ONETEP Tutorials

Release 0.0.1

ONETEP Collective

Sep 05, 2024

CONTENTS:

CHAPTER

ONE

INTRODUCTION

The current set of tutorials for ONETEP can be found here.

CHAPTER

TWO

TUTORIAL 1: SETTING UP SIMPLE ONETEP CALCULATIONS

Author
Nicholas D.M. Hine and Chris-Kriton Skylaris

Date
July 2024

2.1 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. 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 : 500.0 eV
```

The value in eV's will be converted internally to atomic units (Eh 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 keyowrd database: https://onetep.org/resources/keyword-database/.

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.

2.1.1 Setting up the Input File

We will start by running a simple job on a silane molecule SiH_4. Create a working directory in which to run ONETEP

```
> mkdir silane
> cd silane
```

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

```
> vi silane.dat &
```

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 40.0 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:

1. *Your* symbol for the atomic species (this can be the same as the element symbol).
2. The element symbol itself.
3. The atomic number Z.
4. The number of NGWFs per atom.
5. 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? Answer: one s- and three p-orbitals for each carbon atom and one s-orbtial for each hydrogen atom).

For this molecule, 6.0 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 `pseudo` directory. Copy them to your working directory now (or make a symbolic link by `ln -s /path/to/hydrogen.recpot hydrogen.recpot`).

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. While it is not requirement in ONETEP that all the atoms should lie within the simulation cell, it is best (for visualisation purposes) 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 = 2/3 and b = 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.

2.1.2 Running the Job

Examine the output: if you have followed these instructions it should converge very quickly (8 iterations) to a total energy of around -6.1897 Eh.

2.1.3 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 300 eV cutoff, ONETEP will have chosen a 48 × 48 × 48 grid, hence a grid spacing of 0.833 Bohr. This may be too coarse: move your old output file to a new name (e.g., `SiH4.out_Ec300`) and try changing the cutoff energy to 500 eV, 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 Eh (nearly 1 eV, or 0.2 eV/atom). This suggests 300 eV was too low initially. Try increasing the cutoff in steps of 100 eV (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 suficiently familiar with bash scripting)

Plot the total energy (ET) as a function of cutoff energy. You should see a monotonic decrease in ET as a function of Ecut: try to evaluate at what value you think the total energy is converged to about 0.1 eV/atom of its asymptotic limit. Note that the calculation time increases rapidly with cutoff energy, because the number of grid points in each FFT box is growing rapidly with cutoff energy, and thus each FFT takes longer, so do not try going beyond around 1200 eV.

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, which is based on energy differences. 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!

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, 500.0 eV) 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 coeficients to simultaneously optimise. You may also wish to try converging with respect to the number of NGWFs per atom (e.g., 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 as calculations with a truncated kernel tend to converge in a less stable manner.

2.1.4 Crystalline Silicon

You may wish to try out also 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 (e.g., `Si64.dat`) in a new directory (e.g., `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 600 eV. 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 or even C or FORTRAN) to write a list of the positions. You will need to repeat the basis (atoms at (0, 0, 0) and (1/4,1/4,1/4)a) at each of the positions of the f.c.c. lattice: (0, 0, 0), (1/2,1/2,0), (0,1/2,1/2), and (1/2, 0, 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.

The calculation run for while but 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. This illustrates a common type of convergence failure in solids, whereby the NGWF optimisation is failing to find a well-defined minimum energy. Several solutions are possible: increasing the NGWF radius or count, though both of these are relatively expensive in computational terms. You could 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 would now find the calculation converges nicely, but will take rather longer to run.

Another option is to active adaptive kinetic energy preconditioning by setting `kzero: -1`. This should result in convergence within about 9-10 iterations.

Psinc Grids

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: 25.0` and set the code to perform 1 NGWF iteration only `maxit_ngwf_cg: 1` (otherwise the calculation would take longer to run, you can try this if you have time). 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 600.0 eV so as not to increase the run time too much) and and set off the 1000 atom job. This should not take too long on 32 cores.

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!

2.1.5 Diagnosing Common Failures

With badly-chosen input settings, even 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 (e.g., `****`'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`, e.g., 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`, e.g., 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 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(N3) 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.

- **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 the kinetic 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.

This completes tutorial 1.

Files for this tutorial:

- `SiH4.dat`
- `Si8000.dat`
- `Si64.dat`

- `Si1000.dat`

CHAPTER

THREE

TUTORIAL 2: ASE ONETEP INTERFACE

Author
Tom Demeyere

Date
August 2023, updated June 2024

This tutorial will guide you through the use of the ASE interface to ONETEP. The Atomic Simulation Environment (ASE) is a set of tools and Python modules for setting up, manipulating, running, visualizing and analyzing atomistic simulations. The ASE interface to ONETEP allows you to set up and run ONETEP calculations from Python scripts.

Some tutorials (1, 4, 10) have Jupyter notebooks that you can use to run examples interactively. You can download them below, along with the needed pseudopotentials, input files and ONETEP launching script. Alternatively, you can clone the entire repository, and have them almost ready to run (you will still need to tweak the *onetep_launch.sh* for your machine.)

- `Jupyter notebooks`
- `Jupyter notebooks required files`

To run these tutorials, please have a look inside the *launch_onetep.sh* script you downloaded in order to load correctly all modules and environment variables needed on your machine. Similarly, you might need to change the paths of the *OnetepProfile* object inside the Jupyter Notebooks to match your paths. The OnetepProfile is an ASE object that must be created and passed to the ASE ONETEP calculator in order to specify both ONETEP command and pseudopotential path. More information is available just below.

This tutorial will mainly focus on running ONETEP calculations with ASE, if you are not familiar with ASE as a whole, feel free to consult the mini-tutorial here:

- `learn_ase_in_y_minutes.py`

3.1 ASE Configuration File

To run ONETEP with ASE, you need to have a configuration file that specifies the path to the ONETEP binary and the location of the pseudopotentials. There is no default configuration file, so you need to create one.

ASE looks for a configuration file named `config.ini` in the default location `$HOME/.config/ase/`. The configuration file should follow the pattern (do not put quotes around the values):

```
[onetep]
command = mpirun -np 10 -v /path/to/onetep/binary
pseudo_path = /path/to/pseudos
```

Replace `/path/to/onetep/binary` with the actual path to your ONETEP binary. If you want to use a different location for the configuration file, you can set the `ASE_CONFIG_PATH` environment variable. For example:

```
$ export ASE_CONFIG_PATH="/path/to/custom/config.ini"
```

Alternatively, if you don't want to use the configuration file, you can create a `OnetepProfile` directly in your script. For example:

```
from ase.calculators.onetep import Onetep, OnetepProfile

profile = OnetepProfile(
  command="mpirun -np 10 -v /path/to/onetep/binary",
  pseudo_path="/path/to/pseudos"
)
calc = Onetep(profile=profile)
```

This will override the configuration file and use the `OnetepProfile` object instead.

3.2 Pseudopotentials

If no pseudopotentials are passed ASE will try to guess the files based on the element used and the pseudo_path variable. Otherwise you can pass a dictionary with the element as key and the path to the pseudopotential as value. The path can be either absolute or relative to the `pseudo_path`.

```
# Explicitly providing each path:
calc = Onetep(pseudopotentials = {'H': '/path/to/pseudos/H.usp', 'O
↪': '/path/to/pseudos/O.usp'})
# Using relative paths:
calc = Onetep(pseudopotentials = {'H': 'H.usp', 'O': 'O.usp'})
# ASE will try to guess them if you don't provide them:
calc = Onetep()
```

For ASE to correctly guess the pseudopotentials, it is best to use a `pseudo_path` that contains only one pseudopotential file for each element. If there are multiple files for the same element, ASE will not be able to guess which one to use.

3.3 ONETEP Calculator

Simple calculations can be setup calling the Onetep calculator without any parameters, in this case ONETEP's default parameters will be used. For more complex cases using the `keywords` parameters is necessary. The `keywords` parameters is a dictionary, in which each of the keys is a string that should be a ONETEP keyword, and the corresponding value is what you want to set that keyword to in the input.

```
from ase.calculators.onetep import Onetep

# Default parameters
calc = Onetep()

# Custom parameters
keywords = {
    'xc' : 'PBE',
    'do_properties' : True,
    'cutoff_energy' : 35,
    'output_detail': 'verbose',
    'elec_energy_tol': 1.0e-5,
}

calc = Onetep(keywords=keywords)
```

Alternatively you can read an already existing input file with the function `read_onetep_keywords`

```
from ase.io.onetep import read_onetep_keywords

keywords = read_onetep_keywords('input_file.dat')

# Let's change one specific keyword
keywords['xc'] = 'PBE0'

calc = Onetep(keywords=keywords)
```

3.4 Examples

Here is an example python script which sets up a calculation on a water molecule:

```
from ase.build import molecule
from ase.calculators.onetep import Onetep

water = molecule('H2O', vacuum=10)

calc = Onetep(xc='PBE', paw=True)
water.calc = calc

water.get_potential_energy()
```

Here is a more complex example, this time setting up a Pt_{13} cluster and running a geometry optimisation, note that here as far as ONETEP is concerned we are running singlepoint calculations, the geometry optimisation is done by ASE's BFGS optimiser:

```
import numpy as np

from ase.build import molecule
from ase.calculators.onetep import Onetep
from ase.cluster import Octahedron
from ase.optimize import BFGSLineSearch

# Pt13 from ase.cluster
nano = Octahedron('Pt', 3, 1)
nano.center(vacuum=10)

# ONETEP default are atomic units, one can specify 'cutoff_energy'␣
↪: '600 eV' if needed.
keywords = {
    'xc' : 'rpbe',
    'do_properties' : True,
    'cutoff_energy' : 35,
    'output_detail': 'verbose',
    'elec_energy_tol': 1.0e-5/len(atoms),
    'edft': True,
}

# append = True will not overwrite file at each step
calc = Onetep(
    append = True,
    keywords = keywords)

nanoparticle.calc = calc

opt = BFGSLineSearch(atoms = nano)
opt.run(fmax=0.1)
```

Here is an example of setting up an EELS and LDOS calculation on an N-substituted graphene sheet, demonstrating several more advanced functionalities (tags, species groups, and overrides to pseudopotentials and atomic solver strings)

```
import numpy as np

from ase.build import graphene_nanoribbon
from ase.calculators.onetep import Onetep
from ase.io import write

sheet = graphene_nanoribbon(10, 10, type='zigzag', vacuum = 10)

# Get all distances to center of mass
com = sheet.get_center_of_mass()
distances_to_com = np.linalg.norm(sheet.positions - com, axis = 1)

# Find atoms close to com and change one randomly to N
p, = np.where(distances_to_com < 5)
to_nitro = choice(p)
sheet[to_nitro].symbol = 'N'

shell_rad = np.array([1.5, 2.5, 3.0, 4.0, 4.5])

tags = np.zeros(len(sheet), dtype=np.int32)

# We want to tag atoms that are close to the introduced nitrogen
for idx, rad in enumerate(reversed(shell_rad)):
    # All distances N-C
    dist = norm(sheet[to_nitro].position - sheet.get_positions(),␣
↪axis = 1)
    # Which ones are closest to rad?
    p, = np.where(dist < rad)
    # Cannot be the nitrogen itself
    p = p[p != to_nitro]
    # Tags them
    tags[p] = len(shell_rad) - idx

sheet.set_tags(tags)

tags = ['' if i == 0 else i for i in tags]

species = np.unique(np.char.add(sheet.get_chemical_symbols(), tags))

keywords = {
    'species_core_wf' : ['N /path/to/pseudo/corehole.abinit'],
    'species_solver' : ['N SOLVE conf=1s1 2p4'],
    'pseudo_path': '/Users/tomdm/PseudoPotentials/SSSP_1.2.1',
    'xc' : 'PBE',
    'paw': True,
    'do_properties': True,
```

(continues on next page)

(continued from previous page)

```
    'cutoff_energy' : '500 eV',
    'species_ldos_groups': species,
    'task' : 'GeometryOptimization'
}

calc = Onetep(
    keywords = keywords
)

# Checking the input before running the calculation
write('to_check.dat', sheet, format='onetep-in', keywords =␣
↪keywords)

sheet.calc = calc
# Will actually run the geometry optimisation
# using ONETEP internal BFGS
sheet.get_potential_energy()
```

Quickly restart with solvation effect using the soft sphere solvation model:

```
from ase.io import read
from ase.io.onetep import get_onetep_keywords

# Read from the previous run...
optimized_sheet = read("onetep.out")

# Function to retrieve keywords dict from input file...
keywords = get_onetep_keywords('onetep.dat')

# We add solvation keywords
keywords.update(
    {
    'is_implicit_solvent': True,
    'is_include_apolar': True,
    'is_smeared_ion_rep': True,
    'is_dielectric_model': 'fix_cavity',
    'is_dielectric_function' : 'soft_sphere'
    }
)

optimized_sheet.calc = Onetep(keywords=keywords)
optimized_sheet.get_potential_energy()
```

3.5 Important note

If you are not keen about using ASE to run ONETEP calculations, it is always possible to use ASE to write ONETEP input files and run them manually. This should be done by using the general ASE IO modules `ase.io.write` and `ase.io.read` to write and read ONETEP input files. In every example above, all you need to do is to replace the `get_potential_energy()` call by a `write` call to write the input file, such as `write('input_file.dat', atoms, format='onetep-in', keywords=keywords)`. You can then run the ONETEP binary manually as you always do.

3.6 How to use ASE on HPCs

If the HPC you are using has a module system, you can load the conda module and create an environment with the required packages. If you don't have access to a module system, you can install miniforge in your home directory and create an environment there. A tutorial to do so is available at the end of this document.

3.6.1 How does python launch ONETEP under the hood?

When you run a python script with ASE and ONETEP, ASE will both construst the command to be launched and the input file. The command will be constructed based on the `command` key in the ASE configuration file. Or based on the `command` key in the `OnetepProfile` object if you send the profile manually. The command will be executed with the *subprocess* module using the `check_call` function. The inner working of the `check_call` function is to run the command in a subprocess and wait for it to finish. If the command fails, an exception will be raised. To run the command no new shell is created, and all the environment variables are inherited from the parent process. All stdout and stderr will be redirected to the onetep.out and onetep.err files.

The input file will be created by the IO functions of ASE, namely `ase.io.onetep.write_onetep_input`. This function will write the input file in the format expected by ONETEP. This will be automatically done if a calculation is launched via `atoms.get_potential_energy()` or else.

3.6.2 General case

There are two ways to submit job using ASE on HPC, you can directly sbatch the python script by putting the correct shebang at the top of the script, or you can use an additional bash script to submit the job. The bash script will have to activate the environment and run the python ASE script. Here is an example of such a script:

```
#!/bin/bash
#SBATCH --job-name=ASE_ONETEP
...
```

(continues on next page)

(continued from previous page)

```
conda activate myenv

module load ... # Load all the modules needed by ONETEP
export ... # Set all the environment variables needed by ONETEP

export ASE_CONFIG_PATH="/path/to/scratch/.ase_config.ini"

python my_ase_script.py
```

```
# Your python script can look like this

from ase.build import molecule

from ase.calculators.onetep import Onetep

water = molecule('H2O')

keywords = {
    'xc' : 'PBE',
    'do_properties' : True,
    'cutoff_energy' : 35,
    'output_detail': 'verbose',
    'elec_energy_tol': 1.0e-5/len(water),
}

calc = Onetep(keywords=keywords)

water.calc = calc
water.get_potential_energy()
```

Make sure that the ONETEP command being used contains `srun` for example: `command = srun /path/to/onetep/binary`. Otherwise the job will not dispatch correctly on the compute nodes. This is no different from launching a normal job, with the expection that ASE takes care of the input file and the command to be launched.

3.6.3 Archer2

Archer2 is a Cray system, and the conda module is **not** available. You should install it by having a look at the instruction at the end of this document. **One of the Archer2's particularity to keep in mind is that compute nodes only have access to the scratch space and not to the home directory.** You should make sure that every file which will be used during the calculation is accessible from the scratch space, most likely this will be: the input files, the pseudopotentials, the executable and conda. This also means that if you are using the ase config file, you should make sure to change its location with the `ASE_CONFIG_PATH` environment variable to the scratch space. Once this is done you should have a working environment to run ASE on Archer2.

3.6.4 Iridis5

Iridis5 is an Intel based HPC, with conda available as a module. You can alternavely install your own Conda, following the instruction at the end of this document if you want it. There is no particularity to keep in mind when running ASE on Iridis5, you can use the conda module to create an environment with the required packages. You can then submit a job with the python script directly or with a bash script as shown above. Make sure to use `srun` in the command to dispatch the job on the compute nodes.

3.6.5 Young

The only particularity of Young is that `srun` is not available, instead a home-made wrapper around *mpirun* is made avaible (*gerun*). **This will not cause limitations as long as you keep each job to serial execution.** For example, if you use the ASE NEB module with threading, i.e. launching multiple ONETEP in parallel in the same PBS job, gerun will most likely not distribute the job correctly, and the calculation will either fail, or be very slow. The only way around this is to make use of the `mpirun` command directly and specifying the node to use for each job. Which will not be detailed here, you should probably use another HPC for this kind of calculation.

3.7 Other python packages

Other packages that can be used with Onetep + ASE are numerous, here we do mini-tutorials for some of them.

3.7.1 DFTD3/DFTD4

DFTD3 and DFTD4 are dispersion correction methods that can be used with ONETEP. These packages also interface with ASE, which is why they can be used in conjunction with ONETEP. To install DFTD3 or DFTD4, you can use the conda package manager. Here is how to install them:

```
conda install -c conda-forge dftd3-python
conda install -c conda-forge dftd4-python
```

If you really care about the performance you should probably compile them yourself, although the performance gain should probably be minimal. After installation they can be used in the ASE calculator as follows:

```
from ase.build import molecule
from ase.calculators.mixing import SumCalculator
from ase.calculators.onetep import Onetep
from dftd4.ase import DFTD4

atoms = molecule('H2O')
```

(continues on next page)

(continued from previous page)

```
calc = SumCalculator([DFTD4(method="PBE"), Onetep(xc="PBE")])
atoms.calc = calc

atoms.get_potential_energy()
```

For DFTD3 the code is pretty much the same, just replace `DFTD4` by `DFTD3`. The DFTD3 version requires to have `method` and `damping` parameters set at all times. With both versions you can pass an additional parameter `params_tweaks` where you can manually override the internal D3 parameters, see the documentation for more information.

3.7.2 Alloy Catalysis Automated Toolkit (ACAT)

ACAT (https://gitlab.com/asm-dtu/acat) is a python package that can be used to automate the setup of ONETEP calculations for (alloy) catalysis. ACAT can be used in conjunction with ASE, and can be installed using pip:

```
pip install acat
```

The package allows many operations on both surfaces and nanoclusters, the two main classes are the `ClusterAdsorptionSites` and the `SlabAdsorptionSites`. Which are used to detect all possible binding sites of your systems. Here is a complete example to create ONETEP input files for an alloyed nanocluster:

```
from pathlib import Path

from acat.adsorption_sites import ClusterAdsorptionSites
from acat.build.action import add_adsorbate_to_site
from ase.cluster import Octahedron
from ase.io import write

calc_dir = Path("alloy_project_tutorial")
calc_dir.mkdir(exist_ok=True)

atoms = Octahedron("Ni", length=7, cutoff=2)

# Let's create our alloy
for atom in atoms:
    if atom.index % 2 == 0:
        atom.symbol = "Pt"

atoms.center(vacuum=5.0)

# We create the ACAT object with our parameters,
# Many more are available, check the documentation
cas = ClusterAdsorptionSites(
    atoms,
```

(continues on next page)

(continued from previous page)

```
    composition_effect=True,
    label_sites=True,
    surrogate_metal="Ni",
)

# Only unique sites, we don't want to duplicate calculations
sites = cas.get_unique_sites(unique_composition=True)

for site in sites:
    # add_adsorbate_to_site is modifies the object in place
    # so we copy it to avoid modifying the original object
    tmp = atoms.copy()

    add_adsorbate_to_site(tmp, "O", site)

    # We create a unique custom label based on the information
    label = (f"{tmp.get_chemical_formula(mode='metal').lower()}"
             f"_{site['surface']}_{site['site']}_{site['label']}")

    # The directory for this specific calculation
    current_dir = calc_dir / label
    current_dir.mkdir(exist_ok=True)

    # ASE can of course, write onetep input files
    # In practice you would have to specify keywords and␣
↪pseudopotentials
    write(current_dir / "onetep.dat", tmp, format="onetep-in")
```

You will have a directory called *alloy_project_tutorial* with a subdirectory for each adsorption site, each containing an input file for ONETEP. You can then run these input files manually or with ASE as shown in the previous examples. Alternatively you can visualise them using the `ase gui` tool.

3.7.3 Phonopy

Phonopy (https://github.com/phonopy/phonopy) is a python package that can be used to calculate phonon properties of materials. and can be installed using pip or conda:

```
pip install phonopy
```

Phonopy can be used to calculate the phonon band structure of a material. Usually everything is done using the CLI but I personnaly prefer to use the API directly, here is an example for a water molecule:

```
from ase.build import molecule
from phonopy import Phonopy
from phonopy.structure.atoms import PhonopyAtoms
```

(continues on next page)

(continued from previous page)

