Tutorial 1

Basic SIESTA execution: energy optimization of H_2 molecule

In this tutorial, we perform the energy optimization of a H_2 molecule with SIESTA. Our aim is to give a flavour of the general system descriptors included in the input file of a SIESTA calculation, and a brief description of the self-consistent field method.

1.1 Execution of the program

Enter in the directory /T1.01/H2. This directory contains the files that SIESTA requieres to optimize the energy of a H_2 molecule; H2.fdf (input file) and H.psf (pseudopotential file).

To execute the code, type in the console the following command,

```
SIESTA<H2.fdf>H2.txt
```

where SIESTA is the siesta executable, and H2.fdf and H2.txt are the input and output files, respectively. This simple SIESTA execution will take a few seconds to finish. After a successful run of the program, you should have several files in your directory, see Figure 1.1.

SIESTA users must be aware that there are a series of steps that should be followed to obtain a well converged calculation, see Figure 1.1 . The generation of a pseudopotential is a complex process, and every pseudopotential must be thoroughly checked before use. Tutorial 3 provides basic guidelines on the generation and test of pseudopotentials. Because SIESTA falls into category of methods with atom-centered basis sets, the basis set must also be optimized in order to perform calculations within reasonable time and

Figure 1.1: Schematic representation of the input and outputs files in a SIESTA calculation. The prefix H2 is specified by the parameter SystemLabel in the input file.

precision, see Tutorial 4. The specific parameters that define the SIESTA calculation (k-points, meshcutoff, XC, spin, etc) must also be carefully chosen.

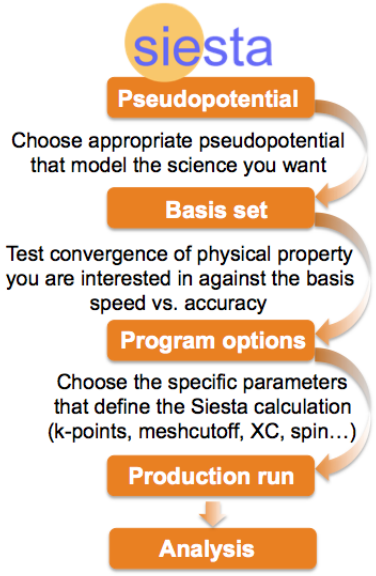

Figure 1.2: Series of steps that should be followed to obtain a well converged calculation

1.2 FDF file: description of program options

The input file contains parameters specifying both the system we want to study, and the accuracy of the calculation. Whereas the former parameters are mandatory, other parameters have default values assigned and, in principle, it is not necessary to explicitly include them in the input file. In this section, we focus on the description of the general

system descriptors. However, it is important to stress out that the default values do not always guarantee a converged calculation, see Tutorials 5, 6 and 7.

1.2.1 General system descriptors

1. **Name and Label**

   SystemName is a string of one or several words containing a descriptive name of the system (max. 150 characters). SystemLabel is a single word (max. 20 characters without blanks) containing a nickname of the system, used to name output files.

```
SystemName    H2     (default_value: blank)
SystemLabel     H2     (default_value: siesta)
```

2. **Chemical species and Atoms**

   As said before, the parameters specifying the system have to be necessarily specified in the input file. Those parameters are the number of different species and atoms present in the unit cell, the list of the different species, atomic numbers, atom positions and the label associated with each atomic specie.

   NumberOfSpecies (integer): Number of different atomic species in the simulation. Atoms of the same species, but with a different pseudopotential or basis set are counted as different species.

   NumberOfAtoms (integer): Number of atoms in the simulation.

   ChemicalSpeciesLabel (data block): It specifies the different chemical species that are present, assigning them a number for further identification. Siesta recognises the different atoms by the given atomic number.

```
NumberOfSpecies     1
NumberOfAtoms       2
%block ChemicalSpeciesLabel
   1 1 H
%endblock ChemicalSpeciesLabel
```

   The first number in a line in the data block ChemicalSpeciesLabel is the species number. It is followed by the atomic number, and then by the desired label. This

label will be used to identify corresponding files, namely, pseudopotential file, user basis file, basis output file, and local pseudopotential output.

3. **Cell and Atoms positions**

The size of the cell itself is specified, using some combination of the options Lattice Constant, LatticeVectors, LatticeParameters, and SuperCell. In this tutorial we use the combination of the first two parameters.

Lattice Constant It is a real parameter that define the scale of the lattice vectors. The value of this parameter is multiplied by each of the matrix LatticeVectors element.

LatticeVectors (block) It is a real matrix that contains (row) vectors that define the lattice. Thus, we have full autonomy over the shape and size of cell. The cell vectors are read in units of the lattice constant defined above. They are read as a matrix CELL(ixyz,ivector), each vector being one line.

