems truncation of the density kernel can be a good way to speed up the calculation — indeed, asymptotically it is only by truncating the kernel that true 'linear-scaling' behaviour of the computational effort will be observed.

The kernel cutoff is controlled by the `kernel_cutoff` keyword — this defaults to 1000 bohr (i.e. effectively infinite). Density kernel truncation should be used with a degree of caution: generally speaking, one would want to be able to run a full calculation for a fairly large system first, with an infinite cutoff, to establish a known baseline. Then, try decreasing the kernel cutoff from that point and see what the effect is on the total energy, on the level of NGWF convergence (as measured by the NGWF RMS gradient), and on the computation time. If significant time savings can be achieved without trading in too much accuracy, it may be worthwhile to bring down the cutoff for all similar calculations. Proceed with care, though — calculations with a truncated kernel tend to converge in a less stable manner.

## Crystalline Silicon

If time permits, you may wish to try out a calculation on a periodic solid. As it is fairly well-behaved but illustrates some interesting concepts, let's try crystalline silicon, in the diamond (f.c.c.) structure. We will build a $2 \times 2 \times 2$ version of the 8-atom simple-cubic unit cell.

Copy your $SiH_4$ input to a new file (eg `Si64.dat`) in a new directory (eg `Silicon`) and remove the references to hydrogen from the `species` and `species_pot` blocks. Copy `silicon.recpot` to this directory as well. In the new input file, set the NGWF radius to 7.0 bohr, the number of NGWFs per atom to 4, and the cutoff energy to 600eV. Edit the cell side length so that it is $2\times$ the lattice parameter of crystalline silicon in the LDA (around 10.1667 bohr). For reasons that will become clear if you read the last section of this tutorial, on Common Problems, also set `ngwf_cg_max_step: 8.0` to prevent the CG line step being capped unnecessarily and `maxit_ngwf_cg: 30` to terminate the NGWF CG after 30 iterations (in case it's not converging).

Typing out the positions would be rather time-consuming and error-prone with 64 atoms in the cell, so use your favourite scripting/programming language (bash, awk, python, perl, etc would all

be suitable — even C or FORTRAN) to write a list of the positions. You will need to repeat the basis (atoms at $(0,0,0)$ and $(\frac{1}{4},\frac{1}{4},\frac{1}{4})a$) at each of the positions of the f.c.c. lattice: $(0,0,0)$, $(\frac{1}{2},\frac{1}{2},0)$, $(0,\frac{1}{2},\frac{1}{2})$, and $(\frac{1}{2},0,\frac{1}{2})$. Copy the result into your `positions_abs` block. An example input file for this job can be found on the tutorial web page on the wiki.

Modify your job script to run on 16 cores (4 nodes) and submit it to DARWIN. The calculation should take around 10-15 minutes. Feel free to stop it as soon as you see what is happening, since you will find that the calculation fails to converge: the RMS gradient remains stuck above the threshold for convergence. Likewise, the total energy will not converge to a fixed value.

Make a copy of your output and modify the NGWF radius in the input file to 8.0 bohr and the number of NGWFs per Si atom to 9. This introduces NGWFs with $d$-like symmetry rather than just $s$ and $p$, allowing much more variational freedom. You should now find the calculation converges nicely, but will take rather longer to run (20-30 minutes on 16 cores).

Now try activating `write_forces: T` to calculate the forces on each atom. All the forces should be small: in principle they are constrained by the symmetry of the crystal to be exactly zero. However, you will see that they are not exactly zero because the symmetry of the system is broken by the psinc grid, which is not necessarily commensurate with the unit cell. However, in this small cell, it will not be possible to fix this as the number of points across the FFT box must be odd, and in a small cell the simulation cell and the FFT box coincide, so the number of points across the simulation cell must also be odd.

Adjust your script to write a $5 \times 5 \times 5$ supercell of the crystal (1000 atoms). Reduce the kernel cutoff to 25 bohr with `kernel_cutoff: 25` and set the code to perform 1 NGWF iteration only `maxit_ngwf_cg: 1` (otherwise the calculation would take several hours). To restore the symmetry, adjust the `psinc_spacing` value to be a divisor of the supercell length such that an exact number of grid points spans each unit cell of the crystal (pick a value which gives an effective cutoff energy close to 600eV so as not to increase the run time too much) and and set off the 1000 atom job — as long as not too many other people are trying to use our dedicated nodes for earlier steps. This should not take too long on 32 cores. Even though the electronic structure is

If there is space on the DARWIN nodes we have use of for the tutorial overnight, you may wish to try running this $5 \times 5 \times 5$ supercell (1000 atoms) and an $8 \times 8 \times 8$ supercell (8000 atoms) for 1 iteration overnight on DARWIN to see how the runtime scales. Ask an instructor to look over your

input file before submitting it as this is a lot of core hours to burn! Completed output files will be posted for you to look at if you don't get a chance to try the calculation yourself.