```
from ase.calculators.onetep import Onetep

water = molecule('H2O', vacuum=10)

calc = Onetep()

phonopy_atoms = PhonopyAtoms(symbols=water.get_chemical_symbols(),
                             positions=water.get_positions(),
                             cell=water.get_cell())

phonopy = Phonopy(phonopy_atoms, supercell_matrix=[[1, 0, 0], [0, 1,
↪ 0], [0, 0, 1]])

phonopy.generate_displacements(distance=0.01)

displacements = phonopy.supercells_with_displacements

forces = []

for i, disp in enumerate(displacements):

    disp_dir = Path(f"displacement_{i}")
    disp_dir.mkdir(exist_ok=True)

    atoms = Atoms(disp.get_chemical_symbols(),
                  disp.get_positions(),
                  cell=disp.get_cell()
    )

    calc.directory = str(disp_dir)

    atoms.calc = calc

    forces.append(atoms.get_forces())

phonopy.forces = forces
phonopy.produce_force_constants()

phonon.save("ifc.yaml", settings={'force_constants': True})

print(phonon.get_frequencies_with_eigenvectors((0, 0, 0))[0]*33.356)
```

With the annoying fact that the `Atoms`` object has to be manually transfered to `PhonopyAtoms` back and forth. The phonon frequencies are in THz, to convert them to cm-1 you have to multiply by 33.356. The *ifc.yaml* file can be used for further processing. See the phonopy documentation for more information.

3.7.4 Many more

There are many more packages that can be used with ONETEP and ASE. Some of them are listed below:

- **pymatgen**: A python package for materials analysis, which can be used to generate structures, calculate band structures, and much more. (https://github.com/materialsproject/pymatgen)
- **phono3py**: A python package for calculating phonon lifetime and thermal conductivity. (https://github.com/phonopy/phono3py)
- **HiPhive**: A python package to compute higher order force constants without using a specific set of configurations. (https://hiphive.materialsmodeling.org/index.html)
- **Sella**: Sella is a utility primarily intended for refining approximate saddle point geometries. Interfaces well with ASE. (https://github.com/zadorlab/sella)
- **QuAcc**: The Quantum Accelerator (QuAcc) is a python package that can be used to create automated workflows and run them concurrently with workflow managers like Parsl, Dask or Covalent. ONETEP has an interface and a few recipes. (https://github.com/Quantum-Accelerators/quacc)

3.8 Conda for the Impatient

3.8.1 Why Conda?

- **Do not pollute your system-wide python, you might regret it**: Conda creates isolated environments, keeping your system Python clean and preventing conflicts between different projects.
- **Stop compiling your tools, use binaries by Conda**: Conda can manage packages for various languages, including R, C++, and Fortran, making it a versatile tool for scientific computing.
- **Complement Conda with pip**: While Conda handles most python package installations, you might occasionally need pip for packages not available in Conda repositories.
- **Conda is self-contained**: Install it everywere, no need for root access. Even HPC systems encourage the use of Conda. Conda will not break your system, and you can remove it easily.

3.8.2 Installing Mambaforge on Linux

1. Download the Mambaforge installer (Linux x86_64) from the Conda Forge repository:

   ```
   wget https://github.com/conda-forge/miniforge/releases/latest/download/Mambaforge-Linux-x86_64.sh
   ```

2. Run the installer:

   ```
   bash Mambaforge-Linux-x86_64.sh
   ```

3. Follow the prompts, agreeing to the license and choosing the installation location.

4. Initialize Mambaforge by running:

   ```
   conda init
   ```

5. Close and reopen your terminal for the changes to take effect.

3.8.3 Installing Conda on Windows

To install Conda on Windows, follow these steps:

1. Visit the official Anaconda website (https://www.anaconda.com) and download the Anaconda Navigator.

2. Run the installer and follow the installation prompts. Make sure to select the option to add Conda to your system's PATH environment variable.

3. Once the installation is complete, open the Anaconda Navigator application to manage packages and environments. You can create environments, install packages, and launch Jupyter notebooks directly from the Navigator interface.

4. If you want to install python packages that are only available through pip you can launch a terminal from the navigator inside the environment you want to install the package and run *pip install package_name*

3.8.4 Creating and Managing Environments

Create a new environment:

```
conda create --name myenv
```

Activate the environment:

```
conda activate myenv
```

Deactivate the environment:

```
conda deactivate
```

3.8.5 Installing Packages

Install packages in the active environment:

```
conda install numpy pandas
```

For packages not available in Conda repositories, use pip:

```
pip install somepackage
```

3.8.6 Updating and Removing Packages

Update a package:

```
conda update somepackage
```

Remove a package:

```
conda remove somepackage
```

Update all packages in the current environment:

```
conda update --all
```

3.8.7 Managing Environments

List all environments:

```
conda env list
```

Remove an environment:

```
conda env remove --name myenv
```

Export an environment to a YAML file:

```
conda env export > environment.yml
```

Create an environment from a YAML file:

```
conda env create -f environment.yml
```

CHAPTER

FOUR

TUTORIAL 3: SETTING UP SPIN-POLARISED CALCULATIONS

Author
Nicholas Hine

Date
July 2024

4.1 Input files

TBC

This completes tutorial 3.

Files for this tutorial:

- `SiH4.dat`

CHAPTER

FIVE

TUTORIAL 4: GEOMETRY OPTIMIZATION

Author
Simon M.-M. Dubois

Date
June 2021 (revised Aug 2023)

5.1 Introduction

This tutorial aims at showing how to run a simple geometry optimization with ONETEP.

Geometry optimization is one of the primary tasks in quantum simulation. The essence of the calculation is for the constituting atoms to be moved to the positions where the total energy is minimal. In general, this can be tackled efficiently if the forces on the atoms can be computed. Over the past twenty years, various schemes have been derived to solve this problem in the framework of ab initio calculations. These range from simple approaches based on molecular dynamics, such as the steepest descent and damped dynamics methods, to the more sophisticated conjugated gradient and Quasi-Newton methods.

The geometry optimization scheme implemented in ONETEP relies on the isolation of the atomic and electronic subsystems (i.e. the Born-Oppenheimer approximation). For a given configuration of the ionic positions, the electronic degrees of freedom are completely relaxed so that the electronic subsystem stays on the Born-Oppenheimer surface. All the possible configurations of the ionic positions therefore define a multi-dimensional potential energy surface for which we want to find the global minimum. The atomic forces are calculated by application of the Hellmann-Feynman theorem and the ionic positions are moved around by means of the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method in order to find the minimum of the potential energy. At this point, one has to keep in mind that several local minima may be present in the configuration space and the algorithm can get trapped in one of those. Therefore, despite the sophistication of the minimization method, the location of a global minimum still requires the intuition of a good starting point.

The calculation flow of a geometry optimization in ONETEP is a three step process:

- Given an ionic configuration, the electronic degrees of freedom are relaxed (cfr. self-consistent optimization of the density kernel and NGWFs).
- The total energy and atomic forces are computed and compared with those of previous ionic configurations. The threshold chosen as stopping criterion for the geometry optimization is

tested.

- The atomic position are updated by means of the BFGS algorithm.

5.1.1 The Ethene Molecule

As a first example, we will deal with the geometry optimization run a geometry optimization of the ethene molecule. To do this, we need to set:

```
task : GeometryOptimization
```

For the SCF part, try running a calculation with an energy cutoff of about 650 eV, NGWF radii of about 6.0 Bohr and a cubic simulation cell of side-length 40 Bohr:

```
cutoff_energy          : 650.0 eV
output_detail : VERBOSE

%block species
H  H  1  1  6.75
C  C  6  4  6.75
%endblock species

%block lattice_cart
Ang
20.0 0.0 0.0
0.0 20.0 0.0
0.0 0.0 20.0
%endblock lattice_cart

%block positions_abs
Ang
C  5.0000000000000000  5.9228319999999997  6.9051750000000007
C  5.0000000000000000  5.9228319999999997  5.5702150000000001
H  5.0000000000000000  6.8456639999999993  7.4753900000000009
H  5.0000000000000000  5.0000000000000000  7.4753900000000009
H  5.0000000000000000  6.8456639999999993  5.0000000000000000
H  5.0000000000000000  5.0000000000000000  5.0000000000000000
%endblock positions_abs
```

You will also need the pseudopotential block:

```
%block species_pot
H   hydrogen.recpot
C   carbon.recpot
%endblock species_pot
```

A full example input/output file can be found `ethene.tar.gz`.

Now you are ready to run ONTEP:

```
mpirun -n 2 onetep ethene.dat | tee ethene.out
```

The calculation should take ~15 min on 2 MPI processes (on a Intel Xeon Silver 4114 cpu). In the meantime you may want to repeat the procedure with varying parameters in order to converge the calculation with respect to the cutoff energy, the NGWF radii, as well as the size of the simulation cell. Besides, if you aim to compute a properties (e.g. the C-C bond length) with a given computational accuracy (e.g. 0.005 Ang), you should also check that the `geom_max_iter` and `ngwf_treshold_orig` parameters do not prevent to reach the desired accuracy.

The output of ONETEP consists principally of two files: `ethene.out` (the main output file) and `ethene.geom`. This latter contains one block of information for each iteration of the geometry optimization. Each block looks like:

```
                            4
             -1.37265351E+001     -1.37265351E+001                    ␣
↪     <-- E
              3.77945227E+001      0.00000000E+000      0.
↪00000000E+000  <-- h
              0.00000000E+000      3.77945227E+001      0.
↪00000000E+000  <-- h
              0.00000000E+000      0.00000000E+000      3.
↪77945227E+001  <-- h
C     1       9.44793440E+000      1.11928879E+001      1.
↪30335653E+001  <-- R
C     1       9.44790699E+000      1.11928871E+001      1.
↪05404458E+001  <-- R
H     1       9.44899454E+000      1.29429511E+001      1.
↪41255628E+001  <-- R
H     1       9.44900396E+000      9.44173342E+000      1.
↪41253623E+001  <-- R
H     1       9.44897328E+000      1.29435030E+001      9.
↪45003514E+000  <-- R
H     1       9.44897086E+000      9.44121996E+000      9.
↪45024026E+000  <-- R
C     1      -1.75749466E-005     -2.60718796E-004     -3.74725290E-
↪004  <-- F
C     1      -1.20705498E-005     -2.14434259E-004      2.65511453E-
↪004  <-- F
H     1       1.13344672E-005      2.37875189E-004      1.82501034E-
↪005  <-- F
H     1       9.56687732E-006      1.39393204E-005     -1.03600237E-
↪004  <-- F
H     1       4.64762662E-006      1.58612511E-004      4.33446848E-
↪005  <-- F
H     1       4.09652530E-006      6.47260346E-005      1.51219285E-
↪004  <-- F
```

where all values are in Hartree atomic units and

- The first line is the iteration number.

- The second line is the total energy.
- The next three lines are the lattice vectors expressed in Caresian coordinates.
- The next N lines (where N is the number of atoms) give the atomic coordinates.
- The following N lines give the atomic forces.

The main informations regarding the geometry optimization are gathered in the `ethene.geom` file, however you may want to visualize the results in a glimpse. You can use the perl script `geom2xyz` to generate a `.xyz` file containing the atomic coordinates at each iteration of the geometry optimization:

```
chmod 700 geom2xyz
geom2xyz ethene.geom
```

This should produce a file `ethene.xyz` that you can visualize with your favourite package (e.g. XCrysDEN). Though the film of the relaxation provides you with crucial information such as the appearance of dissociation, symmetry breaking, etc. It is a good practice to keep track of the energy and forces at each iteration in order to assess the relaxation process. The `-- E` tag which labels the total energies in the `ethene.geom` file may be used for that purpose. Create a new file `ethene_energy.dat` and plot the evolution of the total energy using:

```
$ grep ' E' ethene.geom | awk '{print $1}' > ethene_energy.dat
$ gnuplot plot with lines 'ethene_energy.dat'
```

You should notice that the total energy of the system decreases monotonically. Similarly, you can keep track of the maximum rms force on the ions at each iteration by running:

```
$ grep "<-- BFGS" ethene.out | grep "|F|max"
```

This should produce you something like:

```
|   |F|max    |    2.038842E-002 |    2.000000E-003 |     Eh/Bohr | No ␣
↪| <-- BFGS
|   |F|max    |    3.567221E-003 |    2.000000E-003 |     Eh/Bohr | No ␣
↪| <-- BFGS
|   |F|max    |    5.188186E-003 |    2.000000E-003 |     Eh/Bohr | No ␣
↪| <-- BFGS
|   |F|max    |    1.375629E-003 |    2.000000E-003 |     Eh/Bohr | Yes␣
↪| <-- BFGS
|   |F|max    |    4.568394E-004 |    2.000000E-003 |     Eh/Bohr | Yes␣
↪| <-- BFGS
```

The second column is the calculated value of the maximum rms force on the atoms, the third column is the force threshold that the code is trying to achieve, the fourth column provides the units, and the fifth column informs you as to whether convergence of the force has been achieved or not. You may visualize this information using gnuplot:

```
$ grep "<-- BFGS" ethene.out | grep "|F|max" | awk '{print $4}' >␣
↪ethene_force.dat
$ gnuplot plot with lines 'ethene_force.dat'
```

You are now familiar with the geometry optimization scheme in ONETEP. You might examine in more details the input variables that allow to control the process. The keywords associated with the geometry optimization all start with the `geom_` prefix. Their description is found on the ONETEP Documentation. In particular, take a few minutes to have a look at the variables:

```
geom_max_iter
geom_convergence_win
geom_disp_tol
geom_energy_tol
geom_force_tol
```

Though their default values may appear to be convenient in most circumstances, these latter are the very basic input variables to master before launching a geometry optimization. Here, it is important to note that the three tolerance criteria (`geom_disp_tol`, `geom_energy_tol`, and `geom_force_tol`) are not exclusive. The three criteria have to be satisfied in order for the optimization to stop. You might have noticed that during the relaxation of ethene molecule, the default threshold imposed on the atomic forces (`geom_force_tol : 0.02 Ha/Bohr`) has been reached before the one associated with the convergence of the energy (`geom_energy_tol : 10e-06 Ha/Atom`).

Like all the quasi-Newton schemes, the BFGS algorithm accumulates information about the Hessian matrix. As the the number of iteration increases, BFGS improves its knowledge of the the potential energy surface around the minimum and the matrix used to build the quadratic model of the potential energy surface converges towards the true Hessian matrix corresponding to the local minimum. However, the Hessian matrix is poorly approximated during the first few relaxation steps. It is therefore important to properly initialize the BFGS scheme. This may be conveniently done by means of a unique parameter `geom_frequency_est`. For the best efficiency, its value should corresponds to a rough estimation of the average of the optical phonon frequencies at the center of the Brillouin zone.

In the case of the ethene molecule, the average of the experimentally reported vibration frequencies is 0.0081 Hartree. This value is very close to the default setting of `geom_frequency_est` and we do not expect any speed-up of the relaxation process by adjusting it.

In various circumstances, it may appears convenient to impose some constraints to the atomic positions during the geometry optimization. Note that in the case of molecular systems it is often a good idea to keep an atom or an axis fixed during the optimization process in order to avoid losing computational time due to the rotations and/or translations of the system. Therefore it is worth having a quick look at the meaning of the variables:

```
species
species_constraints
```

Finally, it is worth noting that, when running a geometry optimization, ONETEP produces a `.continuation` file. This latter contains all the information regarding the optimization process and can be very helpful to restart an optimization from a previous run. In such a case, the only thing you will need is to turn on the flag `geom_continuation`. In the same line of thought, a appropriate use of the keywords that control the reading/writting actions of the code, may help you to save some precious computational time:

```
write_converged_dkngwfs
read_denskern
read_tightbox_ngwfs
```

For example, to use `write_converged_dkngwfs : T` is a good practice when running a molecular optimization as it avoid you to lose time in writing the density kernels and NGWFs on the disk.

5.1.2 The Sucrose Molecule

At this point, you should be familiar with most of the keywords needed to run a proper geometry optimization. Therefore, we suggest you to leave the ethene molecule and to try to optimize a larger organic molecule. You can find an example input file for the sucrose molecule `here`. You should edit and read it carefully. You see that the `write_converged_dkngwfs` flag has been activated. In addition, the values of `ngwf_cg_max_step` and `lnv_cg_max_step` have been increased in order to allow unconstrained line search during the conjugate gradient optimization of the density kernel and NGWFs respectively.

The calculation should take a bit more than an hour (with 16 MPI processes on a Intel Xeon Silver 4114 CPU). Keeping trace of the atomic forces, you should notice a rapid decrease of the maximum rms force on the ions during the first relaxation steps. However, the hydrogen atoms tend to wiggle quite a lot and it takes a some time for the positions to settle down according to relaxation criteria.