Notice that although the system considered in this tutorial is aperiodic (H_2 molecule), SIESTA still does use of periodic boundary conditions (PBC). Thus, it is needed to introduce a vacuum region that should be large enough that periodic images corresponding to adjacent replicas of the supercell do not interact significantly, see Figure 1.3.

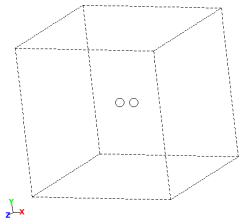

Figure 1.3: H_2 molecule placed in the center of a large box vacuum

```
LatticeConstant        10.00 Ang
%block LatticeVectors
  1.0   0.0   0.0
  0.0   1.0   0.0
```

```
    0.0   0.0   1.0
%endblock LatticeVectors
```

Once that the size of the cell is specified, the positions of the hydrogen atoms within the cells are given using the AtomicCoordinatesAndAtomicSpecies block. In this block the atom index are also indicated. Notice that the hydrogen atoms are placed in the center of the cell, see Figure 1.4.

```
%block AtomicCoordinatesAndAtomicSpecies
  4.5000   5.0000   5.0000    1
  5.5000   5.0000   5.0000    1
%endblock AtomicCoordinatesAndAtomicSpecies
```

Figure 1.4: Schematic illustration of the super-cell approach

1.3 Convergence of the SCF cycles

The self-consistent Kohn-Sham method is used to calculate energies and forces. This method consists in defining an initial electron density to solve Kohn-Sham equations. Then, from the solution of these equations a new electronic density is calculated. The cycle is repeated until a self-consistent electron density is achieved, see Figure 1.5

We can inspect the output file H2.txt to analyse the electronic density convergence

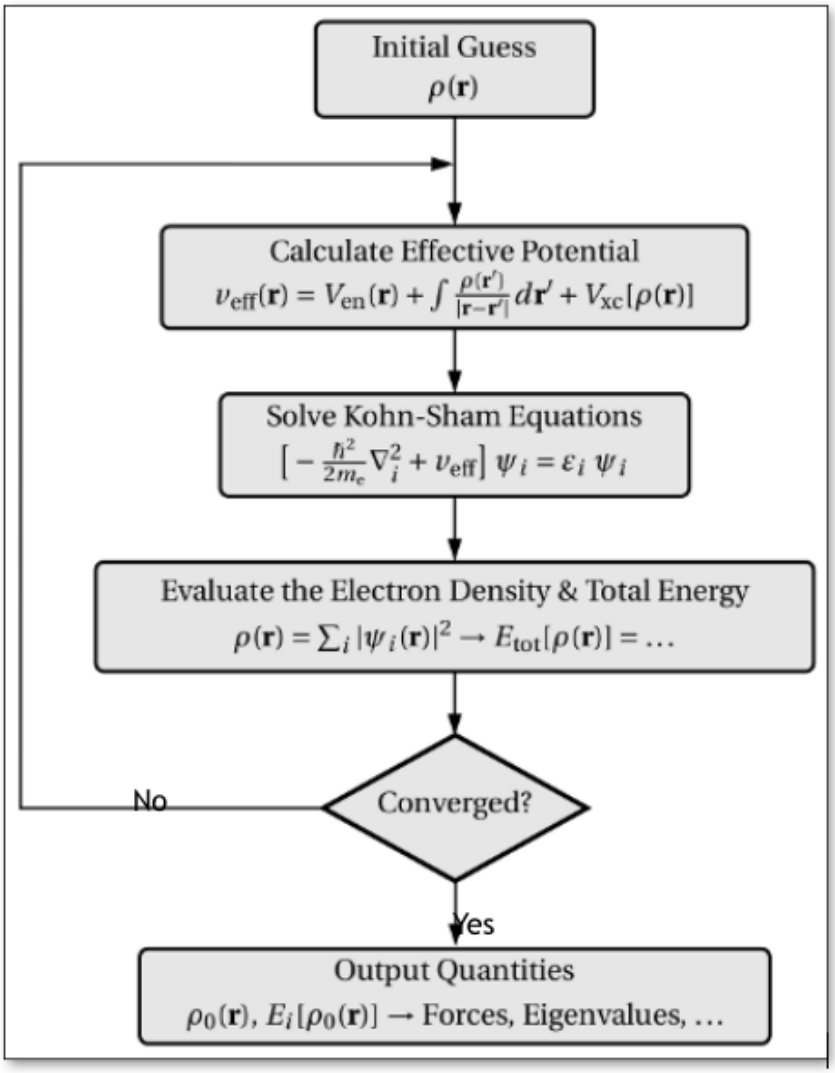

Figure 1.5: Self Consistent Method

of the H_2 molecule. Type in the console the following command to check the energy of the system in each of the SCF steps,

```
cat  H2.txt|grep  scf:
```

```
scf:  iscf    Eharris(eV)       E_KS(eV)    FreeEng(eV)     dDmax   Ef(eV)
scf:     1       -34.1562       -30.0304       -30.0304   0.56242  -2.7224
scf:     2       -34.1379       -34.1291       -34.1291   0.02752  -2.2036
         .          .              .                      .
         .          .              .                      .
scf:    18       -34.1363       -34.1362       -34.1362   0.00007  -2.3057
```

