SIESTA Molecular Orbital Computation for Custom Organic Molecules using Python

Nanometric System Simulation - Nanoscience and Nanotechnology UAB 2022/23

Xabier Oianguren Asua

The objective of this practice was to use the Python library `sisl` [1] to generate the SIESTA [2] configuration and structure files for a custom molecule, to run siesta and learn to plot the molecular orbitals (MOs) using VMD [3], by following steps similar to the ones explained in the `sisl` tutorial Ref. [4].

We generated a pair of scripts that allow the input of any organic molecule involving C, H, O and N atoms. As an example of their use, we will show the results for Benzene C_6H_6 for it is the quintessential aromatic organic molecule.

1 The Scripts

Two scripts were generated for the practice, which can be found in our github repository [5]. The first script, `Pseudopotential_file_generator.py`, contains some functions to generate the pseudopotential profile of each atom in a custom path. The numbers were obtained from the vault offered by the SIESTA developers in Ref. [6].

The second script, `Generate_Molecules_Simulate_and_Plot.py`, is the main script, which is summarized in Listing 1. It takes six command line arguments:

- The positions of Carbon atoms, of Hydrogen atoms, Nitrogen atoms and Oxygen atoms in angstroms, each set of positions as a string that is the list of triplets of 3D coordinates (if no atom then an empty list).
- The "experiment label" to be given to the output folder and files.
- An optional argument indicating the atomic orbitals to be used in the calculation, following the SIESTA `.fdf` syntax. Its default value is `SZP`, meaning to use the necessary orbitals to fill them following Aufbau's principle, but with polarization, which should be enough for basic organic molecules involving the allowed atoms.

One can introduce these arguments by the redirection of an option file, by running

```
python Generate_Molecules_Simulate_and_Plot.py $(cat "example.txt" | sed "s/[[:blank:]]*//g"),
```

for the example input file `example.txt` provided in our github.

Instead of inputting the six arguments or the option file as stated above, one can place the flag `-benzene` or `-b`, which will build the Benzene molecule we show in the Example Results section (under a new directory `./Benzene`). Its atom geometry was obtained from Ref. [7].

The pipeline is as follows. It first generates the molecule geometry with `sisl`, using periodic boundary conditions in a super-cell of sides that are six times bigger than the maximum diameter in the x or y axis. This is plotted for an initial sanity check. Then, the SIESTA configuration file, molecule structure file and pseudo-potential files are generated (within the new directory with the name of the experiment label) and finally the system command line is called from Python to run SIESTA, the log output of which is redirected to a log `.txt` file. The script continues to plot the employed atomic orbitals for the simulation of the first and last introduced atoms together with the `.cube` files. Finally,

using the results of the simulation, a `.cube` file is generated for the HOMO (Highest Occupied MO), the LUMO (Lowest Unoccupied MO) and the electron density, after the HOMO-LUMO wavefunction norms are printed. Finally, VMD is called with the HOMO `.cube` file as an example.

Listing 1: Generate_Molecules_Simulate_and_Plot.py script.