```
| |F|max | 1.212754E-002 | 2.000000E-003 | Ha/Bohr | No | <-- BFGS
| |F|max | 1.407689E-002 | 2.000000E-003 | Ha/Bohr | No | <-- BFGS
| |F|max | 1.253042E-002 | 2.000000E-003 | Ha/Bohr | No | <-- BFGS
| |F|max | 4.859480E-003 | 2.000000E-003 | Ha/Bohr | No | <-- BFGS
| |F|max | 1.052111E-002 | 2.000000E-003 | Ha/Bohr | No | <-- BFGS
| |F|max | 5.953036E-003 | 2.000000E-003 | Ha/Bohr | No | <-- BFGS
| |F|max | 6.344620E-003 | 2.000000E-003 | Ha/Bohr | No | <-- BFGS
| |F|max | 6.587572E-003 | 2.000000E-003 | Ha/Bohr | No | <-- BFGS
| |F|max | 5.241521E-003 | 2.000000E-003 | Ha/Bohr | No | <-- BFGS
| |F|max | 5.455889E-003 | 2.000000E-003 | Ha/Bohr | No | <-- BFGS
| |F|max | 3.623343E-003 | 2.000000E-003 | Ha/Bohr | No | <-- BFGS
```

5.1.3 Periodic Crystals

Here above, the geometry optimization scheme has been illustrated by means of two molecular systems. Obviously, the same scheme holds for periodic crystals. As an example, we will investigate the adsorption of ammonia on a (10,8) carbon nanotube.

The carbon nanotube considered here contains 488 carbon atoms in its unit-cell and its chiral periodicity is of 62.87 Bohr.

Following the prescriptions stated above, you should be able to write an input file for the nanotube (an example is given in `here`.). Note that for large systems, the spatial expansion of the density

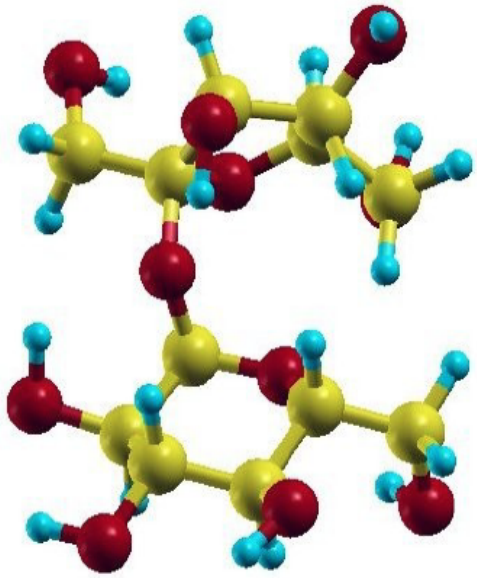


Fig. 5.1: Ball stick representation of the sucrose molecule.

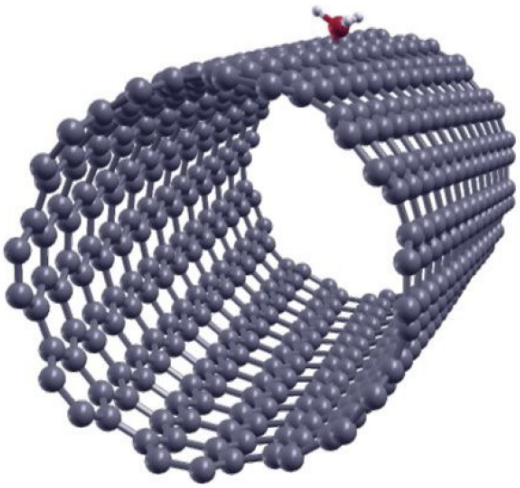


Fig. 5.2: Ball-stick representation of the ammonia adsorbed on a CNT (10,8)

kernel has to be truncated in order to achieve the linear scaling. This can be done with the `kernel_cutoff` variable. Obviously, stringent truncation of the density kernel is expected to affect the accuracy of the calculation. Therefore, the cutoff length has to be carefully adjusted. You should already notice a significant decrease of the forces after the first few iterations.

CHAPTER

SIX

TUTORIAL 5: ANALYSIS AND VISUALIZATION

Author
Jacek Dziedzic, Chris-Kriton Skylaris

Date
July 2008 (revised April 2010, July 2023)

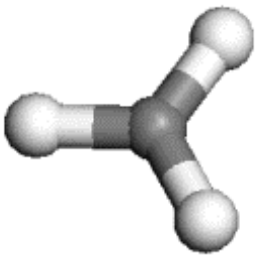
6.1 Introduction

This tutorial demonstrates how to:

- Use ONETEP to calculate various electronic properties,
- Instruct ONETEP to generate files needed for later visualization of orbitals, electronic densities and potentials,
- Visualize these properties using VMD[1],
- Set up and run a calculation on a nanostructure using a cut-off for the density kernel.

6.2 Density, spin density, Kohn-Sham orbitals and the electrostatic potential for CH_3

In this part we will perform a calculation on the CH_3 radical:



[1] VMD (Visual Molecular Dynamics) is a free of charge visualization package available from https://www.ks.uiuc.edu/Research/vmd

As this molecule contains an odd number of electrons we need to perform a spin-polarised (unrestricted) calculation. In ONETEP this is achieved by optimising a different density kernel for the "up" and the "down" spin:

$$\rho\left(\mathbf{r},\mathbf{r}'\right)=\sum_{\alpha\beta}\phi_{\alpha}(\mathbf{r})K^{\alpha\beta(\uparrow)}\phi_{\beta}^{*}\left(\mathbf{r}'\right)+\sum_{\alpha\beta}\phi_{\alpha}(\mathbf{r})K^{\alpha\beta(\downarrow)}\phi_{\beta}^{*}\left(\mathbf{r}'\right)$$

The ONETEP input is in the file `methyl.dat` and the coordinates (in angstroem) are in the file `methyl.pdb`. The ONETEP input file contains the coordinates as well (in atomic units), but not in a form directly readable by visualization packages. The `.pdb` file can be directly visualized in VMD.

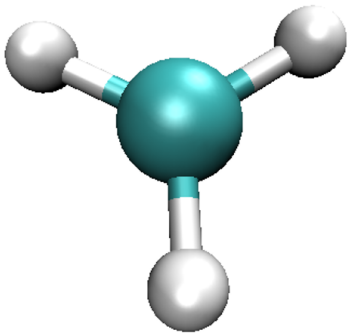
The `methyl.dat` file specifies a single point energy calculation (`TASK SINGLEPOINT`) with a psinc kinetic energy cutoff of 800 eV (`CUTOFF_ENERGY 800.0 eV`), the Perdew-Zunger variant of the LSDA exchange-correlation functional (`XC_FUNCTIONAL CAPZ`) and the spin-polarised option (`SPINPOLARIZED TRUE`). Also notice the input flag `DO_PROPERTIES TRUE`, which proceeds with the calculation of various electronic properties at the end of the single point energy calculation.

Run the input, redirecting the output to a file such as `methyl.out`. We also provide a reference `methyl.out` file. The calculation should take a minute or two to run. Once it completes, you will notice that a number of `.cube` files have been created, including the file `methyl_spindensity.cube`. Let us examine this first. ONETEP can output volumetric data (such as spin densities, charge densities, potentials, etc.) in Gaussian `.cube` format (`CUBE_FORMAT TRUE`), Materials Studio `.grd` format (`GRD_FORMAT TRUE`) and OpenDX `.dx` format (`DX_FORMAT TRUE`). The `.cube` format has the advantage of having the ionic positions output in addition to the volumetric data. In this tutorial we will use the `.cube` format which can be viewed with a number of free molecular visualisation programs. The instructions that follow are assuming that the VMD program can be used to visualize the files but in priciple you can use any other software that can display `.cube` files (such as VESTA, Molekel, gOpenMol, XCrySDens, etc).

Start VMD by typing `vmd` in the terminal, use `File/New molecule/Browse` to find `methyl_spindensity.cube`, then click on `Load` to load the molecule. You should be able to see a crude, line-based representation of the molecule in a separate window. You can now get rid of the `Molecule file browser` window. Choosing `Graphics/Representations...` opens another window which lets you control the look of your molecule. In this window, change the `Drawing Method` from `Lines` to `CPK`, which will render your molecule in a ball-and-stick fashion, with the customary colouring[2]. Increase both `Sphere Resolution` and `Bond Resolution` (30 is a good value) to get rid of the jagged edges. You may wish to adjust `Sphere Scale` and `Bond Radius` to your liking as well.

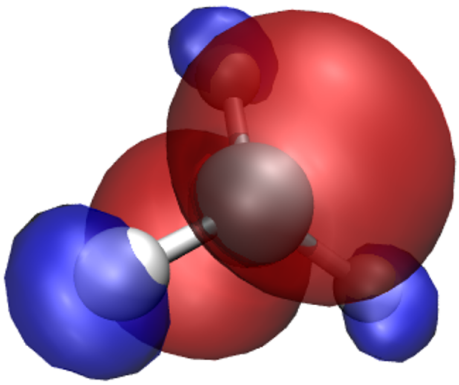
Try dragging with your mouse over the window that shows the molecule to rotate it. Try scrolling the mouse wheel to get closer or further away from the molecule. You may press the = key at any time to reset the view. Pressing the `T` key will get you to `Translate Mode`, where dragging with the mouse translates the molecule, instead of rotating it. To go back to `Rotate Mode`, press `R`. If your mouse lacks the scroll wheel, pressing `S` to go to `Scale Mode` might be of use. You should be able to obtain a representation similar to the one shown here.

[2] The colouring is described here: https://en.wikipedia.org/wiki/CPK_coloring.



So far we've only looked at the nuclei in the system. Let's try some electronic properties, starting from the spin density which we have already loaded, but not visualized yet. A neat thing about VMD is that you can use several representations at once. Thus, we can overlay the spin density isosurfaces on top of the CPK representation of the ions. In the `Graphics/Representations...` window click on `Create Rep`. This will clone the CPK representation, leaving you with two identical representations. Now change one of them to `Isosurface`. Not much will appear initially, because the default way of showing the isosurface is by using *points*. This is computationally cheap, but visually so as well. You can change this under `Draw`, by choosing `Solid Surface`. *Before you do it*, however, make sure to move the `Isovalue` slider to something different than the default 0.0 (or type a value in the box). This is because there is a huge number of points in our system (some 400000) where the spin density is exactly or almost exactly zero (everywhere outside our molecule). Trying to draw a surface through these points usually confuses VMD to the point of crashing or at least stuttering. For this reason it is best to pick any value other than the default of 0.0 to start from, before choosing `Solid Surface`.

Experiment with the settings (`Coloring Method`, `Material`, `Isovalue`) to get a feel for how they work. It makes sense to set `Coloring Method` to `ColorID` here, as this lets us to manually pick a colour for the isosurface (from the drop-down box near `ColorID`). After some adjustments you should obtain an isosurface similar to the one shown here. Do not worry if you cannot get the transparency right – it's – only possible when you render "production quality" images, think of what you see as a draft.



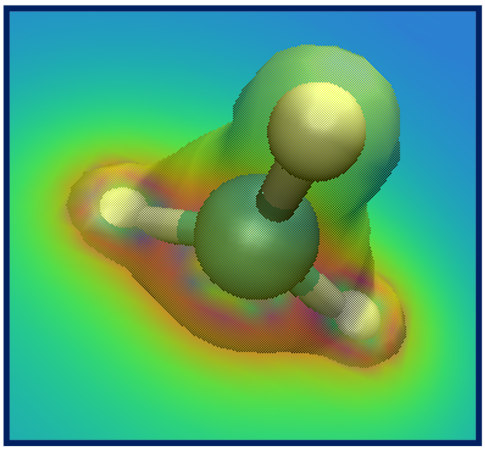
What we have obtained is the textbook picture of the spin density of a methyl radical. It has positive as well as negative regions which is a consequence of the fact that the spatial parts of the Kohn-Sham orbitals for each spin are allowed to be different, even for doubly occupied states.

The properties calculation also produces *Kohn-Sham orbitals*. Their energies for each spin are printed in the output file (try to find them, they are towards the very end, copy them into the table

below) and `.cube` files for the squares of some of the orbitals are also produced. HOMO orbitals are written, separately for each spin, to `methyl_HOMO_DN.cube` and `methyl_HOMO_UP.cube`, and their LUMO counterparts to `methyl_LUMO_DN.cube` and `methyl_LUMO_UP.cube`. Similarly named files contain the orbitals just below the HOMO and just above the LUMO (not provided here, but generated during the calculation).

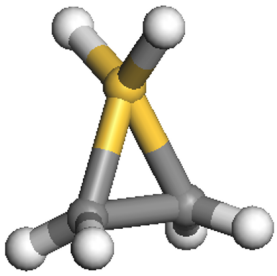
Quantity	Value
Energy of HOMO for spin 1 (UP)	
Energy of LUMO for spin 1 (UP)	
Energy of HOMO for spin 2 (DOWN)	
Energy of LUMO for spin 2 (DOWN)	
Energy of the 1st orbital above LUMO for spin 1 (UP)	
Energy of the 1st orbital below HOMO for spin 1 (UP)	

Finally, let's try visualizing the local potential (sum of the ionic, Hartree (Coulomb) and XC potentials), which is written out to `methyl_potential.cube`. Isosurface plots of potentials can be obtained similarly to the isosurface plots of densities. Let's also try to do a contour plot. This can be accomplished by choosing `VolumeSlice` for `Drawing Method`. Try playing with `Slice Axis` and `Slice Offset` to get the hang of it. Admittedly, the quality of the contour plot is not too good, even if you set `Render Quality` to `High`. It is improved, however, when you create a production image. Try obtaining a composite CPK + isodensity + contour plot similar to the one shown here.



6.3 Visualizing NGWFs and NNHOs for C_2SiH_6

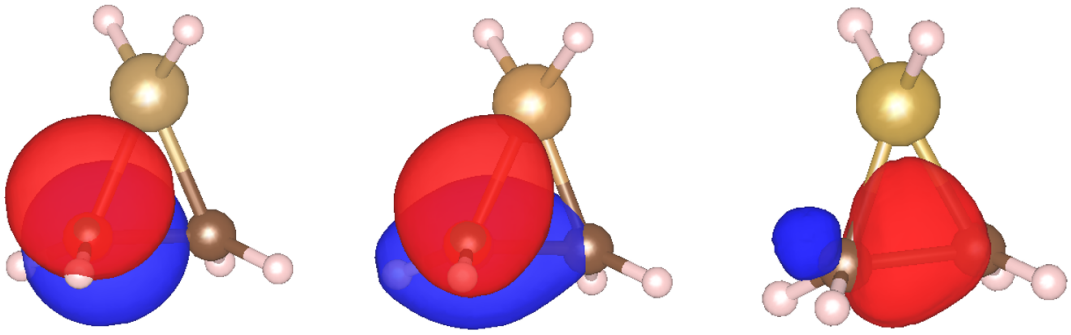
In this example we will perform two sets of calculations on the C_2SiH_6molecule:



The first calculation will use the input file `C2SiH6_NGWF.dat`, which has similar parameters (and, thus, keywords) to the previous example but also contains the `WRITE_NGWF_PLOT TRUE` keyword that allows output of selected NGWFs in the scalarfield formats we discussed earlier (`.cube` by default). The NGWFs that will be outputted are selected by the `species_ngwf_plot` block in which the *species* of atoms whose NGWFs are to be outputted are listed. In this example we output NGWFs of the Si atom and of the first H and C atoms (as written in the input coordinates). The second input file is `C2SiH6_NNHO.dat`, which contains the additional keyword `NNHO TRUE` which instructs ONETEP to perform a same-centre rotation of the NGWFs to transform them to non-orthogonal natural hybrid orbitals (NNHOs). These contain the same information as the NGWFs but are more "natural" as they conform with chemical concepts, such as being directed towards chemical bonds, and physical concepts, as in several of their properties they resemble proper Wannier functions. The mixing of NGWFs to NNHOs is done according to the procedure by Foster and Weinhold (J. P. Foster and F. Weinhold, *J. Am. Chem. Soc.* **102**, 7211 (1980)). For this calculation we will use the PBE GGA exchange-correlation functional (`XC_FUNCTIONAL PBE`).

Run the calculation to completion with the two inputs (in separate directories), it should take no more than five minutes for each of them. Reference outputs are provided here: `C2SiH6_NGWF.out` and here: `C2SiH6_NNHO.out`.

Examine some of the NGWF and NNHO output files. As an example, below we show plots of the third function (NGWF or NNHO) of atom 2 (one of the carbons). Try to obtain similar plots.



Left: Isosurface of initial, unoptimised functions from `initial_atom00002_ngwf03.cube`.
Middle: Isosurface of `final_atom00002_ngwf03.cube` after completion of the `C2SiH6_NGWF.dat` calculation.
Right: `final_atom00002_ngwf03.cube` after completion of the `C2SiH6_NNHO.dat` calculation.

You can observe that initially the function is a p-atomic orbital (as it is initialised by ONETEP).

After the calculation the NGWF is rather distorted but still contains quite a lot of p character. The NNHO however is a mixture of all the 4 NGWFs of the carbon atom and is optimally pointed along the C-C bond. You can quantify these observations by comparing the two output files, `C2SiH6_NGWF.out` and `C2SiH6_NNHO.out`, which contain an `NGWF s/p/d/f Character Analysis` section towards the bottom of the file (thanks to the `NGWF_ANALYSIS TRUE` keyword in the input). You will see how much the NGWFs differ from the NNHOs. Of course all the other quantities (energies, Kohn-Sham orbitals, orbital energies, etc.) are independent of whether you use NGWFs or NNHOs. Check this by completing the table below.

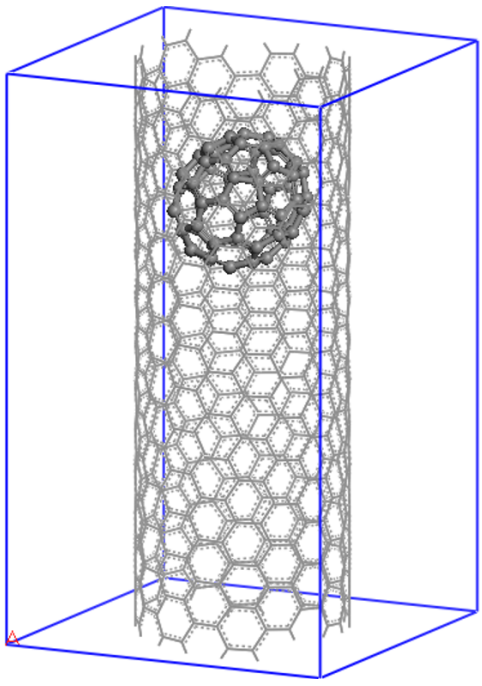
Table 6.1: Calculated binding free energy of catechol to the protein.

Quantity	Value
Total energy of the system	
Energy of HOMO	
Energy of LUMO for spin 2(down)	

Finally, examine the atomic population in the output files (we have asked for it using the keyword `POPN_CALCULATE TRUE` in the input) and confirm that the charges on each atom are consistent with their relative electronegativities.

6.4 A calculation on a nanostructure

Let us now see how to set up and visualize a calculation on a nanostructure whose size is in the region where conventional cubic scaling codes become very inefficient, while linear-scaling codes like ONETEP are still at the beginning of their capabilities. We will perform a calculation on the following “nano-peapod” structure, which consists of a C_{70} fullerene inside a single repeat-unit of a (10,8) carbon nanotube.

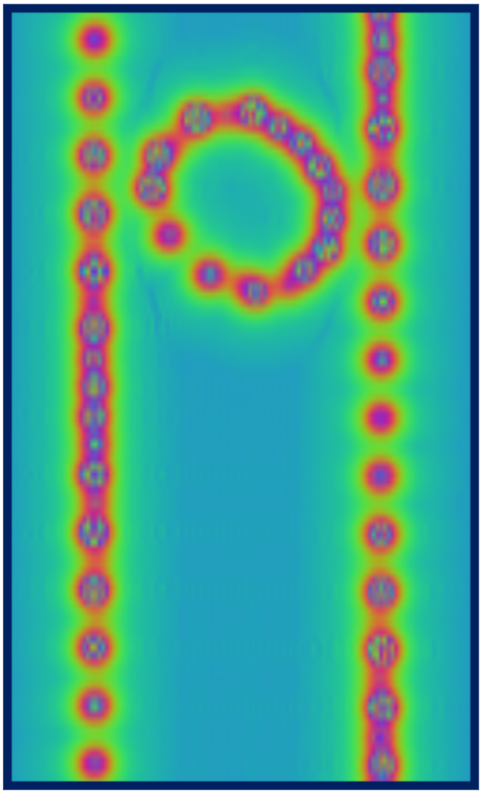


The (10,8) is a chiral nanotube with 488 atoms in each repeat-unit, so the peapod input consists of 558 atoms, with no symmetry, in a unit cell of 20.0 x 20.0 x 33.27 (angstroem), which is equivalent to 37.795 x 37.795 x 62.874 (bohr). The ONETEP input is in the file `C70_in_10-8.dat`. We impose a density kernel cut-off of 30.0 bohr (`KERNEL_CUTOFF 30.0 bohr`) in order to achieve linear-scaling behaviour.

This calculation is best run on a parallel computer, but you can run it on a desktop machine where it should complete in about two-three hours. It took just under 8 minutes when run on 5 nodes (360 CPU cores) in 2023. If you do not want to wait or do not have the sufficient resources, here's the reference output: `C70_in_10-8.out`.

Let us start by examining this file. At the beginning of the calculation the *filling* (the opposite of sparsity) of various matrices is reported. You will notice that the density kernel is not 100% full as a consequence of the cut-off that is imposed in the input. Information about the psinc grid sizes is also provided, including the actual plane-wave cut-off to which they correspond and the size of the FFT box. The calculation converges in 7 NGWF iterations, which is the point where the NGWF gradient threshold set in the input (`NGWF_THRESHOLD_ORIG 0.00003`) has been satisfied. Normally you'd likely use a tighter threshold for extra accuracy (the default is 2E-6).

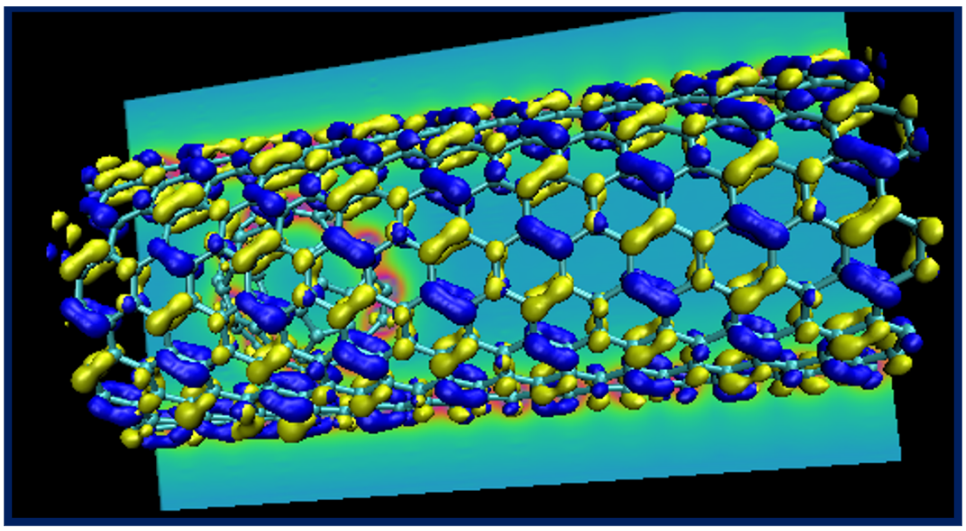
As before, a range of properties are calculated (`DO_PROPERTIES T`). As an example, you can examine the total potential (the sum of ionic, Hartree and exchange-correlation potentials) which is outputted to the file `C70_in_10-8_PROP_potential.cube`. We do not provide this file here due to size considerations. A contour plot on a plane containing the nanotube axis of the potential will look similar to what you see below, which is compatible with the chiral nature of the nanotube and reveals also the asymmetric way in which the oblong C_{70} is is located inside it.



Red regions correspond to large and positive values of the potential (standard electrostatic conventions) and reveal the location of nuclei, whose distance from the plane varies along the axis of the tube,

as a result of the chirality. You can go on and explore other properties of the nano-peapod from the `C70_in_10-8.out` file and the other output files that were produced by the properties calculation.

If you are in an ambitious mood, try creating a fancy plot showing the structure of the nano-peapod system with its HOMO and LUMO orbitals and a contour plot of the potential, similar to the one below.



This completes tutorial 5.

CHAPTER

SEVEN

TUTORIAL 6: TIME-DEPENDENT DFT

Author
Tim Zuehlsdorff

Date
January 2016 (revised August 2023)

7.1 TDDFT in ONETEP

This tutorial aims at showing how to run a simple linear-response TDDFT calculation in ONETEP. Linear-response TDDFT is the main method of choice when computing optical excitations for large system sizes. In this formalism, excitation energies are directly obtained as the solutions to an effective eigenvalue equation. The approach implemented in ONETEP can be made to scale fully linearly with systems size, allowing for the computation of excitations in systems containing thousands of atoms.

In linear-response TDDFT, an excitation is expressed through a transition vector that can be written in the basis of all possible Kohn-Sham transtions from occupied to unoccupied states. Thus unlike in standard ground state DFT, where a good representation of the occupied KohnSham states is sufficient to obtain accurate results, in a TDDFT calculation an equally good representation of a subset of the unoccupied Kohn-Sham space is vital. Since the ONETEP support functions are optimised *in situ* to ideally represent the occupied manifold, they generally provide a relatively poor description of the unoccupied manifold, making them insufficient for describing excitations in a linear-response framework. In practice, this problem is removed by optimising a second set of support functions to ideally span a low energy subset of the conduction space in a post-processing step after a `Singlepoint` calculation. Together, the two sets of support functions form an ideal representation of any given low energy excitation, thus enabling efficient and accurate TDDFT calculations in very large systems.

7.2 Setting up a TDDFT calculation in ONETEP

A TDDFT calculation in ONETEP thus proceeds in three separate steps:

- A standard ground state calculation using `Task: Singlepoint`. The code will write out `.dkn` and `.tightbox_ngwfs` files containing the valence density kernel and the NGWF support functions.
- A conduction optimisation using `Task: Cond`. The code will read in the converged `.dkn` and `.tightbox_ngwfs` files from the ground-state optimisation and perform a postprocessing `conduction optimisation` of some low energy part of the conduction space manifold. It will write out `.dkn_cond` and `.tightbox_ngwfs_cond` files containing an effective density kernel for the low energy conduction space, as well as the NGWF support functions for the conduction space.
- A TDDFT calculation using `Task: Lr_tddft`. The code will read in the density kernels and support functions for both the ground state and the conduction space calculations and then calculate the lowest n excitations of the system, where n is determined by the keyword `lr_tddft_num_states`.

In practice, it is possible to string these tasks together using `Task: Singlepoint Cond Lr_tddft` in an input file that lists appropriate keywords for all three calculation types. The tasks will be performed one after another, without the user having to restart the calculation in between.

7.3 An NN-substituted nanoribbon

We turn to a simple example of an excited state calculation in a periodic system, namely that of an infinitely long graphene nanoribbon with two nitrogen substitutions. We will aim to compute excited states that are associated with the substitution and that are thus relatively localised in character. In a first step, we aim to build an input file for a pristine graphene nanoribbon in periodic boundary conditions. A primitive unit cell for the nanoribbon in question can be found below:

```
%block positions_abs
C 10.90 10.0 0.000
C 13.58 10.0 0.000
H 7.50 10.0 2.327
C 9.56 10.0 2.327
C 14.93 10.0 2.327
H 16.99 10.0 2.327
%endblock positions_abs
```

To generate an appropriate input file, copy the above primitive unit cell block into a new input file called `ribbon.dat`. ONETEP does not make use of any k-point sampling and all calculations are effectively done at the Γ-point. Therefore, in order to accurately simulate periodic structures it is necessary to simulate supercells rather than primitive cells as would be done in standard

plane wave DFT codes. An appropriate supercell from the above primitive cell can be created by repeating the primitive cell 7 times along the z-axis, translating the z-coordinates of the atoms by the lattice vector of the primitive unit cell. The resulting supercell should contain 48 atoms in total.

Now create an appropriate `%block lattice_cart` for the system. The length of the supercell is specified by the length of your primitive cell and the number of repeats of that cell. However, the size of the unit cell in the x and y direction is free to specify by the user. Since the ONETEP method uses periodic boundary conditions in all three directions, it is necessary to include enough vacuum in the x and y direction of the simulation cell to prevent it from interacting with its periodic neighbours. In this case, about 20 a_0 of vacuum between any atoms of different periodic images is a reasonable length.

Using the lessons learnt from previous tutorials add a `%block species` and `%block species_pot` block to your input file. Choose an appropriate NGWF radius (try 8 a_0) and kinetic energy cutoff (around 600 eV) for your calculation. When choosing the NGWF radius, note that ONETEP will not allow you to pick an NGWF diameter that is larger than the dimensions of your simulation cell, as this would cause an NGWF to interact with its periodic image. Finally, perform a singlepoint calculation of the system. You can add a keyword `do_properties: T` to the input file. This will trigger the code to perform a properties calculation, write out a file `.val_bands` containing all Kohn-Sham energies and write cube files of selected Kohn-Sham states around the band gap.

7.3.1 Conduction optimisation

Once the ground state calculation is finished, it is necessary to perform a conduction optimisation. First, set `Task: Cond` and add a new block `%block species_cond` to the input file. This block is made up in an identical way as `%block species` and specifies the number of support functions and their radius in the conduction optimisation. In many practical examples, it is often advisable to choose a larger NGWF radius for the conduction optimisation than for the ground state calculation, especially if one is interested in lightly bound states, as unoccupied Kohn-Sham states tend to be more diffuse than occupied ones (try 10 a_0 for example).

A second keyword required for the conduction optimisation is the number of conduction states that should be explicitly optimised (`cond_num_states`). It is normally advisable to optimise only well-bound states as unbound states are difficult to describe with localised orbitals. In order to do so, have a look at the file `.val_bands` and count the number of Kohn-Sham states with negative energy that are unoccupied. Note that `cond_num_states` expects the number of electrons to be optimised as an input, thus in order to optimise the five lowest unoccupied Kohn-Sham states in a spin-degenerate system, one should choose `cond_num_states: 10`.

Once you have run the calculation, have a look at the output. You will find that ONETEP has generated a number of files such as cube files of unoccupied Kohn-Sham states expressed in terms of the optimised conduction NGWFs.

7.3.2 TDDFT calculation

We can now perform a full ONETEP TDDFT calculation of the system at hand. To do so, set `Task: Lr_tddft` in the input file. Furthermore, add the keywords `lr_tddft_num_states: 6`, `lr_tddft_write_densities: T` and `lr_tddft_analysis: T`. The code will compute the lowest 8 singlet excitations of the system and generate cube files for the electron, hole and transition density for each excitation that can be visualised. Furthermore `lr_tddft_analysis: T` triggers a breakdown of the converged TDDFT eigenvectors into Kohn-Sham transitions, allowing you to study which are the dominant transitions for each excitations.

Once you have performed the TDDFT calculation, look at the output file. You will see that the excitation energies and oscillator strengths for each of the excitations are printed out, as well as a detailed breakdown of excitation energies into Kohn-Sham transtions. Have a look at some of the cube files produced. Where are the excitations located in the system?

7.3.3 Nitrogen substitution

We can now move on from the case of the pristine nanoribbon to one with two nitrogen substitutions. For this purpose, copy the input file `ribbon.dat` to a new file `ribbon_NN.dat`. In that file, remove two C-H from the `%block positions_abs` that are opposite to each other in the ribbon, and replace them by two N at the same positions where the C were located.

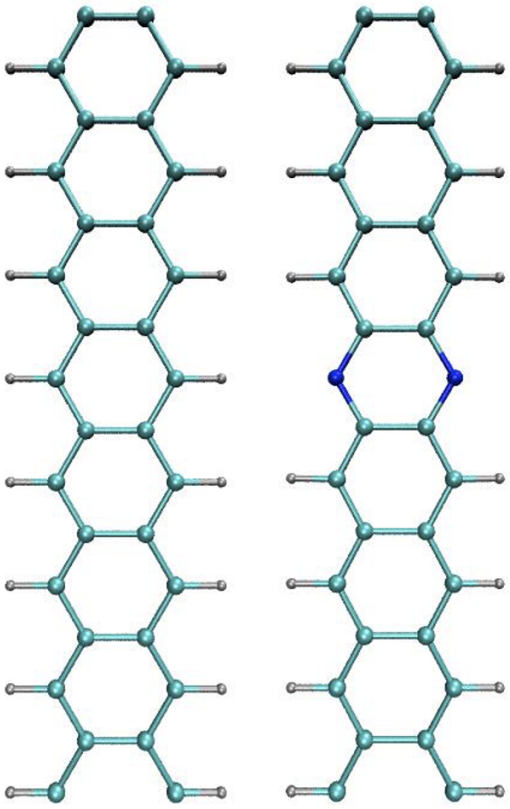


Fig. 7.1: Left: Pristine nanoribbon. Right: Nanoribbon with two carbons and two hydrogens substituted for two nitrogens

Note that in order to run the calculation, you will have to add the nitrogen species to the `%block species_pot`, `%block species` and `%block species_cond` blocks. Change the task to `Task: Singlepoint Cond Lr_tddft`. The code will run a ground state and conduction optimisation, followed by a TDDFT calculation for the full system. Have a look at the output. How do the excited states change due to the nitrogen substitutions? Where is each excited state located within the system?

7.4 Additional tasks

Substituting nitrogen atoms in the same place as carbon atoms does not yield a relaxed ground state structure, as the N-C bond is not of the same length as the C-C bond. Thus in order to obtain more realistic results for the substituted system, perform a geometry optimisation (see tutorial 4), followed by a ground state, conduction and TDDFT calculation of the full system. How do the results change? Furthermore, create a system where the nitrogen atoms are not substituted at exactly opposite positions in the structure, but an asymmetry along the z-axis is introduced. How does the character of the low energy excitations change?

7.5 Input files

All the files needed for the simulation can be downloaded from

- `ribbon_pristine.dat`
- `ribbon_pristine_NN.dat`
- `ribbon_pristine.out`
- `ribbon_pristine_NN.out`
- `carbon.recpot`
- `hydrogen.recpot`
- `nitrogen.recpot`

CHAPTER

EIGHT

TUTORIAL 7: SPECTRAL FUNCTION UNFOLDING

Author
Nicholas Hine

Date
July 2024

This completes tutorial 7.

8.1 Input files

All the files needed for the simulation can be downloaded from

- `ribbon_pristine.dat`

CHAPTER

NINE

TUTORIAL 8: IMPLICIT SOLVATION, VISUALISATION AND PROPERTIES: PROTEIN-LIGAND FREE ENERGY OF BINDING FOR THE T4 LYSOZYME

Author
Lennart Gundelach, Jacek Dziedzic

Date
June 2021 (revised June 2023)

9.1 Introduction

9.1.1 Protein-Ligand Free Energies of Binding

The binding free energy is a measure of the affinity of the process by which two molecules form a complex by non-covalent association. An example of this, of central importance in biology, is the binding of a ligand to a protein. Many methods to computationally approximate the binding free energies of protein-ligand interactions have been proposed with the ultimate goal of computationally predicting small molecule drug candidates which bind strongly to the protein of interest.

9.1.2 Quantum Mechanics in Binding Free Energies

A key limitation common to most computational methods of estimating binding free energies is the assumption of the validity of classical mechanics. The atoms and electrons that constitute biological molecules, like proteins, are, however, governed by the laws of quantum mechanics. Charge transfer, polarization and non-local interactions are not captured by traditional classical mechanical force-fields. Thus, a true description of protein-ligand binding requires a quantum mechanical (QM) treatment of the problem. In theory, a full, *ab-initio* QM approach would be system-independent, parameter-free and would describe the full spectrum of physical phenomena at work.

Unfortunately, high-level QM methods like coupled-cluster (CC) are prohibitively expensive and often have cubic or worse scaling with system size. Thus, even the ligands alone are often too large for routine calculations with these methods.

9.1.3 Linear Scaling Density Functional Theory

Due to the cubic scaling of conventional density functional theory, full-protein calculations on many thousands of atoms are not feasible. To study larger systems, linear-scaling versions of DFT have been developed [Bowler2012]. The ONETEP code [Prentice2020_T8] is one such linear-scaling DFT implementation, exploiting hybrid MPI-OMP parallelism [Wilkinson2014] for efficient and scalable calculations. The unique characteristic of ONETEP is that even though it is linear-scaling, it is able to retain large basis set accuracy as in conventional cubic-scaling DFT calculations. The implicit solvation model is a minimal-parameter Poisson-Boltzmann (PB) based model which is implemented self-consistently as part of the DFT calculation [Dziedzic2011] [Womack2018] and uses the smeared-ion formalism and electron-density iso-surfaces to construct solute cavities.

9.1.4 T4 Lysozyme

The protein under investigation in this tutorial is a double mutant of the T4 lysozyme (L99A/M102Q). This protein has been artificially mutated to form a buried polar binding site and has served as a model or benchmark system for various protein-ligand binding free energy studies [Mobley2017]. Although this protein is not directly pharmaceutically relevant, it is a useful model system due to it its relatively small size (2500 atoms), structural rigidity and well-defined, buried binding site, which can accommodate a wide variety of ligands. Fig. 9.1 shows the ligand catechol inside the buried binding site of the T4 lysozyme L99A/M102Q mutant. PDB files of the complex, host and ligand are provided as part of this tutorial for you to visualize the system. The picture shown uses the `NewCartoon` representation for the protein with coloring based on secondary structure and `CPK` (ball-and-stick) for the ligand with element based coloring.

9.1.5 QM-PBSA Binding Free Energies

In this tutorial we will calculate the binding free energy of catechol to the T4 lysozyme L99A/M102Q mutant. We will employ a simplified QM-PBSA approach [Fox2014] [Gundelach2021] on a single snapshot of the protein-ligand complex.

The QM-PBSA approach is a quantum-mechanical adaptation of traditional MM-PBSA, which is an end-point, implicit solvent, binding free energy method. In this approach, the binding free energy is given by

$$\Delta G_{\text{bind}} = G_{\text{complex}} - G_{\text{host}} - G_{\text{ligand}}, \tag{9.1}$$

where G_{complex}, G_{host}, and G_{ligand} is the free energy of, respectively, the complex, host and ligand in an implicit solvent. Each of these can be decomposed into three terms,

$$G = E + \Delta G_{\text{solv}} - TS, \tag{9.2}$$

where E is the total gas-phase energy, ΔG_{solv} is the free energy of solvation and $-TS$ is an entropy correction term. In this tutorial, the entropy term will be ignored, as it is usually calculated in other programs using normal mode analysis. The linear-scaling DFT code ONETEP will be used to calculate the gas-phase and solvation free energy of the complex, host and ligand at a fully quantum-mechanical level.

Fig. 9.1: Catechol bound in the buried binding site of the T4 lysozyme L99A/M102Q double mutant. Visualization in VMD.

9.2 Setting up the calculations

We will set up three separate calculations, one each for the protein-ligand complex, the protein (host) and catechol (ligand). The structure of the complex was taken from a molecular dynamics simulation of the complex used in two QM-PBSA studies on this system [Fox2014] [Gundelach2021]. The structure of the unbound ligand and host were obtained from the complex by deletion of the respective molecules. Apart from the atomic coordinates, we must specify the details of the ONETEP single-point calculations, provide pseudopotentials for the atoms present in the system and adapt job submission scripts to run the calculations on the supercomputer of choice.

9.2.1 The input files

The ONETEP input file, referred to as the `.dat` file, contains two main elements: 1) the coordinates and atom types of the system (i.e the structural information) and 2) the details of the calculation. Due to the large system size, we have split theses two components across separate files: the `.dat` file, which contains the structural information, and a `.header` file which contains instructions for ONETEP. This header file is included in the `.dat` file via the command `includefile`. All information could also be contained in a single `.dat` file; however, the use of a separate header file can make it easier to set up hundreds or even thousands of calculations which differ only in the coordinates and not the calculation settings.

`.dat` file

The two blocks included in the `.dat` file are `lattice_cart` and `positions_abs`, which specify the simulation cell and absolute positional coordinates of each atom within the simulation cell, respectively. The `includefile` command on the first line specifies the header file to include for the calculation.

`.header` file

This `.header` file contains all further details of the ONETEP calculation. The `species` block specifies the name, element, atomic number, number of NGWFs and the NGWF radius for each atom type in the system. The `species_pot` lists the names of the pseudopotential files for each atom type. The rest of the file consists of ONETEP keywords which control the details of the calculation. The provided header files are fully commented, and details on each keyword are given in the ONETEP keyword directory (http://onetepkeywords.icedb.info/onetepdoc). We will be performing single-point energy calculations using the PBE exchange-correlation functional, the D2 dispersion correction and ONETEP's minimal paramater implicit solvent model. The calculation will output verbose detail and an `.xyz` file for easy visualization. The total system charge is +9 for the complex and host and 0 of the ligand. The implicit solvent is set to use the default parameters for water.

9.2.2 Submission Scripts

Due to the large system size of over 2500 atoms, these single-point calculations can only be run on a supercomputer. Thus, a submission script appropriate for the HPC environment you are working on will be necessary. The standard distribution of ONETEP provides sample submission scripts for a variety of HPC systems. These can be found in your ONETEP directory under `hpc_resources`.

We recommend to run the complex and host calculations on multiple compute nodes, making full use of the hybrid MPI-OMP capabilities of ONETEP. On the national supercomputer ARCHER2, the use of 4 compute nodes (128 cores each) with 32 MPI processes and 16 OMP threads per process results in a wall-time of about 8 hours. Due to the much smaller size of the ligand, the calculation on the ligand in solvent should be limited to a single node, with at most 10 MPI processes.

9.3 Evaluating the Outputs

Upon successful completion of the calculations, we will examine the three `.out` files created. Each of these files contains the full details and log of the calculation, as well as the final results and some timing information. While much information about the system can be gained from the output files, we will focus first only on the final results necessary to estimate the binding free energy of the ligand, catechol, to the protein.

Table 9.1: Calculated binding free energy of catechol to the protein.

Kcal/mol	Complex	Host	Ligand	Complex-Host-Ligand
E	-7372184.3	-7328209.2	-43940.1	-35.0
ΔG_{solv}	-2615.0	-2613.3	-9.7	8.0
G	-7374799.3	-7330822.5	-43949.7	-27.1

As outlined in equations (9.1) and (9.2) we need to calculate the total free energy of the complex, host and ligand before subtracting the total energy of the host and ligand from that of the complex. As stated before, we will be ignoring any entropy contributions in this tutorial. The total energy is then the sum of the total gas phase energy and the solvation free energy. These energies are summarized in an easy to read section at the very end of the output files, just before the timing information. To find it, search the output file for `Total energy in solvent`. This section breaks down the different energy contributions and states the total energies in vacuum (gas phase) and in solvent as well as the solvation free energy. Table 9.1 summarizes the energy values obtained. To estimate the binding free energy we simply apply equation (9.1) to yield:

$$\Delta G_{\text{bind}} = G_{\text{complex}} - G_{\text{host}} - G_{\text{ligand}} = \\ = -7374799.3 - (-7330822.5) - (-43949.7) = -27.1\text{kcal/mol}$$

Thus, the estimated binding energy of catechol to the T4 lysozyme is -27.1 kcal/mol. However, there are a number of severe limitations of this estimate: 1) the entropy correction term $-TS$ has been neglected; 2) only a single snapshot was evaluated; 3) the implicit solvent model incorrectly interprets the buried cavity in the T4 lysozyme, and 4) the QM-PBSA method is designed to

calculate relative binding free energies between similar sets of ligands. For an in depth look at the full application of the QM-PBSA binding free energy method to 7 ligands binding to the T4 lysozyme and a discussion of the errors, convergence and limitations of the method, please consult our recent publication [Gundelach2021].

9.3.1 Cavity Correction

The minimal-parameter PBSA solvent-model implemented in ONETEP incorrectly handles the buried cavity in the T4 lysozyme (L99A/M102Q). This is a known issue for solvent models based on the solvent accessible surface area, and has been described in detail in 2010 by Genheden *et al.* [Genheden2010], and in 2014 by Fox *et al.* [Fox2014].

In the un-complexed protein calculation, i.e the host, the surface area of the interior of the buried binding site is counted towards the solvent accessible surface area (SASA) used to calculate the non-polar solvation term. Thus, the non-polar term of just the protein is larger than that of the complex indicating the formation of a larger cavity in the solvent. Conceptually, the SASA model creates an additional, fictitious, cavity in the solvent with the SASA of the buried binding site. Because the non-polar term of both the protein and complex are known, a post-hoc cavity-correction may be applied to remove the additional (spurious) contribution of the buried cavity to the non-polar solvation energy. A full derivation is provided in [Fox2014].

$$E_{\text{cav-cor}} = 7.116(E^{\text{host}}_{\text{non-polar}} - E^{\text{complex}}_{\text{non-polar}}) = 7.116(289.5 - 286.2) = 23.5 \text{ kcal/mol}.$$

Applying the cavity correction term calculated above to the binding free energy, we obtain a cavity-corrected binding free energy of $-27.1 + 23.5 = -3.6$ kcal/mol. For comparison, the experimental binding energy of catechol to the T4 lysozyme is -4.4 kcal/mol. It should however be noted, that the close correspondence of this single snaphot QM-PBSA binding free energy to the absolute experimental energy is likely a lucky coincidence, as the QM-PBSA method is mainly applicable to relative binding free energies and the entropy correction term has not yet been included.

9.4 Properties

We will now show how a number of useful properties of the system can be studied through a *properties calculation*. In the interest of saving computational time, and for clarity of presentation, we will use the ligand system as an example.

Add the following keywords to the `.header` file of the ligand calculation:

```
do_properties T
dx_format T
cube_format F
```

and run it again.

The first of these keywords instructs ONETEP to perform a *properties calculation* towards the end of the run. This will calculate, among others, Mulliken charges on the atoms, bond lengths, the HOMO-LUMO gap, the density of states (DOS) and some grid-based quantities, such as the

HOMO and LUMO canonical molecular orbitals, electronic charge density and potential. The grid-based quantities (often called *scalarfields*) can be output in three different formats: `.cube`, `.dx`, and `.grd`. By default `.cube` files are written, and not the other two formats. In this example we switch off `.cube` output and turn on `.dx` output. This is effected by the last two keywords.

Once your calculation finishes, you will see that quite a number of `.dx` files have been produced:

- `_HOMO.dx` – density of the canonical HOMO orbital.
- `_LUMO.dx` – density of the canonical LUMO orbital.
- `_HOMO-`n`.dx` – density of the n-th canonical orbital below HOMO.
- `_LUMO+`n`.dx` – density of the n-th canonical orbital above LUMO.
- `density.dx` – the electronic density of the entire system.
- `potential.dx` – the total potential (ionic + Hartree + XC) in the system.
- `electrostatic_potential.dx` – the electrostatic potential (ionic + Hartree) in the system.

These files correspond to the calculation in solvent. There will be a second set of `.dx` files with `vacuum` in their names – these correspond to the calculation in vacuum. This lets you study and visualize in-vacuum and in-solvent properties separately and to perform comparisons between the two. Here, you can expect the scalarfields to be rather similar between in-vacuum and in-solvent because the ligand is charge-neutral and polarizes the solvent only very slightly.

There is a separate tutorial (Tutorial 5) devoted to visualization. You can use the skills taught there to create fancy visualizations of the properties of your choice. Here we will only show how to produce a neat visualization of the electronic density coloured by the electrostatic potential using VMD.

Load the electronic density and the electrostatic potential into one molecule, and the atomic coordinates into a separate molecule. This will make it easier treat the scalarfields and the atomic coordinates separately. To achieve this, issue:

```
vmd ligand_2001_density.dx ligand_2001_electrostatic_potential.dx -
↪m ligand_2001.xyz
```

Once VMD loads the files, go to `Graphics/Representations`. Ensure `Selected Molecule` (at the top of the window) is the `.xyz` file (atomic coordinates). Under `Drawing Method` Choose `CPK` – this will create a ball-and-stick drawing of the ligand. Switch `Selected Molecule` to the `.density.dx` file to operate on the electronic density scalarfield. Under `Drawing Method` choose `Isosurface` if it is not chosen already. Choose an `Isovalue` of `0.1` to pick a reasonable density isovalue to plot. Under `Coloring Method` choose `Volume` (you might need to scroll to the very bottom to get there). In the tiny drop-down window to the right of `Coloring Method` switch from scalarfield 0 (the density itself) to scalarfield 1 (the potential) – this will colour the density *with* the potential. For `Material` (further to the right) choose `Glass2` – this will choose a somewhat translucent material that will let us see both the ball-and-stick model and the electronic density. Under `Draw` in the bottom-right of the window, choose `Solid Surface` instead of `Points`. Finally, let's change the range of the potential to the kinds of values that occur at the distance from the molecule at which our electronic

density isosurface lies. These have been determined by trial and error. There are four tabs just above `Coloring Method`. Somewhat counterintuitively, switch to `Trajectory`, where, under `Color Scale Data Range` you can enter the minimum and maximum values for the potential (in eV). Enter `-1` in the left field and `1.5` in the right field and click `Set`. This should give a nice representation, which you can then rotate and translate to your liking using the mouse in the `OpenGL Display` window. Once you are satisfied, you can render the final image by going to `File/Render`. In the top drop-down menu choose `Tachyon` and click on `Start Rendering`. After a short while you will get a `.tga` ("TARGA format") file in the directory you are working in. It will look more or less like the graphics in Fig. 9.2. Most graphics manipulation programs and graphics viewers read `.tga` files. If you have `ImageMagick` installed, you can use it to convert the image to a more common format. For example to get a `.png` file, you can:

```
convert vmdscene.dat.tga vmdscene.dat.png
```

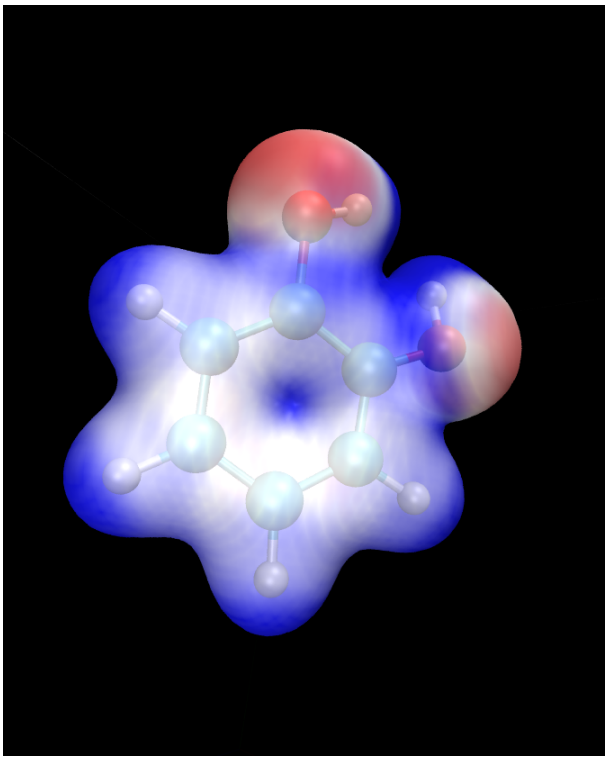


Fig. 9.2: Visualization of the ligand in VMD. A ball-and-stick model of the molecule is shown, together with an isosurface of the electronic density, coloured by the electrostatic potential.

9.4.1 Atomic charges

Mulliken population analysis

By default, during a properties calculation, ONETEP performs Mulliken population analysis, calculating partial atomic charges. The charges are written to the output file, in a table that looks like this:

```
    Mulliken Atomic Populations
    ---------------------------
```

(continues on next page)

(continued from previous page)

```
Species  Ion     Total    Charge (e)
==================================
  O       1       6.750     -0.750
  H       2       0.448      0.552
  C       3       3.817      0.183
...
==================================
```

The partial charges (in the electrons-are-negative sign convention) are output in the last column.

Mulliken population analysis has a number of drawbacks, chief among which is that it depends on the basis set used and there is no well-defined complete basis set limit. Below we discuss two alternative schemes that can be used in ONETEP: Natural Population Analysis (NPA) and Density-Derived Electrostatic and Chemical (DDEC) analysis.

Natural Population Analysis

In Natural Population Analysis the set of non-orthogonal, optimized NGWFs is transformed into a set of orthogonal atom-centered Natural Atomic Orbitals (NAOs). This approach lets empty, highly-diffuse orbitals distort to achieve orthogonality with their more highly-preserved occupied counterparts, ensuring the final NAO population is stable with respect to basis set size. More details, and references to papers on the method, can be found in the documentation for this functionality – chapter "Population Analysis" in the main ONETEP documentation.

To perform Natural Population Analysis *in lieu* of Mulliken population analysis, add the following keyword to your previous ligand calculation:

```
``write_nbo T``
```

and run it again. Keep the three keywords you added last time. Once your calculation completes you will find the results of NPA in your output file. They will look like this:

```
=============================================
               Natural Population
---------------------------------------------
 Summary
---------------------------------------------
   Atom         Population (e)      Charge (e)
---------------------------------------------
 O         1          6.7313861      -0.7313861
 H         2          0.4487370       0.5512630
 C         3          3.7852506       0.2147494
...
---------------------------------------------
```

Density-Derived Electrostatic and Chemical (DDEC) analysis

ONETEP uses the DDEC3 method [Manz2012] to effect atoms-in-molecule electron density partitioning, producing partial charges, as well as higher multipoles (if desired), which are both chemically meaningful and give a faithful reproduction of the electrostatic potential of the QM system. More details, and references to papers on the method, can be found in the documentation at www.onetep.org/pmwiki/uploads/Main/Documentation/ddec.pdf.

To perform DDEC analysis *in lieu* of Mulliken population analysis, add the following keyword to your previous ligand calculation:

```
ddec_calculate T
```

You will also need to add a `block ddec_rcomp` that will specify where the reference ion densities can be found. You will need *two* reference density files for every atomic species in your system – one for the core and one for the total density, except for H and He which only require the total density file. The reference density files for a number of often-found elements can be found in the `c2_refdens` directory of your ONETEP installation. Fortunately all the files necessary for our ligand calculation (so, reference densities for C, H and O) are already there. Add the following block to your ligand input file:

```
%block ddec_rcomp
H ALL "H_c2.refconf"
O ALL "O_c2.refconf"
O CORE "O_c2.coreconf"
C ALL "C_c2.refconf"
C CORE "C_c2.coreconf"
%endblock ddec_rcomp
```

and copy the five files listed in the block from the `c2_refdens` directory to where your calculation resides. The documentation explains where you can find reference density files for other elements, should you ever need them.

Once you re-run your ligand calculation, you will find the results of DDEC analysis towards the end of your output file. They will look like this:

```
-----------------------------------------------
             DDEC Charges (X=0.21)
-----------------------------------------------
   Atom         Population (e)      Charge (e)
-----------------------------------------------
 O        1          8.5534066      -0.5534066
 H        2          0.5775414       0.4224586
 C        3          5.8305022       0.1694978
...
-----------------------------------------------
```

Comparison of Mulliken, NPA and DDEC charges

The three approaches for calculating partial charges are compared in .:numref:*list-table T8_charges*. Mulliken charges are, in general, the most pronounced out of the three, while DDEC partial charges are overall smaller in absolute value. The predictions of NPA are rather close to Mulliken analysis, while DDEC differs more from the first two.

Table 9.2: Comparison of three approaches for calculating partial charges for the ligand.

Atom number	Species	Mulliken Charge	NPA Charge	DDEC Charge
1	O	-0.750	-0.731	-0.553
2	H	0.552	0.551	0.422
3	C	0.183	0.215	0.169
4	C	-0.319	-0.301	-0.229
5	H	0.311	0.251	0.160
6	C	-0.320	-0.261	-0.158
7	H	0.295	0.237	0.130
8	C	-0.313	-0.252	-0.124
9	H	0.298	0.241	0.131
10	C	-0.309	-0.300	-0.243
11	H	0.296	0.240	0.146
12	C	0.230	0.246	0.216
13	O	-0.711	-0.685	-0.510
14	H	0.557	0.549	0.444

But… tables are boring. How can we visualize the charges using VMD? This is not as straightforward as we would like. The structure (atomic coordinates) is contained in the `.xyz` file, but the charges are not. Some programs can visualize a quantity added in an extra column in the `.xyz` file (which would become something like an `.xyzq` file), but not VMD, at least not easily.

Fortunately VMD can read a different format named `.vtf`, which contains both the atomic coordinates and some scalar quantity, like charge. It is easy to convert an `.xyz` file and a list of charges to a `.vtf` file. We provide a simple `bash` script with this tutorial that does exactly that. It scans a ONETEP `.out` file for charge information (be it Mulliken, NPA or DDEC charges) and extracts the values of the charges on all atoms. It then looks for a corresponding `.xyz` file and, if found, it produces a `.vtf` file ready for visualizing with VMD.

To use it, download the provided script called `out2charge`, put it in your `$PATH`, and run it on your output:

```
out2charge ligand_2001.out
```

If everything goes well, you should see the following output:

```
Charges were output to ligand_2001.charge.
The files ligand_2001.xyz and ligand_2001.charge will be used
to construct ligand_2001.vtf.
```

(continues on next page)

(continued from previous page)

```
Load ligand_2001.vtf into VMD and select 'Coloring method -> charge
↪'.
```

Indeed, a new file `ligand_2001.charge` will be produced, containing the charges extracted from the `.out` file. These charges, together with the information in the `.xyz` file will be used to construct a `.vtf` file readable by VMD. Load this file into VMD:

```
vmd ligand_2001.vtf
```

and go to `Graphics/Representation`. For `Drawing Method` choose `CPK` and for `Coloring Method` choose `Charge`. You will get a nice ball-and-stick model of your ligand, with the atoms coloured accorind to charge. In Fig. 9.3 we show a comparison of the plots for the three ways of partitioning charge that we described earlier.

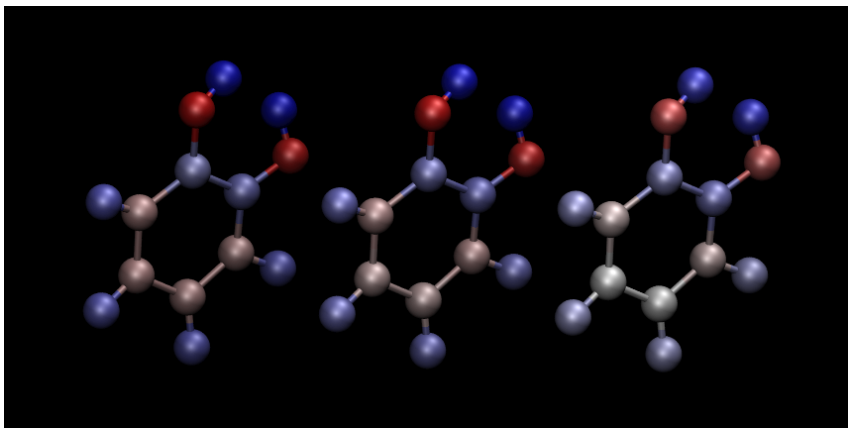


Fig. 9.3: Comparison of atomic charges on the ligand: Mulliken (left), NPA (middle) and DDEC (right). Warm colours correspond to negative charges. Visualization in VMD.

This completes tutorial 8.

Files for this tutorial:

- `out2charge`
- `T8_files.zip`

9.4.2 References

CHAPTER

TEN

TUTORIAL 9: DFT+U ON STRONGLY CORRELATED MAGNETIC MATERIALS: A CASE STUDY ON ANTIFERROMAGNETIC HEMATITE

Author
Davide Sarpa

Date
July/Aug 2023 (revised June 2024)

10.1 Introduction

The goal of the tutorial is to provide a working example on how it is possible to model strongly correlated magnetic materials applying the DFT+U method [Anisimov1991] [Anisimov1997] [Dudarev1998]. We will be working with hematite as an example of such materials.

10.1.1 Hematite

Hematite is a blood-red iron oxide with formula α-Fe_2O_3 with a melting point of $1350°C$. It belongs to the hexagonal crystal family, in particular, it is a ditrigonal scalenohedral with a $R\bar{3}C$ space group (167), sharing the same structure as corundum. The lattice parameters a, b, c are $a = b = 5.0356$ Å and $c = 13.7489$ Å with 6 formula units per cell with a band gap of $1.9 - 2.2$ eV. Its structure is an hpc anion stacking of O^{2-} along the [001] direction with Fe^{3+} occupying $2/3$ of the interstitial octahedrical positions [Cornell2003]. Below the Néel temperature ($T_N = 963K$), Fe_2O_3 is antiferromagnetic with weak ferromagnetism. The high-spin d^5 Fe^{3+} cations within one bilayer in the (0001) planes are ferromagnetically coupled to each other while antiferromagnetically coupled to the adjacent Fe bilayers [Parkinson2016]. The magnetic moment is determined to be $4.6\ \mu_B$ per atom. The top of the valence band is dominated by oxygen p states,while the bottom of the conduction band is dominated by Fe d minority spin states. Hematite is generally considered to be a charge transfer insulator rather than a Mott-Hubbard insulator [Naveas2023] [Huang2016] [Si2020].

10.1.2 Magnetism

The weak ferromagnetism is due to spin-canting which is a relativistic effect. Luckily for us it is possible to obtain hematite in an antiferromagnetic state with magnetic moments close to the experimental values by properly setting up the calculation without the need to include relativistic effects explicitly.

If we consider the primitive hematite cell along the Z axis, there are 3 possible different antiferromagnetic states

- $+-+-$ (up-down-up-down),
- $++--$ (up-up-down-down),
- $+--+$ (up-down-down-up).

Out of all these states the last one ($+--+$) is the ground state, we would like to be able to force our system to end up in the ground state.

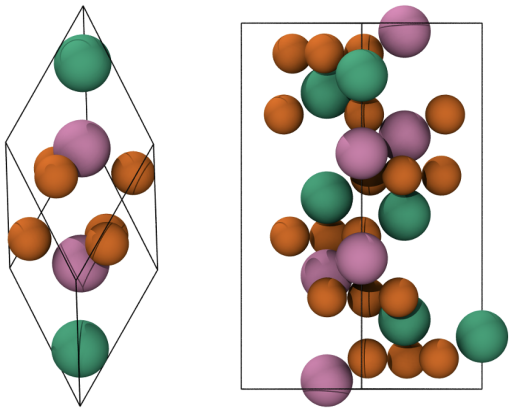


Fig. 10.1: Primitive rhombohedral cell (left), conventional hexagonal cell (right). Fe atoms with spin up and down are in green and pink, respectively. O atoms are in orange.

Magnetic systems are challenging to model due to the existence of very many different local minima which, in most cases, are very close in energy. Forcing the system into a specific state may not be easy, but there are methods that can help us achieve what we want.

10.1.3 DFT+U

The problem of DFT to describe correlated systems can be attributed to the tendency of xc functionals to over-delocalize valence electrons and to over-stabilize metallic ground states, this prevents materials like hematite from being described by DFT. LDA and GGA both predict hematite to be a metallic system, they also underestimate local magnetic moments. The reason behind this delocalisation is rooted in the inability of the approximated xc to completely cancel out the electronic self-interaction contained in the Hartree term. The main advantage of the DFT+U method is that it is within the realm of DFT, thus it does not require significant effort to be implemented in existing DFT codes and its computational cost is only slightly higher than that of normal DFT computations.

The DFT+U method selectively localizes specific orbital sets, typically d and f orbitals, while maintaining the delocalized nature of other orbitals at the LDA/GGA level. This is achieved by projecting the electronic bands onto a localized basis and calculating a modified potential. The DFT+U method can be used to penalise the non-integer occupancy of these orbitals, tending to fill states with occupancy greater than 0.5 and to empty states with occupancy less than 0.5.

$$\hat{V}^{(\sigma)}_{DFT+U} = \sum_I U^{(I)} |\varphi^{(I)}_m\rangle (\frac{1}{2}\delta_{mm'} - n^{(I)(\sigma)}_{mm'}) \langle\varphi^{(I)}_{m'}|.$$

The U and J values are screened Coloumb and exchange terms, which are system and implementation-dependent. In general, you are not able to plug and play a U or J value from the literature. What is usually done is empirically testing different values (run multiple calculations with different combination of U and J), or most software (including ONETEP) have a linear response theory implementation to calculate the parameters from first principles [O-Regan2010] [O-Regan2012] [Cococcioni2005].

10.2 Setting up the calculations

We will configure a bulk hematite calculation implementing a DFT+U correction specifically for the Fe $3d$ orbitals. We apply distinct labels to Fe atoms, enabling us to assign different parameters to spin-up and spin-down Fe atoms. This labeling strategy facilitates the achievement of the desired antiferromagnetic (AFM) state in hematite. You will see that the cell and atoms we are using are neither from a primitive or a conventional cell, It is a 4x4x1 supercell generated from the conventional cell. Such a big cell is necessary to accomodate NGWFs with a radius of 11 Bohr.

10.2.1 Tutorial files

ONETEP requires different files to work properly.

1. A `.dat` file which contains all the information about your sytem and the simulations parameters;
2. pseudopotentials files, we will be using on the fly generated by CASTEP, but you could use your favourites.

All the files needed for the simulation can be downloaded from

- `Fe_NCP19_PBE_OTF.usp`,
- `O_NCP19_PBE_OTF.usp`,
- `hematite.out`,
- `hematite.dat`.

Input File

The first two blocks are the cell and atomic positions. You might see that iron atoms are labelled `Fe1` or `Fe2`, depending on whether they will be treated as spin up atoms or spin down atoms.

The third block is:

```
%BLOCK SPECIES
  Fe1 Fe 26 13 11.000000
  Fe2 Fe 26 13 11.000000
  O O 8 4 11.000000
%ENDBLOCK SPECIES
```

The block defines the elements and enables the user to specify labels (such as `Fe1`, `Fe2` and `O`), atomic numbers, and the number of NGWFs to be used for each atom type inside the calculation. Additionally, it allows the user to set the radius for these NGWFs. For strongly correlated systems NGWFs radius of 11.0 Bohr or more is suggested. The next block is:

```
%BLOCK SPECIES_ATOMIC_SET
  Fe1 "SOLVE conf=3s2 3p6 3d5 4s0 4p0 SPIN=+5 CHARGE=0"
  Fe2 "SOLVE conf=3s2 3p6 3d5 4s0 4p0 SPIN=-5 CHARGE=0"
  O  "SOLVE INIT SPIN=0 CHARGE=-1"
%ENDBLOCK SPECIES_ATOMIC_SET
```

This block sets up the initial electronic configurations for the atoms. `Fe1` or `Fe2` atoms will have a spin of +5 or -5, respectively. The atomic solver generates the first guess for the density kernel for the first SCF iteration.

The next block is the Hubbard block where we set up the DFT+U parameters:

```
%block hubbard
  Fe1 2 6.0 0.0 -10.0 0.00 0.0
  Fe2 2 6.0 0.0 -10.0 0.00 0.0
%endblock hubbard
```

We assign a U value of 6 to the d orbitals ($l = 2$) in this block. For all other columns, we use default parameters.

The remaining blocks instruct ONETEP which atom types to use for calculating the local density of states (LDOS) and density of states (DOS). Two relevant parameters are also important:

1. `dos_smear`: Controls the Gaussian smearing applied to the DOS;
2. `pdos_max_l`: Specifies the maximum angular momentum quantum number (l) for computing the projected DOS.

Most other parameters are self-explanatory, with a few exceptions:

- `maxit_palser_mano`
- `maxit_hotelling`

These are associated with the diagonalization library and calculation of the inverse of the overlap matrix. For more detailed explanations of any parameters, consult the ONETEP keyword database.

Pseudopotentials

The number of NGWFs is determined by your choice of pseudopotentials. If you're using different pseudopotentials from the one provided here, make sure to adjust the number of NGWFs accordingly. Regarding the kinetic energy cutoff: The unusually high value is necessary due to the Fe

pseudopotential. This particular pseudopotential includes $3s$ and $3p$ semi-core states, which require a higher cutoff for accurate representation.

10.3 Evaluating the outputs

ONETEP will generate many files based on how we configured the calculations, but for this tutorial we will be focusing on only a few.

- `.out`: the main output file,
- `DOS.txt`: density of states file,
- `LDOS.txt`: local density of states file,
- `PDOS.txt`: projected density of states file,
- `spindensity.cube`: cube file necessary to visualise the spin density.

10.3.1 What to look for in the main output file

The first thing is to check is the whether the atoms are in the configuration we wanted (in our case a Fe^{3+} with spin up or down). This can be seen by looking at this block for each atom (shown here the down Fe atom)

```
Orbitals (num,spin,occ):  5  1     1.00 3.00 0.00 0.00 0.00
Orbitals   (num,spin,l):  5  1        0    1    2    0    1
Orbitals (num,spin,occ):  5  2     1.00 3.00 5.00 0.00 0.00
Orbitals   (num,spin,l):  5  2        0    1    2    0    1
```

The first number refers to the total number of orbitals ($3s$, $3p$, $3d$, $4s$, $4p$ as defined previosly), the spin channel either 1 or 2 and the orbital occupancies. In this case we have 1 spin up and 1 spin down electron in the $3s$ orbital, 3 up and 3 down electrons in the $3p$ orbitals and 5 spin down electrons in the $3d$ orbitals the $4s$ and $4p$ are empty.

The second step is, as explained in the DFT+U part, the occupancies for the majority spin channel (either up or down for different Fe atoms) has to be > 0.5 while < 0.5 for the minority spin channel. This is very important to allow DFT+U to do its job and it can be checked in the following table by looking at the diagonal elements.

```
########################################################################
 ↪############
DFT+U information on atom      1 of Hubbard species Fe1
########################################################################
 ↪############
Occupancy matrix of Hubbard site      1 and spin      1 is
  m_l =   -2          -1           0           1           2
  0.98760734  0.00754848 -0.00233330  0.00015001 -0.00147641
  0.00754493  0.99044110  0.00093484  0.00063070  0.00195361
 -0.00233979  0.00093793  0.99053553  0.00062471  0.00142290
```

(continues on next page)

(continued from previous page)

```
  0.00014994  0.00063069  0.00062302  0.99083622 -0.00700465
 -0.00147664  0.00195472  0.00141925 -0.00700844  0.98744366
########################################################################
 ↪############
Occupancy matrix of Hubbard site      1 and spin      2 is
  m_l =   -2          -1           0           1           2
  0.19734987 -0.07593555 -0.02935837 -0.01152995 -0.01749110
 -0.07589974  0.26431985  0.00033807  0.00686795 -0.01256107
 -0.02943958  0.00033830  0.10618329  0.00064404  0.01701648
 -0.01152456  0.00686813  0.00063868  0.25542523  0.07653629
 -0.01749366 -0.01256804  0.01696807  0.07657798  0.17892533
########################################################################
 ↪############
Total occupancy of Hubbard site      1 is          5.94906741 e
Local magnetic moment of Hubbard site      1 is    3.94466029 mu_B
DFT+U energy of Hubbard site      1 is             0.08933769 Ha
########################################################################
 ↪############
```

Another important thing to check are the bands' occupancies. Hematite is a semiconductor with a 2 eV band gap, we would then expect to have fully occupied bands and unoccupied virtual bands. If we were to treat it as a metal we could expect fractional occupancies occurring, but that would be physically wrong for our system.

If you look at the band occupancies for both spin up and down channel, you can see that we indeed obtain fully occupied bands and unoccupied bands. This reassure us that the structure we obtained is chemically and physically sensible.

```
                       Spin 1           |                  Spin 2    ␣
 ↪       |
 Orb |    H-eigenvalues     Occupancies |    H-eigenvalues    ␣
 ↪Occupancies |
   1 |    -2.7569116405    1.0000000000 |    -2.7569258300    1.
 ↪0000000000 |
                      ------                                ------
2396 |     0.5911355692    1.0000000000 |     0.5911388571    1.
 ↪0000000000 |
2397 |     0.5931137905    1.0000000000 |     0.5931136453    1.
 ↪0000000000 |
2398 |     0.5931148723    1.0000000000 |     0.5931148136    1.
 ↪0000000000 |
2399 |     0.5936028814    1.0000000000 |     0.5936016525    1.
 ↪0000000000 |
2400 |     0.5936039546    1.0000000000 |     0.5936026510    1.
 ↪0000000000 |
           - Gap at zero temperature - |     - Gap at zero␣
 ↪temperature -
            Finite temp. Fermi level   |      Finite temp. Fermi␣
 ↪level
```

(continues on next page)

(continued from previous page)

```
2401 |     0.6272424125    0.0000000000 |     0.6272633138    0.
↪0000000000 |
2402 |     0.6297211476    0.0000000000 |     0.6297567335    0.
↪0000000000 |
2403 |     0.6297236475    0.0000000000 |     0.6297598360    0.
↪0000000000 |
2404 |     0.6302246277    0.0000000000 |     0.6302507711    0.
↪0000000000 |
2405 |     0.6302330454    0.0000000000 |     0.6302577875    0.
↪0000000000 |
                        ------                                ------
3648 |     1.1980169016    0.0000000000 |     1.1980204435    0.
↪0000000000 |
```

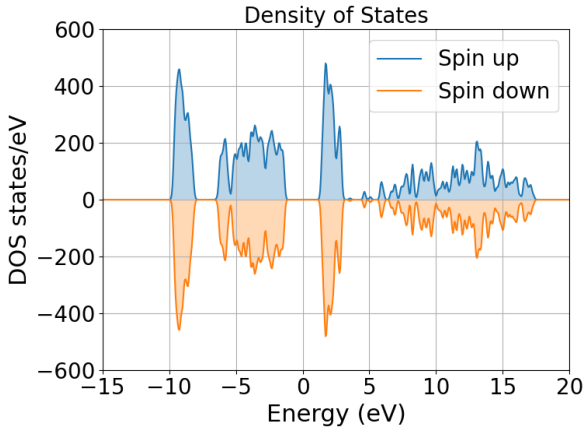
Lastly we should also check that we obtain a band gap and its value is close to experiment. This can be seen from the output by looking for these lines.

Why do we get two band gaps? Because we are studying a magnetic system, we get a band gap for each spin channel and for an AFM material the bandgap should be the same (numerical errors aside).

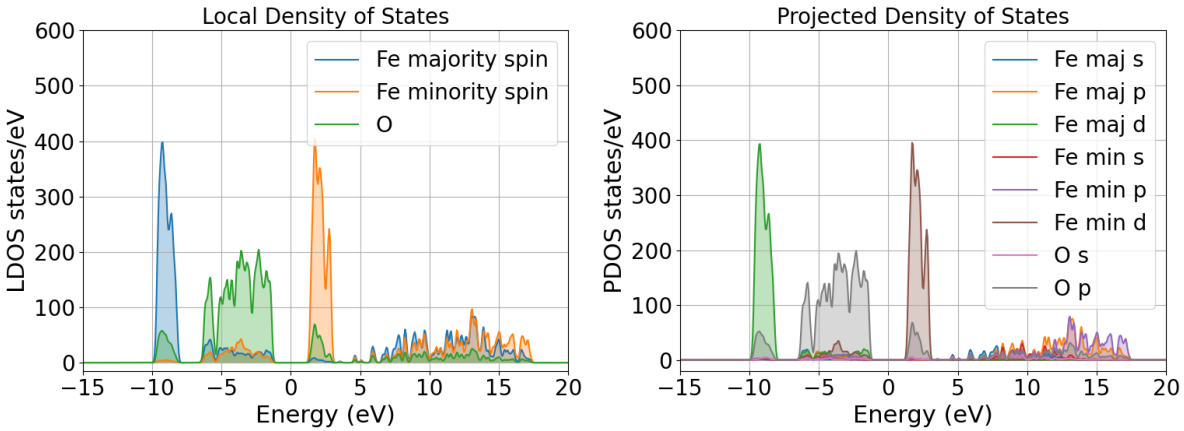
```
HOMO-LUMO gap:        0.101182637 Eh
HOMO-LUMO gap:        0.101174972 Eh
```

10.3.2 DOS and PDOS

Next step is to plot the density of states. It will tell us the distribution of electrons and states in our system



We indeed obtain a gap between the states but it does not tell us much more. To obtain more information we will be plotting the local density of states (LDOS) and the projected density of states (PDOS).



From the local density of states we can immediately notice that the lowest lying bands in the plot are mostly comprised of Fe majority spin channel states but, this is very important, the top of the valence band is mostly composed of O p states. The bottom of the conduction band is composed of Fe minority spin states. This allow us to classify hematite as a charge transfer insulator between the O and the Fe atoms. What if we would like to know which atomic orbitals contribute the most to this charge transfer, we need to plot the PDOS.

This will project the bands into the atomic components, in this way, as you can see in the plot the top of the valence band is dominated by O $2p$ states while the bottom of the conduction band is dominated by Fe minority spin $3d$ states.

10.3.3 Mulliken population analysis

Mulliken population analysis is a very good tool to understand if our system is behaving correctly. In an AFM material the total spin should be 0 and the atomic spin should be the same for the same atoms. In this case we have two different types, the spin up and down Fe atoms. The absolute value of the atomic spin should be the same only with different sign.

The material is also charge neutral and we would expect that similar atoms should carry similar charges.

Species	Ion	Total	Charge (e)	Spin (hbar)
O	1	6.923	-0.923	0.00
O	2	6.923	-0.923	-0.00
O	3	6.922	-0.922	-0.00
O	4	6.922	-0.922	-0.00
O	5	6.922	-0.922	0.00
O	6	6.922	-0.922	0.00
Fe	7	14.617	1.383	2.21
Fe	8	14.616	1.384	2.21
Fe	9	14.617	1.383	-2.21
Fe	10	14.617	1.383	-2.21

As you can see from this snapshot, we do indeed obtain the same charge and same spin for all similar atoms as we would expect.

10.3.4 Spin density

Now it is time to visualise the spin density, which is the total electron density of electrons of one spin minus the total electron density of the electrons of the other spin. We would like to visualise it to know if we obtained the AFM state we wanted, the up-down-down-up configuration.

You can directly open and visualise the `.cube` file generated at the end of the calculation with VESTA, VMD or other softwares.

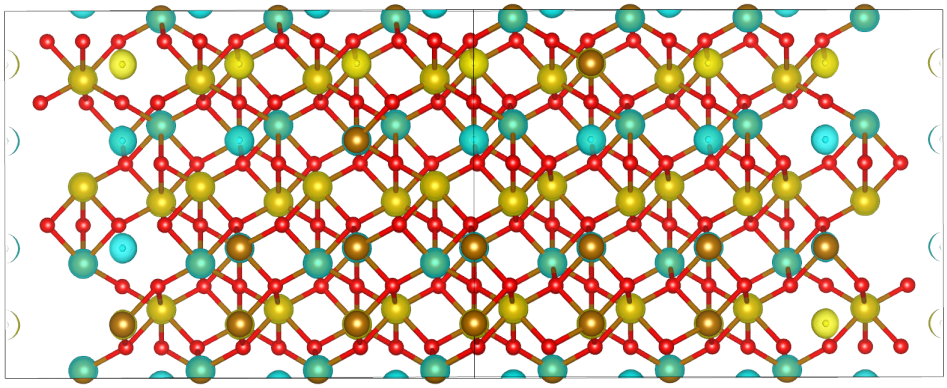


Fig. 10.2: Hematite spin density, blu spheres refers to atom with up spin and yellow to down spin

You can see from the picture that we did get the AFM states with $+--+$ configuration as we wanted.

10.3.5 What to do next

The tutorial is now complete, but you could still move forward. What can you do next? ONETEP outputs more information than what we covered so far. You can plot:

- the electrostatic potential,
- the orbitals,
- the electronic density.

You can then relax the structure and recompute the properties to see what changed and how. We have chosen to use $U = 6$, but you could try different U values and see how that affects the system.

CHAPTER

ELEVEN

TUTORIAL 10: SIMULATION CELL RELAXATION

Author
Chris-Kriton Skylaris

Date
August 2023

11.1 Introduction

This tutorial demonstrates how to use ONETEP to relax the simulation cell of a crystalline material

11.2 Cell relaxation of bulk crystalline silica

This calculation will relax the lattice of a silica (SiO_2) simulation cell, which is depicted below:

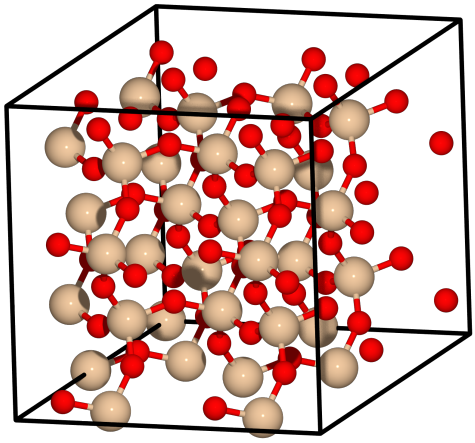


Fig. 11.1: The simulation cell of silica.

The simulation cell of silica used in this tutorial. The silicon atoms are colored beige and the oxygen atoms are colored red.

11.2.1 Input file keywords

The input file, which is provided, is called `silica96.dat` and contains 96 atoms in total.

To perform cell relaxation the