Notice that both total energy and electron density density variation (dDmax) are minimal in the last SCF step, see Figure 1.6 and 1.7. The accuracy of the calculation of energies and forces in a self-consistent-field loop can be tuned by specifying the number of maximum and minimum number of SCF iterations per time step, mixing options, etc. Those parameters are described in advances tutorials.

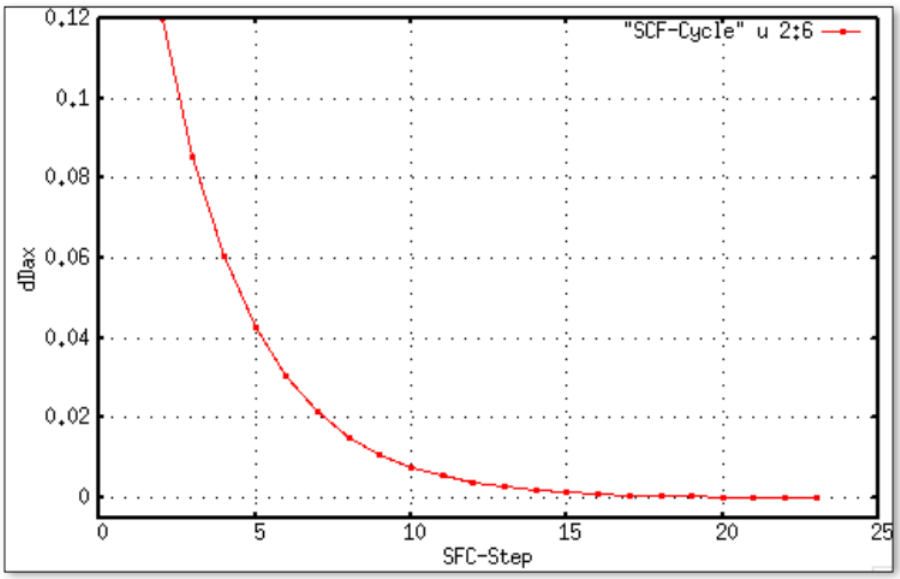

Figure 1.6: Electron density density variation (dDmax) as a function of SCF step

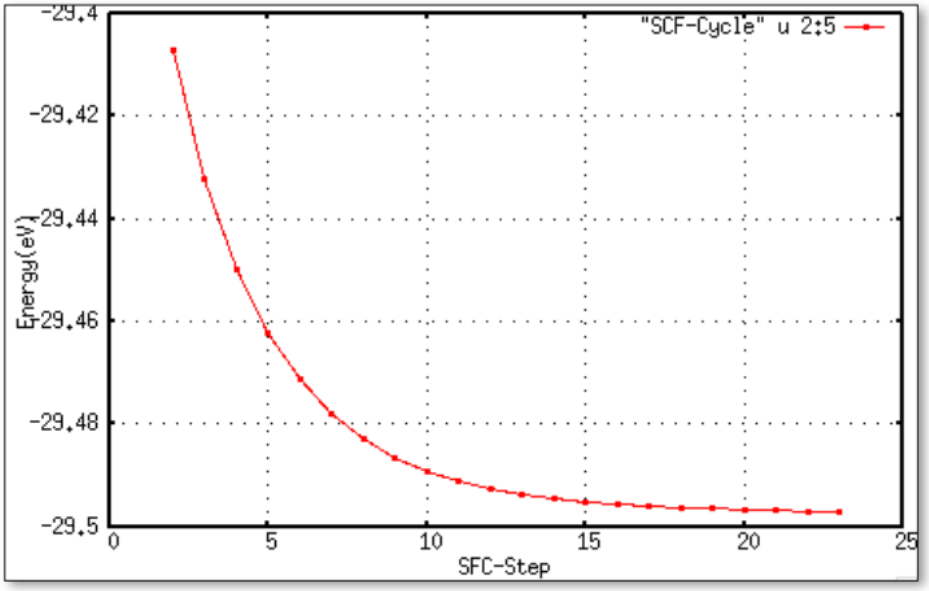

Figure 1.7: Total Energy as a function of SCF step