Beyond around 500 atoms, the calculation should be into the so-called 'linear-scaling' regime, so the 8000 atom calculation should only take a little over 8 times the 1000 atom calculation. This is rather better than the nearly 512 times longer it would take with traditional cubic-scaling DFT!

## Diagnosing Common Failures

Performing DFT calculations with ONETEP is, unfortunately, not yet quite as close to being a 'black-box' type approach as is plane-wave DFT with codes such as CASTEP. With badly-chosen input settings, many fairly standard calculations in ONETEP will not converge, or may even converge to the wrong result. Fortunately, many of these problems are easy to fix with a bit of experience. In general, it is advisable to run with full output verbosity (`output_detail: VERBOSE`) the first few times you run a new kind of system, and to be on the lookout for any warnings or garbage numbers in the output (eg ****'s in place of what should be real numbers). Remember that for the energy to be accurate, we must have simultaneous convergence of both the density kernel and the NGWFs. If either of these are not converging well by the end of the calculation, there may be a problem.

In this section, we will briefly examine some reasons behind common types of convergence failure, and what to do to eliminate those failures and perform accurate simulations.

- Problem: Repeated 'safe' steps (of 0.150 or 0.100) during NGWF Conjugate Gradients optimisation, leading to poor or no convergence. This often means that the step length cap for NGWF CG is too short.
  - Solution: increase ngwf_cg_max_step, eg to 8.0.
- Problem: Repeated 'safe' steps (of 0.150 or 0.100) during LNV Conjugate Gradients optimisation, leading to poor or no convergence. This often means that the step length cap for LNV CG is too short.
  - Solution: increase lnv_cg_max_step, eg to 8.0.
- Problem: Occupancies 'break' during LNV optimisation of kernel. Examine the output with `output_detail: VERBOSE` and look at the occupancy error and occupancy bounds during the "Penalty functional idempotency correction" section of each LNV step. Check for occupancies outside the stable range (approx -0.3:1.3) or RMS occupancy errors not decreasing (particularly if no kernel truncation is applied).
  - Solution: Activate LNV line step checking with lnv_check_trial_steps: T. This checks that the kernel is still stable after the proposed line step is taken.

- Problem: Occupancies are 'broken' from start of calculation. Symptoms as above. Palser Manolopoulos or Penalty Functional may be unstable due to degeneracy or near-degeneracy at the Fermi level. Check the output of Palser Manolopoulos for warnings.
  - Solution: If there is an initial degeneracy at the Fermi level, an O($N^3$) diagonalisation may be required to get a good starting kernel. Set maxit_palser_mano : -1.
- Problem: RMS Commutator (HKS-SKH) of kernel and Hamiltonian stagnates (stops going down with each iteration) during LNV optimisation. This is a sign that the current set of NGWFs is not able to represent a density matrix that both reproduces the electron density that generated the Hamiltonian while simultaneously describing the occupied eigenstates of that Hamiltonian. If this problem does not start to go away after a few steps of NGWF optimisation, a better or larger initial set of NGWFs may be required.
  - Solutions: Increase number of NGWFs per atom, increase radius of NGWFs, improve initial guess for NGWFs by using fireballs.
- Problem: RMS NGWF gradient stagnates (stops going down) during NGWF CG optimisation, while energy is still going down slowly. This often suggests that the NGWFs may have expanded away from their centres to have significant value near the edge of their localisation region, and thus cannot optimise successfully.
  - Solution: Increase NGWF radius. Sometimes increasing energy cutoff helps as well.

For smaller systems and initial tests, a useful check on the accuracy of the final result is to perform a full $O(N^3)$ diagonalisation at the end of the calculation, if it is computationally feasible to do so. To activate this, turn on a properties calculation with `do_properties: T`, and then ask for an eigenvalue calculation of the first 100 eigenvalues either side of the Fermi energy, for the kernel and Hamiltonian matrices, by setting `num_eigenvalues: 100`. If all is well, then the occupation eigenvalues should all be close to 0.00000 or 1.00000 (empty or full) and the Hamiltonian eigenvalues should all be within a sensible range.

One final note if you're not getting the result you expect — check the units on your atomic positions! ONETEP expects positions in bohr if the units are not specified, so if your positions are in Angstroms, you will need to add 'ang' as the first line of the `positions_abs` block.