```
task STRESS
```

keyword is required.

You will notice in the input file also the following keywords related to the cell relaxation:

```
stress_tensor T
stress_elasticity F
stress_relax T
stress_assumed_symmetry tetra1
stress_relax_atoms T
```

Where `stress_tensor T` instructs the code to compute the stress tensor, while the calculation of elestic constants is trned off with `stress_elasticity F`. In this calculation the simulation cell will be relaxed (in order to determine the optimal lattice vectors) and this is denoted by `stress_relax T`. In this calculation, in addition to the simulation cell we want do relax also the coordinates of the atoms and for this we use the keyword `stress_relax_atoms T`. It is worth noting here that the atomic coordinates would also be relaxed if the `stress_relax_atoms` was set to `F` (False), but in this case they would only be "stretched" to be commensurate with the change in the lattice vectors, in other words they would retain the same fractional coordinates. On the other hand if the `stress_relax_atoms` is activated the coordinates of the atoms are fully relaxed and are not comstrained to remain equal to the fractional coordinates they had at the start of the calculation.

Finally the `stress_assumed_symmetry tetra1` instructs the code to assume a particular symmetry for the simulation cell and maintain this symmetry during the cell relaxation. Is the symmetry of the cell is known and is supported by the code (see the user manual) is is important to activate it with this keyword as it will significantly reduce the number of single point energy calculations that will be performed.

11.2.2 Running the calculation

Now run the calculation and examine the output. Lets examine the output step by step noting the various stages of the calculation.

At the very beginning some information about the initialisation of the calculation is produced such as:

- `PSINC grid sizes`: information about the grids used for the psinc basis functions

- `Atom SCF Calculation for...`: here the code initialises the NGWFs with atomic orbitals created specificaly for the valence electrons of the chosen pseudopotentials and confined within the NGWF spherical regions.
- `STRESS: undistorted cell`: this is the beginning of the very first energy calculation from which calculations with applied stains will be subtracted to compute the stress tensor.
- `Atomic positions optimised prior to stress calculation`: a geometry relaxation is performed first since we have specified `stress_relax_atoms T`.

You will notice that this calculation takes 30 NGWF iterations to converge as a very tight convergence criterion for the NGWFs (`ngwf_threshold_orig 1.e-7`) has been applied in the input. This was done to ensure very accurate forces and it may be a bit extreme, but it is better to be on the safe side.

After this energy evaluation the code computes the forces and compares them with the threshold that has been defined for geoemetry relaxation. In this calculation you will notice that the forces are small and below the tolerance (`|F|max 1.e-3 Eh/Bohr`) that has been set for the convergence of the geometry. As a result the code reports that the geoemetry relaxation has been completed with the message "Geometry optimization completed successfully".

11.2.3 Calculation of the stress tensor

Then the calculation proceeds to evaluate the energy of the system at different distortions (strains) of the lattice vectors in order to compute the stress tensor. The beginning of this procedure is denoted by the `STRESS: distorted cells` message.

The stress tensor computed is summarised at the end of the first iteration of cell relaxation:

```
stress_tensor:  iteration   1
```

This stress tensor is now used to change the simulation cell. Again a geometry relaxation is performed which converges at the first step. Then several single point energy calculations follow to compute a new stress tensor until we obtain the summary of the second iteration:

```
stress_tensor:  iteration   2
```

Finally, we see that in the third iteration the cell has been relaxed. The relaxed cell is printed:

```
Relaxed cell:
  bohr
19.07668106      0.00000000      0.00000000
 0.00000000     19.07668106      0.00000000
 0.00000000      0.00000000     27.83315074
```

This completes tutorial 10.

Files for this tutorial:

- `T10_files.zip`

CHAPTER

TWELVE

TUTORIAL 11: ELECTRIFIED ELECTRODE-ELECTROLYTE INTERFACES UNDER POTENTIOSTATIC CONTROL

Author
Brad Ayers, Arihant Bhandari

Date
August 2023

12.1 Introduction

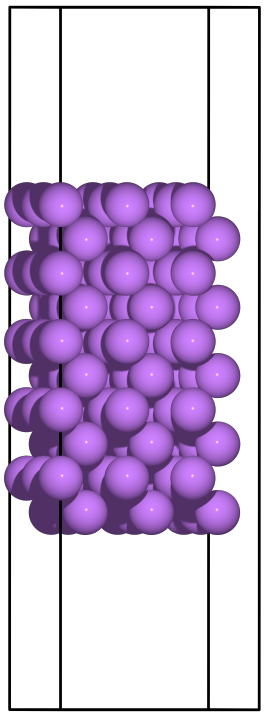
This tutorial endeavours to provide a comprehensive and illustrative example, highlighting ONETEP's ability to conduct potentiostatic calculations through the utilisation of a grand canonical DFT algorithm [Bhandari2021]. Alongside this written tutorial we have provided a Jupyter notebook that will guide you through the process of setting up and analysing a potentiostatic calculation.

Moreover, this tutorial will demonstrate the feasibility of conducting these calculations within a solvent and electrolyte medium, employing the Fisicaro soft-sphere continuum model [Fisicaro2017] for solvation and the neutralisation via the grand canonical electrolyte (NECS) model [Bhandari2020].

12.2 Setting up the calculations

We will begin this tutorial by creating lithium surfaces using a tool of your preference. For the purpose of illustration, we have employed the ASE [Larsen2017] to generate a 10-layer BCC [100] lithium surface. This can be simply visualised within the provided Jupyter notebook, and will look as follows:

Notably, in this tutorial, we've chosen a (3x3x10) supercell with a 50 Å vacuum in total, a decision that allows us to employ 9 Bohr NGWF radii, whilst ensuring ample volume for sufficient electrolyte to neutralise our surface.



12.2.1 Input files

For the purposes of this tutorial ONETEP will require only two files to work:

1. A .dat file, which contains all the information about your system (Atomic positions & Lattice vectors), as well your simulation parameters.
2. A Pseudopotential file, here we will employ the Norm-conserving on the fly generated CASTEP ones, but this is up to the users discretion.

Both of the aforementioned files, as well as a few output files can be downloaded below:

- `Keywords.dat`
- `Li_surface.out`
- `Li_surface.dat`
- `T11_workbook.ipynb`
- `Li_NCP19_PBE_OTF.usp`

Note that the output files required for the visualisation section of this tutorial will have to generated by the user themselves, and will need to be copied to the same directory as the Jupyter notebook.

Furthermore, Li_surface.dat is the .dat file used for your job submission, and Keywords.dat is simply the keywords that are taken in by the ASE script to generate said Li_surface.dat file.

12.2.2 The Dat file

grand canonical Parameters

Upon opening the provided Li.dat file, you will encounter the standardx parameters that ONETEP users are already familiar with: `Task`, `Cutoff_energy`, `Lattice_cart`, and `Positions_frac`.

However, after these, a new section of interest awaits, aptly named:

```
grand canonical Parameters
~~~~~~~~~~~~~~~~~~~~~~~~~

edft_grand_canonical T
edft_reference_potential -1.20 eV
edft_electrode_potential  0.20  V
```

Within the grand canonical formalism in ONETEP, there are two main parameters of interest:

1. **Chemical Potential of the Reference Electrode**

   Represented as μ_e^{ref}, this value (-1.20 eV in this example) corresponds to the reference electrode's chemical potential. It serves as the benchmark against which all applied potentials are measured.

2. **Applied Potential to the Working Electrode**

In this case, a potential of 0.20 volts (U) is applied to the working electrode, giving us a working electrode potential of -1.40 eV (μ_e).

These parameters find application in the following equations:

- **Chemical Potential of the Working Electrode** (μ_e):

$$\mu_e = \mu_e^{\text{ref}} - e \cdot U$$

- **Number of Electrons** (N_e):

$$N_e = \sum_i f_i = \sum_i \frac{1}{1 + \exp\left(\beta(\epsilon_i - \mu_e)\right)}$$

- **Charge on the System** (q):

$$q = Z_{\text{ion}} - e \cdot N_e$$

Hence, by defining the applied potential and reference electrode potential, we can establish the chemical potential of the working electrode. It is worth noting here that only with zero applied potential ($U = 0$) and the reference value set to the potential of zero charge ($\mu_e^{\text{ref}} = \mu_e^{\text{PZC}}$) will we get zero charge on the system ($q = 0$).

As a result, we gain the ability to compute the number of electrons within our system using the Fermi-Dirac distribution. Where, ϵ corresponds to the eigenvalues of the KS-equations, and μ_e is the chemical potential of our working electrode. Thereby allowing us to accurately calculate the charge present in the quantum system for a given applied potential.

For a more detailed explanation of the grand canonical formalism, please refer to [Bhandari2021].

Solvation Parameters

This section will highlight some of the more unintuitive parameters that are required to conduct a solvated calculation. They will be presented in the order in which they appear in the data file as done for the grand canonical parameters above.

This block will highlight the solvation parameters themselves, with the following block highlighting the solvent parameters

```
is_include_apolar T                 # Enables solvent-accessible␣
↪surface-area (SASA) approximation
is_smeared_ion_rep T                # Enables the smeared ion␣
↪representation due to multigrid solver
is_implicit_solvent T               # Enables the solvation model
is_solvation_properties T           # Provides extra properties i.e.
↪ electrolyte concentration
is_dielectric_function soft_sphere  # Utilising the Fisicaro soft-
↪sphere model
```

For a much more detailed overview of each key parameter and its function, please refer to the documentation here: Solvent and Electrolyte model

For the solvent itself we have the following parameters:

```
is_bulk_permittivity 90.7           # permitivity value of our␣
↪solvent        (ethylene carbonate in this case)
is_solvent_surf_tension 0.0506 N/m  # Surface tension of the␣
↪selected solvent (ethylene carbonate in this case)
```

Obviously these values will vary depending on the solvent of choice, and can be found in the literature.

Electrolyte Parameters

```
is_pbe                  full        # Enables the poison-boltzmann␣
↪solver for electrolyte calculations
is_pbe_bc_debye_screening T         # Enables the Debye screening␣
↪boundary condition
is_pbe_temperature   298.15         # Sets the temperature of the␣
↪Boltzmann ions
is_pbe_neutralisation_scheme counterions_auto
```

The above parameters are required to conduct electrolyte calculations, and are fairly self-explanatory. However, it is worth noting that there are several neutralisation schemes available, and the one chosen here is recommended for most cases.

Additionally, the following parameters are required by DL_MG to conduct the multigrid calculations:

```
mg_max_res_ratio 1000.0
mg_max_iters_vcycle 500
mg_max_iters_newton 300
mg_vcyc_smoother_iter_pre 4
mg_vcyc_smoother_iter_post 4
```

These parameters are fairly ubiquitous, and are not expected to change between calculations. For further information, please refer to the documentation here: DL_MG

lastly, we have the following ions block, that define our electrolytes and their concentrations:

```
%BLOCK SOL_IONS
Li    +1 1.0 # all concentrations are in Molar
PF6   -1 1.0
%ENDBLOCK SOL_IONS
```

Note that additional blocks can be added here to further adapt the model such as defining the soft-sphere radii of your system.

12.2.3 Analysis

Now that we have established the parameters required to conduct a solvated calculation, we can proceed to analyse the results obtained from either your own calculations or the provided ones.

The first file of interest is the `Li.out` file, which contains the usual ONETEP output you might be familiar with, but will include additional that are of interest to us. Firstly within CG-DFT inner loop, we can find the following:

```
Iter                                Commutator    Grand pot.(L=E-TS-
↪muN)  DeltaL
  # 2                          0.000001192422   -1269.
↪59513707094357  -2.05E-12

Step                        =       0.000010819110
Energy (E)                  =   -1297.517766665092
Entropy (-TS)               =      -0.065437957621
Chemical potential (-muN)   =      27.988067551769
Grand potential(L=E-TS-muN)=   -1269.595137070944
Est. 0K Energy 0.5*(E+L)    =   -1283.556451868018
Charge on quantum system    =      -3.995764707545
Residual Non-orthogonality =      -0.000000000000
```

If users are familiar with the canonical inner loop printing this will look familiar, however, there are a few key differences:

- **Grand Potential** ($L = E - TS - \mu_e N$):

  This is the grand potential of the system, and is the quantity that is minimised in the grand canonical formalism, rather than the Helmholtz free energy as in the canonical formalism.

- **Chemical Potential of the Working Electrode** ($-\mu_e N$):

  This is the chemical potential of the working electrode, and is the quantity that is constant in the grand canonical formalism, in direct contrast to the canonical formalism, where the number of electrons is constant.

- **Charge on the System** (q):

  This is the charge on the quantum system and is an important value to consider when analysing the results of a potentiostatic calculation, as surface chemistry is dependent on the charge of one's system.

Another crucial section of the `Li.out` file is the bulk concentration of the electrolyte species, which can be found in the following section:

```
+--------------------------- Chemical potential --------------------
↪------+
|   # | Name | Bulk conc. |    mu_ideal    |   mu_excess    |    mu_
↪total    |
|-------------------------------------------------------------------
↪------|
```

(continues on next page)

(continued from previous page)

```
|   1 |   Li |  2.1511516 |   0.000723249 |  -0.000320109 |   0.
↪000403139 |
|   2 |  PF6 |  0.7473276 |  -0.000274995 |  -0.000128144 |  -0.
↪000403139 |
|-------------------------------------------------------------------
↪------|

+----------------- Boltzmann ion concentration and charge ----------
↪-------+
|   # | Name | Bulk conc. | Average conc. | Neutr. ratio |    Total␣
↪charge |
|-------------------------------------------------------------------
↪-------|
|   1 |   Li |  2.1511516 |   2.151153746 |     0.820011 |     6.
↪122921628 |
|   2 |  PF6 |  0.7473276 |   0.747327832 |     0.179989 |    -2.
↪127151420 |
+-------------------------------------------------------------------
↪-------+
                                                      total:     3.
↪995770208*
```

This tells the user the bulk concentration of each species required to neutralise the surface. Note however as this is a grand canonical neutralisation we can have average concentrations greater than 1M. This is due to grand canonical neutralisation alowing for a changing number of species in the system, ensuring that the surface is neutralised at all times.

A final note here is as our calculations are performed within in solvent & electrolyte additional energy components are added to the energy breakdown printed at the end our calculations:

```
---------------- ENERGY COMPONENTS (Eh) ----------------
| Kinetic                      :      937.59573135429741 |
| Pseudo. (local,PBC,corr'd)   :     -443.84297910438869 |
| Pseudopotential (non-local)  :     -714.75069598392804 |
| Hartree (molecular)          :       41.93332431044017 |
| Exchange-correlation         :     -300.25087260634427 |
| Ewald                        :    20583.02565243620480 |
| Dispersion Correction        :       -0.65678636433385 |
| Smeared ion non-self corr.   :   -20592.71568368581570 |
| Smeared ion self corr.       :     -807.85811781290136 |
| Solvent cavitation energy    :        0.02783765502215 |
| Solute-solvent dis-rep en.   :       -0.02001318613680 |
| Elect. mobile ion energy     :       -0.00262591694826 |
| Osmotic pressure energy      :       -0.00778959415557 |
| Acc. corr. (steric) energy   :        0.00044786192622 |
| Ionic atmo. rearrang. en.    :        0.00641482442191 |
| Chemical pot. contribution   :       -0.00161085245160 |
| Total                        :    -1297.51776666508886 |
|-----------------------------------------------------|
```

(continues on next page)

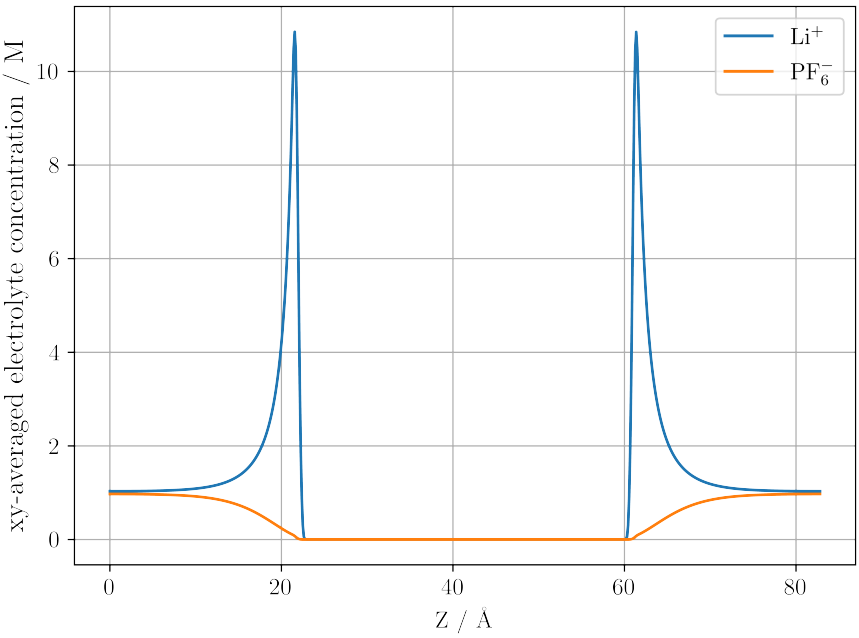
(continued from previous page)

```
| Entropic contribution      :       27.92262959414529 |
| Total free energy          :    -1269.59513707094357 |
-------------------------------------------------------
------ LOCAL ENERGY COMPONENTS FROM MATRIX TRACES ------
| Pseudopotential (local)    :     -443.84297910438886 |
| Hartree                    :      -24.92214858378443 |
-------------------------------------------------------
|Integrated density          :      543.99576470754505 |
|Integrated spin density     :        0.00000000000165 |
|Integrated |spin density|   :        0.00000001531685 |
|Local density approx. <S^2> :        0.00000000765765 |
|Integrated density tr(KS)   :      543.99576470754528 |
|Integrated spin tr(KS)      :        0.00000000000153 |
-------------------------------------------------------
```

We won't delve into the details of each energy component here, but it is worth noting that the solvent and electrolyte contributions are present in the energy breakdown. For a more detailed explanation of each energy component, please refer to [Bhandari2021].

visualisation

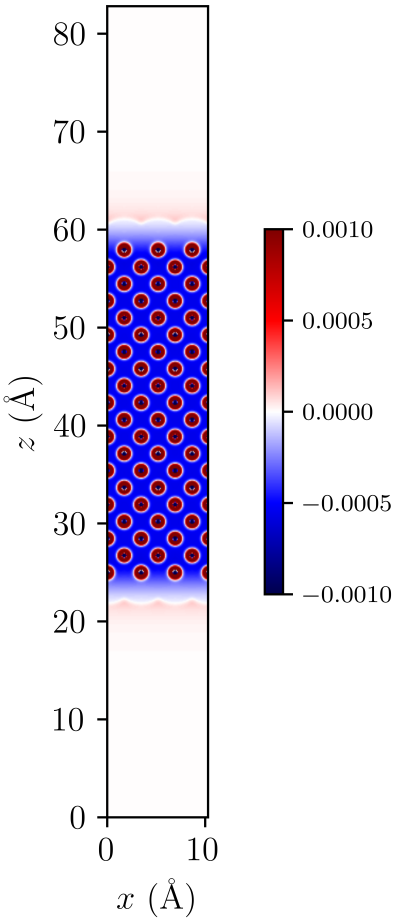
Whilst a majority of the analysis can be conducted using the `Li.out` file provided, there are a few visualisations that can be conducted using the user generated `Li_out_bion_conc_species_1.dx` and `Li_out_bion_conc_species_2.dx` files. Such as plotting the xy-averaged concentration profiles of the electrolyte species, it is worth stating that all code used to generate these plots can be found in the provided Jupyter notebook.



Looking at the figure above we can see that the concentration of the lithium electrolyte species is highest (12M) at the surface of the electrode, and decreases as we move away from the surface, forming a double layer along our surface. In contrary we can observe the PF6 electrolyte species

has a concentration approaching 0M at the surface, and increases as we move away from the surface, note that these are local concentrations and not bulk concentrations.

We can also visualise the densities of our lithium slab and the electrolyte species to illusrate this double layer effect by adding the `Li_out_bion_density_total.dx` and `Li_in_rho.dx` files together and running the provided code:



Here we can see that there is a positive charge denisty (our lithium ions) at the surface of the electrode, that diffuses in intensity as we move away from the surface.

12.3 Where to go next?

This tutorial has provided a brief overview of the grand canonical formalism, and how it can be employed to conduct potentiostatic calculations, as well as highlighting a few key visualisation techniques that can be conducted on the output files.

However, there are a few key areas that have not been covered in this tutorial that the reader should explore, such as:

1. **Defining their reference potentials** (μ_e^{ref}):

   In this tutorial, we have defined the reference potential as -1.20 eV, however, this value is dependent on the reference electrode of choice, and can be found in the literature or by calculating one's own reference potential using the method highlighted in [Bhandari2021].

2. **Cycling through potentials**:

   In this tutorial, we have only conducted a single potentiostatic calculation, however, it is possible to conduct a series of calculations at different potentials, and calculate properties such as capacitance etc.

12.3.1 References

CHAPTER

THIRTEEN

TUTORIAL 12: QUANTUM EMBEDDING WITH (TIME-DEPENDENT) EMBEDDED MEAN-FIELD THEORY: HYDROGENATION AND EXCITATIONS OF PENTACENE

Author
Joseph Prentice

Date
August 2023

13.1 Introduction

13.1.1 The utility of quantum embedding

Although ONETEP [ONETEP2020] makes first principles calculations of systems containing thousands of atoms feasible, particularly when using semi-local exchange-correlation functionals, it can still be very computationally costly to treat entire systems of this size with higher level theory, such as hybrid functionals, which require the computation of exact exchange. This is particularly relevant for excited state calculations, due to the well-known underestimation of the band gap by semi-local DFT, making excitation spectra performed with semi-local DFT quantitatively, or sometimes qualitatively wrong. However, if the physics/chemistry of interest in the system is known or expected to be localised to a particular subregion – we call this the *active region' – with the rest of the system acting as an environment influencing the active region, quantum embedding provides a way to achieve hybrid accuracy with a significantly reduced cost.*

This is achieved by treating the active region alone with the higher level of theory, with the rest of the system (the environment) treated at a lower level of theory. In ONETEP at present, this translates to treating the active region with hybrid DFT, and the environment with semi-local DFT. This is done within a single self-consistent calculation, to ensure that the two regions are able to influence one another. As hybrid DFT is only performed on the active region, which is typically small compared to the environment, the computational cost is significantly reduced.

13.1.2 Embedded mean-field theory

There are many different quantum embedding schemes; the scheme used in ONETEP is embedded mean-field theory (EMFT), originally proposed by Fornace et al. [Fornace2015] This scheme has several advantages, including two particularly relevant to ONETEP: firstly, it partitions the system at a basis-set level, which works well with ONETEP's atom-centred NGWF basis. Secondly, it is a mean-field theory throughout, like DFT, which means that many existing methods based on DFT can be easily modified to accommodate EMFT. This includes linear-response time-dependent DFT (LR-TDDFT), which allows electronic excitations to be computed – we refer to this as TD-EMFT. For more details, please see the original EMFT paper [Fornace2015], and the papers implementing ground state EMFT [Prentice2020] and TD-EMFT [Prentice2022] in ONETEP.

EMFT is also fully compatible with the implicit solvent methods available within ONETEP, allowing for multi-level modelling of systems.

13.1.3 Pentacene

In this tutorial, we will look at both ground state EMFT and TD-EMFT, with the pentacene molecule as our test case. Ground state EMFT is demonstrated by looking at the terminal hydrogenation energy of pentacene, following Prentice et al. [Prentice2020], and TD-EMFT is demonstrated by looking at the first excitation energy of pentacene-doped p-terphenyl, following Prentice. [Prentice2022]

13.2 Ground-state EMFT: terminal hydrogenation of pentacene

13.2.1 Non-EMFT baseline calculations

The terminal hydrogenation reaction for pentacene involves two hydrogen atoms becoming bonded to the two carbon atoms at one end of the pentacene molecule.

Before using EMFT, we must first obtain the terminal hydrogenation energy without EMFT, with everything treated with first the PBE, then the B3LYP functionals. The input files required are `Pentacene.dat`, `HydrogenatedPentacene.dat`, and `H2.dat`.

If you look at these files, you will notice that we have several different labels for carbon atoms (C1, C2, etc.) and similar for hydrogen atoms – these will be used later to vary the size of our active region, by selecting different atoms to be included within it.

To begin, simply run the input files as they are to obtain the ground state energy for the three structures at the PBE level. The hydrogenation energy can then be obtained as:

$$\Delta E_{\text{hyd}} = E_{\text{Hydrogenated Pentacene}} - (E_{\text{Pentacene}} + E_{\text{H2}}). \tag{13.1}$$

Make sure you save the `.tightbox_ngwfs` file from the `H2.dat` calculation for use later!

Now repeat this for B3LYP. In all three `.dat` files, change `xc_functional` to be `B3LYP`, and add the following keywords/blocks to set up Hartree-Fock exchange (deleting any species labels in the `species_swri...` block that are not present in that particular structure):

```
%block swri
  for_hfx 3 12 V 12 12 WE2
%endblock swri

%block species_swri-for_hfx
C
C1
C2
C3
H
H1
H2
H3
%endblock species_swri-for_hfx

hfx_use_ri for_hfx
hfx_max_l 3
hfx_max_q 12
```

For more details on the meaning of these keywords, see the HFx documentation. The most important point for our purposes here is that we can control which atoms are included in the computation of HFx through the `species_swri...` block – this will become important for our EMFT calculations.

For reasons of stability, it is best to start the B3LYP H2 calculation from the PBE-optimised NGWFs: bring back the PBE-optimised `.tightbox_ngwfs` file, and add `read_tightbox_ngwfs : T` to `H2.dat`. Make sure you still keep a copy of the PBE-optimised `.tightbox_ngwfs` file safe, as we will need it later.

Run these three calculations and compute the hydrogenation energy at the B3LYP level.

13.2.2 EMFT calculations

4 carbon atoms

We can now start to use EMFT to see if we can get close to the B3LYP result without treating the entire molecule with B3LYP. Initially, our active region will just include the 4 C atoms closest to the site of the reaction, and the hydrogen atoms bonded to them. We will only need to do EMFT calculations for `Pentacene.dat` and `HydrogenatedPentacene.dat` – the hydrogen molecule will always be in the active region, so should always be treated with B3LYP, although there is one subtlety which will be introduced shortly.

To turn on EMFT, change `xc_functional` back to `PBE`, and add the following keywords to `Pentacene.dat` and `HydrogenatedPentacene.dat` (keep the other modifications you already made for HFx for the moment):

```
use_emft             T
use_emft_follow      T
use_emft_lnv_only    T
block_orthogonalise  F
active_xc_functional B3LYP
parallel_scheme      HOUSE

%block species_ngwf_regions
C1 H1
C C2 C3 H H2 H3
%endblock species_ngwf_regions
```

A brief explanation of each keyword:

- `use_emft`: this turns on EMFT, so that the active region and environment are treated with different levels of theory.
- `use_emft_follow`: this toggles whether a non-EMFT calculation at the lower level of theory (in this case, PBE) is done first, and then uses that as a starting point for the EMFT calculation. For this to happen, the value should be `T`.
- `use_emft_lnv_only`: this toggles whether EMFT is used to optimise both the NGWFs and the density kernel (value is `F`), or just the density kernel (value is `T`). Typically, EMFT is only used to optimise the density kernel (`T`), as NGWF optimisation is poorly behaved under EMFT – NGWFs can unphysically optimise towards the region described by the level of theory that predicts the lowest energy, and the block orthogonalisation procedure designed to counteract this (discussed shortly) makes the NGWF optimisation stall. The NGWFs are optimised at the lower level of theory, and then fixed – the error this introduces is typically <1% of the difference between the high and low levels of theory. For more details, see Prentice et al. [Prentice2020]
- `block_orthogonalise`: this toggles whether a block orthogonalisation procedure is applied to the NGWFs before using EMFT. This transforms the NGWFs of the environment so that they are orthogonal to the NGWFs of the active region, so the off-diagonal blocks of the overlap matrix are 0. This prevents the emergence of unphysical solutions that can occur in some systems. For this to happen, the value should be `T`.
- `active_xc_functional`: this selects the functional that will be used in the active region, whilst `xc_functional` selects the functional used in the environment.
- `parallel_scheme`: this decides how MPI processes should be split between the regions. `HOUSE` means that the processes are distributed proportionally to the number of atoms within each region; `SENATE` means that the processes are distributed equally between all regions; and `NONE` means that each region will use all the processes in turn. `HOUSE` is strongly preferred.
- `block species_ngwf_regions`: this assigns species to regions, with one region per line. The first line is the active region by default.

The first four keywords should be turned to `T` in the order they are listed in. The first three keywords should almost always be `T` for an EMFT calculation, with `block_orthogonalise` turned on if the calculation proves unstable without it. Here, we leave it off.

We also need to modify the HFx set-up to match the fact we only want HFx in the active region. To do this, simply delete any species in the `species_swri...` block that are *not* in the active region. Remember that each species in the active region should be on its own line in the `species_swri...` block, whereas all the species in the active region should be on the first line of the `species_ngwf_regions` block.

Once these additions/modifications have been made, run the calculations for pentacene and hydrogenated pentacene.

Before we can compute the hydrogenation energy from these results, there is one more subtlety. As we optimised the active region NGWFs at the lower level of theory, but the active region density kernel with the higher level of theory, we need to do the same in our hydrogen molecule for consistency. To do this, bring back the `H2.tightbox_ngwfs` file you saved from the PBE calculation earlier, and re-run your B3LYP `H2.dat` calculation with the following modifications/additions:

```
read_tightbox_ngwfs : T
maxit_ngwf_cg : 0
```

You can now use these three results to compute the hydrogenation energy with an active region of this size.

Larger active regions

Next, expand the active region to include the 8 carbon atoms closest to the reaction site. To do this, add the C2 and H2 species to the active region (remember to remove them from the environment region!), and modify the HFx set-up to match. Rerun the pentacene and hydrogenated pentacene calculations, and compute the hydrogenation energy (you don't need to rerun the hydrogen molecule calculation, as you can just reuse the result obtained using PBE-optimised NGWFs).

Finally, expand the active region further to include the 12 carbon atoms closest to the reaction site, by adding C3 and H3 to the active region. Re-calculate the hydrogenation energy.

If you plot the hydrogenation energy vs. the size of the active region, you should see the hydrogenation energy approach the full B3LYP result. This demonstrates the ability of EMFT to obtain high level results at a reduced cost, even when the boundary between regions cuts through covalent bonds.

This also demonstrates the importance of selecting the appropriate active region. In systems made up of weakly bonded parts (e.g. molecular crystals, solvated systems), the appropriate active region will often be obvious – it will be the molecule or molecules of interest (examples of multiple-molecule active regions could include a nearest-neighbour dimer or a solute along with some nearest-neighbour solvent atoms). In extended covalent or ionic systems, the choice of active region may be more difficult, and should be carefully converged, in a similar way to that shown in this tutorial.

Further investigations

To further investigate the use of EMFT in ONETEP, you could look at the effects of:

- changing the active region further – perhaps including more C atoms, excluding H atoms, etc.,
- using block orthogonalisation,
- using other functionals for either the high or low level of theory – semi-local functionals can be used for the higher level although this is of course not expected to produce a significant advantage

… and many other possibilities.

13.3 TD-EMFT: excitations of pentacene-doped p-terphenyl

13.3.1 Non-EMFT benchmark

Here, we will be looking at the S0 to S1 transition in pentacene, which is the lowest excited state observed in TDDFT. This is significantly affected by the environment. In particular, we are interested in pentacene-doped p-terphenyl, which is of interest for room-temperature maser applications, and how the p-terphenyl environment affects the excitation energy.

To give us a reference for isolated pentacene, we first need to perform a high-level TDDFT calculation for pentacene. We will again use B3LYP as our high-level theory. The input file is `Pentacene_isolated.dat` – as this tutorial assumes you are already familiar with running TDDFT calculations with ONETEP, we will not go into any detail, and this calculation can just be run as it is.

13.3.2 TD-EMFT calculation

We now perform a TD-EMFT calculation for a pentacene molecule surrounded by 6 p-terphenyl molecules, as extracted from the pentacene-doped p-terphenyl molecular crystal. The input file is `Pentacene_in_p-terphenyl.dat`. This can be run as it is, but one point regarding EMFT should be noted first. The general set-up of the EMFT calculation is precisely the same as for ground state EMFT, with one addition: the `species_tddft_kernel` block. By using this block, we can specify which species we will restrict our excitations to be localised on. Given that in a TD-EMFT calculation we expect the excitations of interest to be localised within the active region, the species contained within the `species_tddft_kernel` block should be a subset of those in the active region. Typically, the two will be identical, i.e., the contents of the `species_tddft_kernel` block should be the same as the first line of the `species_ngwf_regions` block. For more details, see the LR-TDDFT documentation.

Run this calculation – this may take some time. If you compare this to the results in Prentice [Prentice2022], you can see that the result is very close to the experimental value of 2.09 eV.

You can plot the resulting excitation as a `.cube` file, and visualise it using e.g. VESTA.

13.4 Files for this tutorial

- `Pentacene.dat`
- `HydrogenatedPentacene.dat`
- `H2.dat`
- `Pentacene_isolated.dat`
- `Pentacene_in_p-terphenyl.dat`
- `C_NCP19_PBE_OTF.usp`
- `H_NCP19_PBE_OTF.usp`

13.4.1 References

CHAPTER

FOURTEEN

TUTORIAL 13: ELECTRON ENERGY LOSS SPECTROSCOPY IN ONETEP

Author
Edward Tait and Nicholas Hine

Date
August 2023 (original text c2018)

14.1 Introduction

EELS (Electron Energy Loss Spectroscopy) is a spectroscopic technique which combines high spatial resolution with fair energy resolution, and is thus sensitive to local electronic structure in a material. The theory behind EELS calculations in ONETEP is explained in the documentation page on EELS, as well as in a paper giving an overview of the capabilities[1], and in the thesis of one of the authors (Edward Tait)[2].

EELS calculations in ONETEP proceed by running first a singlepoint calculation (with or without a core hole, as required) and then running a properties calculation. The properties calculation will output files suitable for use with the OptaDoS package, after a little renaming and rearrangement which can be carried out with a script provided.

14.2 System setup

We will demonstrate the procedure for running an EELS calculation on a toy system: silene (the silicon equivalent of ethene). An example input file is provided in the Files section below, as are the PAW potentials for Silicon and Hydrogen and the associated core wavefunction data for Silicon. The tutorial files use the JTH pseudopotentials, with the addition of core orbitals, and we also regenerated the PAW potential and Core orbitals with a core hole, for the later part of this tutorial.

[1] N. D. M. Hine, Linear-Scaling Density Functional Theory using the Projector Augmented Wave Method, J. Phys. Condens. Matter 29, 024001 (2017). https://iopscience.iop.org/article/10.1088/0953-8984/29/2/024001

[2] Linear-Scaling Density Functional Theory and Theoretical Electron Energy Loss Spectroscopy Investigations of Surfaces and Defects in Nanomaterials, PhD Thesis of Edward Tait, 2019, University of Cambridge https://www.repository.cam.ac.uk/items/fcc71788-1468-47f7-989c-9a9d6e349f1e

There are a few differences between the ONETEP input here and one for a typical single point calculation:

- PAW is mandatory
- We specify a second species for the atom whose core electrons we are exciting
- Because a conduction calculation is being performed we must provide a `species_cond` block
- We must provide a `species_core_wf` block and every species must be listed there.

The input file can be run swiftly on a single node and should produce a large number of output files. Most of these files are `.cube` files of wavefunctions produced by default during the properties calculations. The files of interest are the `.elnes_bin` files, which contain OptaDoS compatible matrix elements.

A little more setup is needed before we can run OptaDoS (using the silene example):

- A dummy castep `silene-out.cell` file must be produced, and it must contain a symmetry block
- By default two `.elnes bin` files are produced, one based on Kohn-Sham wavefunctions represented using only the valence NGWFs (`silene_val_grad.elnes_bin`) and a second which makes use of the joint basis of valence and conduction NGWFs (`silene_joint_grad.elnes_bin`)
- As per the discussion above, you should choose the latter and copy it to `silene.elnes bin`.
- A `silene.bands` file must be produced, this is best done by copying `silene.joint bands` to `silene.bands`.
- An OptaDoS input file, `silene.odi` is needed.

To assist in these tasks a utility script, `prep_optados_eels`, is provided in the utils folder of the onetep distribution. Run it with the calculation seed name as its argument and the steps listed above will be completed automatically.

The .odi file produced should be regarded as a basic template, consult the OptaDoS documentation[6] if you wish to use more advanced features. Note that at the moment only fixed broadening is supported by onetep. When you are satisfied with your OptaDoS input file, execute OptaDoS with your calculation seed name as the argument. All being well, you should see a .dat file which you can plot with your favorite tool. Individual edges are listed sequentially in the file, so a little post processing with awk or python is needed to separate the edges for individual plotting

14.3 Files for this tutorial

- `Si2H4_EELS_Example.dat`
- `H.PBE-paw.abinit`
- `Si.PBE-paw.abinit`
- `Si.PBE-paw.corewf.abinit`
- `Si_corehole.PBE-paw.abinit`
- `Si_corehole.PBE-paw.corewf.abinit`

14.3.1 References

CHAPTER

FIFTEEN

TUTORIAL 14: DENSITY FUNCTIONAL TIGHT BINDING IN ONETEP

Author
Arihant Bhandari

Date
July 2024

15.1 Introduction

Density Functional Tight binding models have been derived by a Taylor expansion of the DFT energy functional in terms of the electron density and truncation up to a certain order in the expansion [Foulkes1989]. Within DFTB, the following eigenvalue equations are solved via diagonalization:

$$H_{\alpha\beta}M_i^\beta = S_{\alpha\beta}M_i^\beta\epsilon_i,$$

where $H_{\alpha\beta}$ is the hamiltonian matrix, $S_{\alpha\beta}$ is the overlap matrix, M_i^β are the orbital coefficients, and ϵ_i are energy eigenvalues. The Hamiltonian is built from parameters. [Elstner1998] proposed a self consistent charge (SCC) extension to the traditional DFTB approach, which optmizes the atomic charges self consistently. Henceforth, a series of SCC DFTB methods have been developed [Gaus2014]. Recently, non-SCC methods have undergone a revival because of their speed and applicability to large systems [Bannwarth2020]. E.g. GFN0 is one such method where atomic charges are found using a charge equilibriation scheme [Pracht2019]. The total energy in GFN0 also includes zeroth order terms such as dispersion, repulsion, electrostatic interactions and short range basis correction.

We have implemented the GFN0 method within ONETEP. Here we include D2 dispersion correction [Grimme2006] instead of D4 [Caldeweyher2019]. The standard ensemble-DFT subroutines are used for diagonalization and calculation of electronic energies and forces.

15.2 Keywords

Here are some basic keywords to perform a DFTB calculation.

- `dftb: T/F`:

  [Boolean, default `dftb: F`]. If true, it enables DFTB calculations.

- `dftb_method: GFN0`:

  [Text, default `dftb_method: GFN0`]. Variant of the DFTB method, only GFN0 has been implemented at the moment.

- `dftb_method_param_file: file address`:

  [Text, default `dftb_method_param_file: param_gfn0-xtb.txt`]. Path to the parameter file. A specimen file is supplied in utils-devel/dftb folder.

- `dftb_common_param_file: file address`:

  [Text, default `dftb_common_param_file: param_gfn_common.txt`]. Path to the file for common GFN parameters. A specimen file is supplied in utils-devel/dftb folder.

- `dftb_bc: O O O / P P P`:

  [Boolean, default `dftb_bc: P P P`]. Boundary conditions. Only fully open (O O O) or full periodic (P P P) are implemented.

15.3 Input files

In this tutorial, we will use DFTB to calculate the relaxed geometry of ethylene carbonate molecule (OBC), lithium tetra ethylene carbonate cluster (OBC), and perform molecular dynamics on a lithium graphite system (PBC). Please download the files below and run them using ONETEP.

- `ethylene_carbonate.dat`
- `liec4.dat`
- `li-graphite.dat`

The DFTB-GFN0 parameter files are available with utils-devel repository and also below:

- `param_gfn0-xtb.txt`
- `param_gfn_common.txt`

15.4 References

CHAPTER

SIXTEEN

TUTORIAL 15: FIRST PRINCIPLES CALCULATION OF U AND J

Author
Davide Sarpa

Date
Aug 2024

16.1 Introduction

The goal of the tutorial is to provide a working example on how it is possible to compute the U and J parameters from first principles. We will work on Hematite $+--+$ antiferromagnetic configuration as you should already be familiar with it, if not, refer to tutorial 9.

The reason behind computing the parameters via first principles is because they directly correct the spurious localised self-interaction error (U) and static correlation error (J) and hence the physics of the system. While choosing and empirical U and J might give a better description of a specific property of the material, it does not guarantee that these errors are consistently corrected.

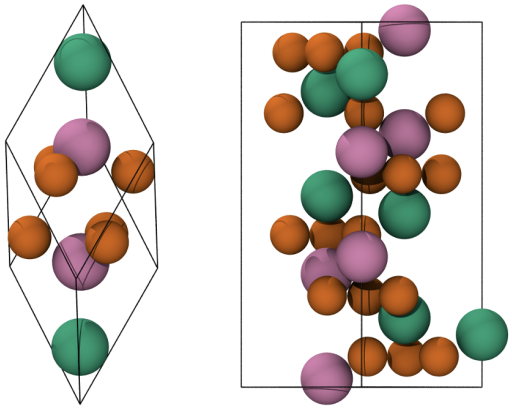


Fig. 16.1: Primitive rhombohedral cell (left), conventional hexagonal cell (right). Fe atoms with spin up and down are in green and pink, respectively. O atoms are in orange.

16.1.1 Theoretical background

We start by defining the response function χ, which describes how the occupation of localised orbitals changes with respect to a shift in the potential acting on these orbitals: The linear response method determines the Hubbard U parameter by comparing the response of the system to a perturbation in standard DFT and DFT+U frameworks.

We define the response function χ as:

$$\chi = \frac{dn^{I\sigma}}{d\alpha}$$

where n is the occupation matrix of the localised orbitals and α is a potential shift applied to these orbitals.

We compute two response functions:

- χ_0: the bare Kohn-Sham (KS) response (without U)
- χ: the interacting response (with U)

These are related by:

$$U = \chi^{-1} - \chi_0^{-1}$$

which allows us to compute U.

The perturbation is applied by shifting the potential of the localised orbitals:

$$V_{\text{ext}}^{p} = V_{\text{ext}} + \alpha \sum_{m,m'} |\varphi_{m'}^{(I)}\rangle\langle\varphi_{m}^{(I)}|$$

This is the conventional linear response and its done in a supercell as the perturbation should not interact with its periodic images. Another approach to compute U and J is known as minimum tracking method [Moynihan2017] [Linscott2018].

16.1.2 Minimum Tracking Method

The minimum tracking method is based on a reformulation of the response matrices based on the ground state density of the perturbed system [Moore2024] . We can redefine the interacting and noninteracting response matrices as (in practice we'll be using simpler yet equivalent formulae)

$$\chi_{IJ} = \frac{dn^{I}}{dv_{\text{ext}}^{J}},$$

$$(\chi_0)_{IJ} = \left[\frac{dn}{dv_{\text{KS}}}\left(\frac{dv_{\text{KS}}}{dv_{\text{ext}}}\right)^{-1}\right]_{IJ}$$

This allows us to work around the practical issues from the conventional linear response. This approach can also be extended to include the J exchange term In practice this is done by modifying the perturbation by including an additional term (spin-splitting):

$$V_{\text{ext}}^{p} = V_{\text{ext}} + \beta \sum_{m,m'} |\varphi_{m'}^{(I\uparrow)}\rangle\langle\varphi_{m}^{(I\uparrow)}| - |\varphi_{m'}^{(I\downarrow)}\rangle\langle\varphi_{m}^{(I\downarrow)}|$$

16.2 Setting up the calculations

We will configure a set (9 total) of bulk hematite single-point calculations to compute U and J for the Fe $3d$ orbitals. We apply distinct labels to Fe atoms, enabling us to assign different parameters to spin-up and spin-down Fe atoms. We will be using a 4x4x1 supercell generated from the conventional cell.

16.2.1 Tutorial files

All the files needed for the simulations can be downloaded from

- `Fe_NCP19_PBE_OTF.usp`,
- `O_NCP19_PBE_OTF.usp`,
- `hematite.out`,
- `hematite.dat`.

16.2.2 Practical calculation

The step by step approach to compute U and J is:

1. add `hubbard_calculating_u : T` in the input file,
2. choose an atom for the atom type we want to compute U or J for, and label it differently. In our case you can see from the input file that we have labelled this single atom Fe1U. It does not matter whether we choose a spin up or spin down atom for an AFM material.
3. apply the perturbation to this atom only and perform single-points calculations,
4. compute U and J with the following formulas:

$$U = \frac{1}{2}\frac{\delta v^{\uparrow}_{\text{Hxc+local}} + \delta v^{\downarrow}_{\text{Hxc+local}}}{\delta(n^{\uparrow} + n^{\downarrow})}$$

$$J = -\frac{1}{2}\frac{\delta v^{\uparrow}_{\text{Hxc+local}} - \delta v^{\downarrow}_{\text{Hxc+local}}}{\delta(n^{\uparrow} - n^{\downarrow})}$$

where $\delta v^{\uparrow}_{\text{Hxc}}$ and $\delta v^{\downarrow}_{\text{Hxc}}$ represent the derivative of the Hxc+local potential with respect to the applied potential (either α to compute U or β to compute J) and $\delta(n^{\uparrow} + n^{\downarrow})$ and $\delta(n^{\uparrow} - n^{\downarrow})$ represent the derivative of the total occupation $n^{\uparrow} + n^{\downarrow}$ with respect to α and of $n^{\uparrow} - n^{\downarrow}$ with respect to β.

16.2.3 How and where to apply the perturbation

Looking at the input file provided you can see we activated the `hubbard_calculating_u` functionality and in the Hubbard block we have

```
%BLOCK HUBBARD
Fe1  2 0.0 0.0 -10.0 0.0 0.0
Fe1U 2 0.0 0.0 -10.0 0.0 0.0
Fe2  2 0.0 0.0 -10.0 0.0 0.0
%ENDBLOCK HUBBARD
```

where the columns of the `hubbard` block are described as follows:

1. **Species Label**

   The species to apply the DFT+U correction to.

2. **Angular Momentum:** l

   The angular momentum of the projectors which the Hubbard correction is applied to. In this example $l = 2$ which corresponds to d orbitals

3. **Hubbard** U **value**

   The value of the Hubbard U for this sub-shell, in electron-volts. We are computing it so we can choose 0 as its value

4. **Hund's exchange** J **value**

   The value of the Hund's exchange J for this sub-shell, in electron-volts. We are computing it so we can choose 0 as its value

5. **Effective Charge Z and Projectors type** The default projectors are NGWFs. For other possibility, refer to the DFT+ U documentation

6. **The** α **prefactor**

   The perturbation term needed to compute U

7. **The spin-splitting factor** β

   The perturbation term needed to compute J.

To compute U you need to change the α value while keeping β equal to 0. To compute J you need to change the β value while keeping α equal to 0.

We have provided you only 1 input file – the one corresponding to 0 for both α and β, you need to generate the remaining 8 files.

The α and β values you need to use for the U calculation are = -0.2, -0.1, 0.0, 0.1, 0.2.

Why these values? We want to apply a big enough perturbation to see an effect and to be able to compute derivatives but also remain in the linear regime. It is not necessary to use 5 datapoints to obtain a good value but it's highly recommended.

16.3 Evaluating the outputs

In order to compute U and J we need the values of the $v^{\uparrow}_{\text{Hxc}}$ and $v^{\downarrow}_{\text{Hxc}}$,which can be found in the following block:

```
#######################################################################
 ↪############
DFT+U information on Hubbard site     72 of species Fe1U and spin  ␣
 ↪    1
The average Hxc+local potential is           -100.04043423 eV.
The average Hubbard potential is               -0.10000000 eV.
#######################################################################
 ↪############
DFT+U information on Hubbard site     72 of species Fe1U and spin  ␣
 ↪    2
The average Hxc+local potential is            -96.03296381 eV.
The average Hubbard potential is               -0.10000000 eV.
#######################################################################
 ↪############
```

Note that we are looking only at the values for Fe1U atom which is the only atom we have applied the perturbation to. There are multiple instances of this block and we are only interested in the last one.

Next, we need to look at occupation of the Hubbard manifold $n^{\uparrow} + n^{\downarrow}$ and $n^{\uparrow} - n^{\downarrow}$,which can be found in the following block:

```
#######################################################################
 ↪############
DFT+U information on atom      1 of Hubbard species Fe1U
#######################################################################
 ↪############
Occupancy matrix of Hubbard site     72 and spin      1 is
m_l =   -2          -1           0           1           2
0.98583311  0.01105739  0.00017283  0.00149346 -0.00039754
0.01106973  0.98239066 -0.00021203  0.00037893  0.00244851
0.00017266 -0.00021405  0.99296562  0.00030517  0.00069962
0.00149451  0.00037878  0.00029134  0.98210951 -0.01203475
-0.00039830  0.00244943  0.00069122 -0.01204334  0.98340592
WARNING: DFT+U ENERGY of Hubbard site     72 and spin      1 is␣
 ↪negative.
#######################################################################
 ↪############
Occupancy matrix of Hubbard site     72 and spin      2 is
m_l =   -2          -1           0           1           2
0.32009924 -0.06393836 -0.00012245 -0.01033413 -0.00070413
-0.06400973  0.33409081 -0.00029354  0.00034179 -0.01142806
-0.00012106 -0.00027777  0.19025018 -0.00114325  0.00745246
-0.01034138  0.00034159 -0.00106271  0.33014982  0.06774687
-0.00070499 -0.01143070  0.00762074  0.06779446  0.29199808
```

(continues on next page)

(continued from previous page)

```
WARNING: DFT+U ENERGY of Hubbard site     72 and spin      2 is␣
↪negative.
#########################################################################
↪############
Total occupancy of Hubbard site     72 is          6.39329292 e
Local magnetic moment of Hubbard site     72 is    3.46011669 mu_B
DFT+U energy of Hubbard site     72 is           -0.02349492 Ha
#########################################################################
↪############
```

The total occupancy of Hubbard site is the $n^{\uparrow} + n^{\downarrow}$, while the local magnetic moment of Hubbard site is the $n^{\uparrow} - n^{\downarrow}$. We now have all the data we need to compute U and J.

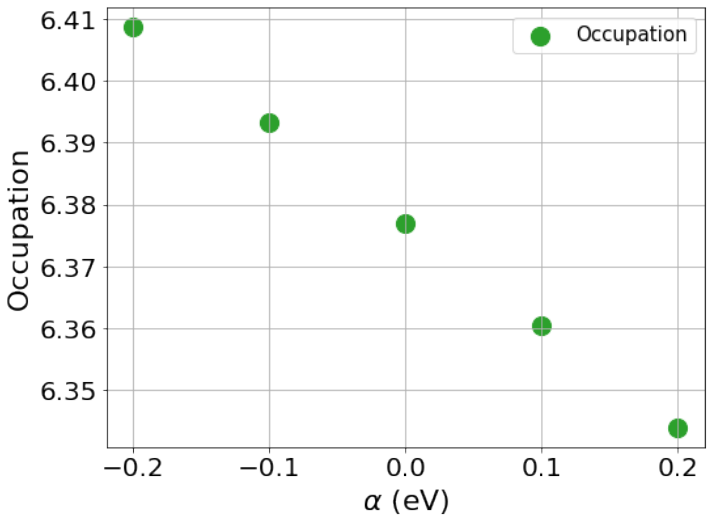
Step by step to compute U :

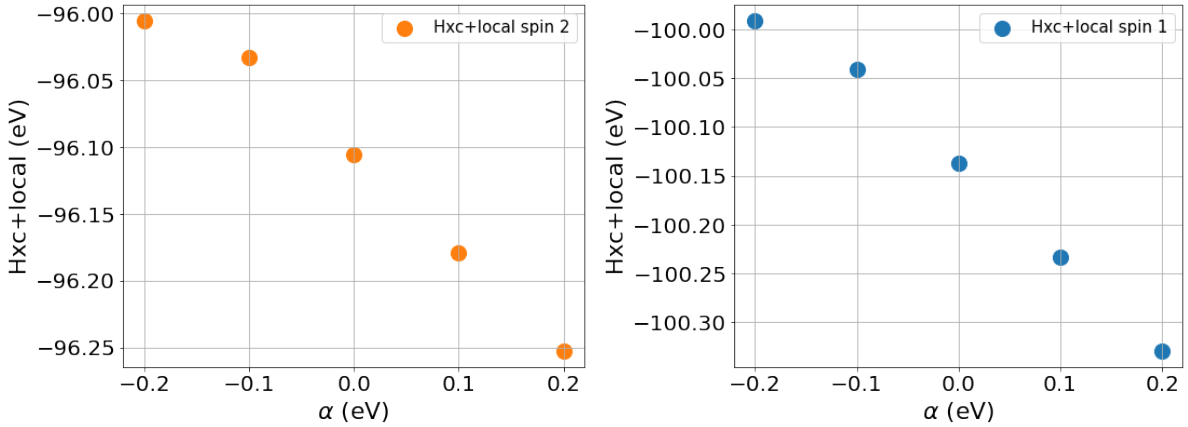
- Calculate the slope of $v^{\uparrow}_{\text{Hxc}}$ and $v^{\downarrow}_{\text{Hxc}}$ with respect to α, these are the $\delta v^{\uparrow}_{\text{Hxc}}$ and $\delta v^{\downarrow}_{\text{Hxc}}$ that appear in the formula to compute U
- Calculate the slope of the $n^{\uparrow} + n^{\downarrow}$ with respect to α this is the denominator appearing in the formula to compute U
- Compute U using the formula provided above.

To compute J follow similar procedure but the derivatives are with respect to β.

IMPORTANT: The actual β values in the calculations are half of the one specified in the input file.

To compute the slope, we first plot the Hxc+local for spin 1 and spin 2 as well as the occupation number against the values of β, the same should be done with values of β to compute J





You can see from the plots that while the changes of the occupation numbers are perfectly linear at all α values, this is not the case for the Hxc+local potential where a degree of non-linearity is present at a value of $\alpha = 0$, this is VERY important as if we were to include this data point in our calculation of U, we would obtain a wrong value as our perturbation goes beyond the linear response regime.

If you discard the non-linear data point, you should obtain the following values.

- $U = 5.158$ eV
- $J = 0.604$ eV

16.3.1 What to do next

The tutorial is now complete, but you could still move forward. What can you do next?

- Compute U for oxygen p states as this is commonly done in transition metal oxides, it's usually large. For more information [Moore2024]

CHAPTER

SEVENTEEN

PDF VERSION OF ALL TUTORIALS

Note: A pdf version of the tutorials can be downloaded `here`

CHAPTER

EIGHTEEN

INDICES AND TABLES

- genindex
- modindex
- search

BIBLIOGRAPHY

[Bowler2012] D.R.Bowler, and T.Miyazaki, O(N) methods in electronic structure calculations, Reports on Progress in Physics, 75 (2012).

[Dziedzic2011] J.Dziedzic, H.H.Helal, C.K.Skylaris, A.A.Mostofi, and M.C.Payne, M.C., *Minimal parameter implicit solvent model for ab initio electronic-structure calculations*, EPL, 95 (2011).

[Fox2014] S.J.Fox, J.Dziedzic, T.Fox, C.S.Tautermann, and C.-K.Skylaris, *Density functional theory calculations on entire proteins for free energies of binding: Application to a model polar binding ste*, Proteins: Structure, Function and Bioinformatics, 82 (2014).

[Gundelach2021] L.Gundelach, T.Fox, C. S.Tautermann, and C.-K.Skylaris, *Protein–ligand free energies of binding from full-protein DFT calculations: convergence and choice of exchange–correlation functional*, Physical Chemistry Chemical Physics, 23 (2021).

[Manz2012] T.A.Manz, and D.S.Sholl, *Improved Atoms-in-Molecule Charge Partitioning Functional for Simultaneously Reproducing the Electrostatic Potential and Chemical States in Periodic and Nonperiodic Materials*, Journal of Chemical Theory and Computation, 8 (2012).

[Mobley2017] D.L.Mobley, and M.K.Gilson, Michael K., *Predicting Binding Free Energies: Frontiers and Benchmarks*, Annual Review of Biophysics, 46 (2017).

[Prentice2020_T8] J.C.A.Prentice, J.Aarons, J.C Womack, A.E.A.Allen, L.Andrinopoulos, L.Anton, R.A.Bell, A.Bhandari, G.A.Bramley, R.J.Charlton, R.J.Clements, D.J.Cole, G.Constantinescu, F.Corsetti, S.M.M.Dubois, K.K.B.Duff, J.M.Escartin, A.Greco, Q.Hill, L.P.Lee, E.Linscott, D.D.O'Regan, M.J.S.Phipps, L.E.Ratcliff, A.Ruiz Serrano, E.W.Tait, G.Teobaldi, V.Vitale, N.Yeung, T.J.Zuehlsdorff, J.Dziedzic, P.D.Haynes, N.D.M.Hine, A.A.Mostofi, M.C.Payne, and C.-K.Skylaris, *The ONETEP linear-scaling density functional theory program*, Journal of Chemical Physics, 152 (2020).

[Wilkinson2014] K.A.Wilkinson, N.D.M.Hine, and C.-K.Skylaris, *Hybrid MPI-OpenMP parallelism in the ONETEP linear-scaling electronic structure code: Application to the delamination of cellulose nanofibrils*, Journal of Chemical Theory and Computation, 10 (2014).

[Womack2018] J.C.Womack, L.Anton, J.Dziedzic, P.J.Hasnip, M.I.J.Probert, and C.-K.Skylaris, *DL-MG: A Parallel Multigrid Poisson and Poisson-Boltzmann*

Solver for Electronic Structure Calculations in Vacuum and Solution, Journal of Chemical Theory and Computation, 14 (2018).

[Genheden2010] S.Genhenden, J.Kongsted, P.Soderhjelm, and U.Ryde, *Nonpolar solvation free energies of protein-ligand complexes*, Journal of Chemical Theory and Computation, 11 (2010).

[Cornell2003] R.M.Cornell et al, in The Iron Oxides, John Wiley & Sons, Ltd, 2003, pp. 9-38.

[Parkinson2016] G.S.Parkinson, Surface Science Reports, vol. 71, no. 1, pp. 272–365, 1 Mar. 1, 2016.

[Naveas2023] Naveas M. et al, iScience 26, 106033, February 17, 2023.

[Huang2016] Huang X. et al, J.Phys.Chem C 2016, 120, 4919-4930.

[Si2020] Si et al, J. Chem. Phys. 152, 024706 (2020).

[O-Regan2012] D.D.O'Regan, N. D. M. Hine, M. C. Payne and A. A. Mostofi, Phys. Rev. B 85, 085107 (2012).

[Cococcioni2005] M.Cococcioni and S. de Gironcoli, Phys. Rev. B 71, 035105 (2005).

[O-Regan2010] D.D.O'Regan, N. D. M. Hine, M. C. Payne and A. A. Mostofi, Phys. Rev. B 82, 081102 (2010).

[Anisimov1991] J.Z.V.I. Anisimov and O. K. Andersen, Phys. Rev. B 44, 943 (1991).

[Anisimov1997] V.I. Anisimov, F. Aryasetiawan, and A. I. Liechtenstein, J. Phys.: Condens. Matter 9, 767 (1997).

[Dudarev1998] S.L. Dudarev, Phys. Rev. B 57, 3 (1998).

[Bhandari2020] Bhandari, A.; Anton, L.; Dziedzic, J.; Peng, C.; Kramer, D.; Skylaris, C.-K. Electronic Structure Calculations in Electrolyte Solutions: Methods for Neutralization of Extended Charged Interfaces. The Journal of Chemical Physics 2020, 153 (12), 124101. https://doi.org/10.1063/5.0021210.

[Fisicaro2017] Fisicaro, G.; Genovese, L.; Andreussi, O.; Mandal, S.; Nair, N. N.; Marzari, N.; Goedecker, S. Soft-Sphere Continuum Solvation in Electronic-Structure Calculations. Journal of Chemical Theory and Computation 2017, 13 (8), 3829–3845. https://doi.org/10.1021/acs.jctc.7b00375.

[Larsen2017] Hjorth Larsen, A.; Jørgen Mortensen, J.; Blomqvist, J.; Castelli, I. E.; Christensen, R.; Dułak, M.; Friis, J.; Groves, M. N.; Hammer, B.; Hargus, C.; Hermes, E. D.; Jennings, P. C.; Bjerre Jensen, P.; Kermode, J.; Kitchin, J. R.; Leonhard Kolsbjerg, E.; Kubal, J.; Kaasbjerg, K.; Lysgaard, S.; Bergmann Maronsson, J. The Atomic Simulation Environment—a Python Library for Working with Atoms. Journal of Physics: Condensed Matter 2017, 29 (27), 273002. https://doi.org/10.1088/1361-648x/aa680e.

[Bhandari2021] Bhandari, A.; Peng, C.; Dziedzic, J.; Anton, L.; Owen, J. R.; Kramer, D.; Chris-Kriton Skylaris. Electrochemistry from First-Principles in the grand canonical Ensemble. Journal of Chemical Physics 2021, 155 (2). https://doi.org/10.1063/5.0056514.

[ONETEP2020] J. C. A. Prentice, J. Aarons, J. C. Womack, A. E. A. Allen, L. Andrinopoulos, L. Anton, R. A. Bell, A. Bhandari, G. A. Bramley, R. J. Charlton, R. J. Clements, D. J. Cole, G. Constantinescu, F. Corsetti, S. M. M. Dubois, K. K. B. Duff, J. M. Escartin, A. Greco, Q. Hill, L. P. Lee, E. Linscott, D. D. O'Regan, M. J. S. Phipps, L. E. Ratcliff, A. Ruiz Serrano, E. W. Tait, G. Teobaldi, V. Vitale, N. Yeung, T. J. Zuehlsdorff, J. Dziedzic, P. D. Haynes, N. D. M. Hine, A. A. Mostofi, M. C. Payne, and C.-K. Skylaris, *The ONETEP linear-scaling density functional theory program*, J. Chem. Phys. 152, 174111 (2020).

[Prentice2020] J. C. A. Prentice, R. J. Charlton, A. A. Mostofi, and P. D. Haynes, *Combining Embedded Mean-Field Theory with Linear-Scaling Density-Functional Theory*, J. Chem. Theory Comput. 16, 354 (2020).

[Prentice2022] J. C. A. Prentice, *Efficiently Computing Excitations of Complex Systems: Linear-Scaling Time-Dependent Embedded Mean-Field Theory in Implicit Solvent*, J. Chem. Theory Comput. 18, 1542 (2020).

[Fornace2015] M. E. Fornace, J. Lee, K. Miyamoto, F. R. Manby, and T. F. Miller, *Embedded Mean-Field Theory*, J. Chem. Theory Comput. 11, 568 (2015).

[Foulkes1989] W.Matthew C.Foulkes, Roger Haydock, *Phys. Rev. B* **1989**, 39, 12520, https://doi.org/10.1103/PhysRevB.39.12520

[Elstner1998] Marcus Elstner et. al., *Phys. Rev. B* **1998**, 58, 7260, https://doi.org/10.1103/PhysRevB.58.7260

[Gaus2014] Michael Gaus, Qiang Cui, Marcus Elstner, "Density functional tight binding: application to organic biological molecules", *WIREs Comput. Mol. Sci.* **2014**, 4, 49, https://doi.org/10.1002/wcms.1156

[Bannwarth2020] Christoph Bannwarth et. al., "Extended tight-binding quantum chemistry methods", *WIREs Comput. Mol. Sci.* **2021**, 11, 1, https://doi.org/10.1002/wcms.1493

[Pracht2019] Philipp Pracht, Eike Caldeweyher, Sebastian Ehlert, Stefan Grimme, "A robust non-self-consistent tight-binding quantum chemistry method for large molecules", *ChemRxiv* **2019**, https://doi.org/10.26434/chemrxiv.8326202.v1

[Grimme2006] Stefan Grimme, "Semi-empirical GGA-type density functional constructed with a long-range dispersion correction", *J. Comput. Chem.* **2006**, 27, 1787, https://doi.org/10.1002/jcc.20495

[Caldeweyher2019] Eike Caldeweyher et. al., "A generally applicable atomic-charge dependent London dispersion correction", *J. Chem. Phys.* **2019**, 150, 154122, https://doi.org/10.1063/1.5090222

[Linscott2018] E.B. Linscott, D. J. Cole, M. C. Payne, D. D. O'Regan, Phys. Rev. B **98**, 235157 (2018). https://doi.org/10.1103/PhysRevB.98.235157

[Moore2024] G. C. Moore, M. K. Horton, E. Linscott, A. M. Ganose, Ma. Siron, D. D. O'Regan, K. A. Persson Phys. Rev. Materials **8**, 014409 (2024). https://doi.org/10.1103/PhysRevMaterials.8.014409

[Moynihan2017] G. Moynihan, G. Teobaldi, and D. D. O'Regan, A self-consistent ground-state formulation of the first-principles Hubbard U parameter validated on one-electron self-interaction error (2017), arXiv:1704.08076