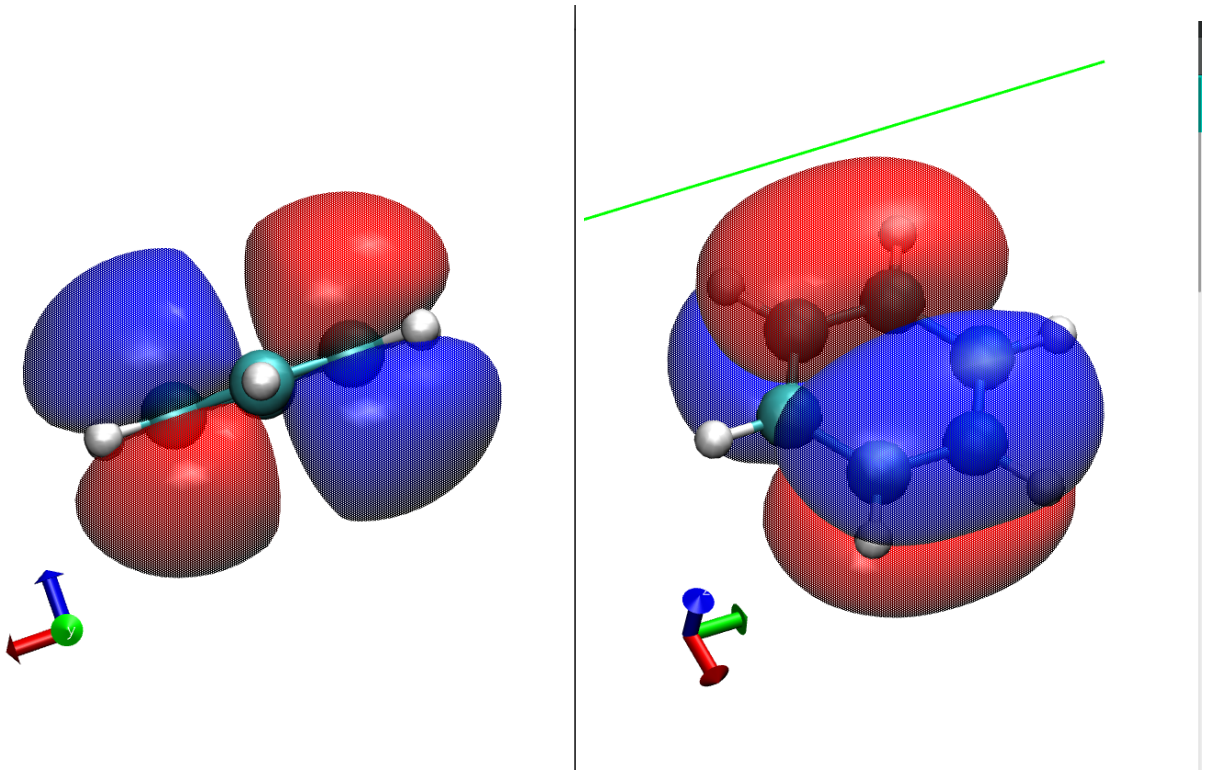
```
1
2  import numpy as np
3  import matplotlib.pyplot as plt
4  import os, sys
5  import sisl
6  from Pseudopotential_file_generator import *
7
8  def string_to_float_list_of_3lists(string):
9      '''Converts the list of 3 lists as a string to an
10     actual list of lists.
11     '''
12     if string in ["[]", "[[]]"]:
13         return []
14     string = string.split("[[")[-1].split("]]")[0].split(",")
15     l=[]
16     for k,st in enumerate(string):
17         if k%3==0:
18             l.append([])
19         l[-1].append(st.split("[")[-1].split("]")[0].split(",")[0])
20     lout=[]
21     for sub in l:
22         subb = []
23         for el in sub:
24             subb.append(float(el))
25         lout.append(subb)
26     return lout
27
28 def get_max_diam(ls):
29     max_p = -np.inf
30     min_n = np.inf
31     for l in ls:
32         for sub in l:
33             for k in sub:
34                 if k<min_n:
35                     min_n=k
36                 if k>max_p:
37                     max_p=k
38     return max_p-min_n
39
40
41 if __name__ == "__main__":
42     if str(sys.argv[1]) in ["-benzene", "-b"]:
43         exp_label="Benzene"
44         # Define positions of atoms
45         CC = 1.39 #A
46         CH = 1.09 #A
47         positions_C = [
48             [0, CC, 0], [0, -CC, 0],
49             [np.sqrt(3)/2*(CC), (CC)/2, 0], [np.sqrt(3)/2*(CC), -(CC)/2, 0],
50             [-np.sqrt(3)/2*(CC), (CC)/2, 0], [-np.sqrt(3)/2*(CC), -(CC)/2, 0]]
51
52         positions_H = [
53             [0, CC+CH, 0], [0, -(CC+CH), 0],
54             [np.sqrt(3)/2*(CC+CH), (CC+CH)/2, 0], [np.sqrt(3)/2*(CC+CH), -(CC+CH)/2, 0],
55             [-np.sqrt(3)/2*(CC+CH), (CC+CH)/2, 0], [-np.sqrt(3)/2*(CC+CH), -(CC+CH)/2, 0]]
56         positions_N=[]
57         positions_O=[]
58         orbs="SZP"
59     else:
60         exp_label = str(sys.argv[5])
61         positions_C = string_to_float_list_of_3lists(sys.argv[1])
62         positions_H = string_to_float_list_of_3lists(sys.argv[2])
63         positions_N = string_to_float_list_of_3lists(sys.argv[3])
64         positions_O = string_to_float_list_of_3lists(sys.argv[4])
65         try:
66             orbs=sys.argv[6]
67         except:
68             orbs = "SZP"
69
```

```
    os.makedirs(exp_label, exist_ok=True)
    os.chdir(exp_label)

    cell_side=6*get_max_diam([positions_C, positions_H, positions_N, positions_O])
    # Generate molecule instance
    benzene = sisl.Geometry(positions_C+positions_H+positions_N+positions_O,
                [sisl.Atom('C')]*len(positions_C)+[sisl.Atom('H')]*len(positions_H)+
                [sisl.Atom('N')]*len(positions_N)+[sisl.Atom('O')]*len(positions_O),
                sc=sisl.SuperCell(cell_side, origin=[-cell_side/2] * 3))
    # Sanity check
    sisl.plot(benzene)
    plt.show()

    # Generate SIESTA configuration file
    print("\n\n>> Generating SIESTA configuration Files...")
    with open(f'RUN_{exp_label}.fdf', 'w') as f:
        f.write(f"""%include STRUCT_{exp_label}.fdf
                    SystemLabel siesta_{exp_label}
                    PAO.BasisSize {orbs}
                    MeshCutoff 250. Ry
                    CDF.Save true
                    CDF.Compress 9
                    SaveHS true
                    SaveRho true
                    """)
    # Generate molecule structure file
    benzene.write(f"STRUCT_{exp_label}.fdf")

    # Generate Pseudopotential Files
    print("\n\n>> Generating Pseudopotential Files...")
    generate_C_psf("./C.psf")
    generate_H_psf("./H.psf")
    if len(positions_N)!=0:
        generate_N_psf("./N.psf")
    if len(positions_O)!=0:
        generate_O_psf("./O.psf")

    # Run SIESTA
    print("\n\n>> Running Siesta...")
    com = f"siesta RUN_{exp_label}.fdf > siesta_terminal_log.txt"
    os.system(com)

    # Import results
    print("\n\n>> Importing Results...")
    fdf = sisl.get_sile(f'RUN_{exp_label}.fdf')
    H = fdf.read_hamiltonian()
    # Create a short-hand to handle the geometry
    benzene = H.geometry
    print(f"\n\n\nObtained result geometry : \n\n{H} \n\n")

    # We plot the fitted orbitals
    print("\n\n>> Generating AO Plots (and their .cube versions) for two example atoms...")
    plot_atom(benzene.atoms[0])
    plot_atom(benzene.atoms[-1])
    plt.show()


    print("\n\n>> Generating Molecular Orbiotals and .cube versions for HOMO and LUMO...")
    #Function integrate
    def integrate(g):
        print('Real space integrated wavefunction: {:.4f}'.format((np.absolute(g.grid) ** 2).
    sum() * g.dvolume))
    #Eigenstates
    es = H.eigenstate()
    # We specify an origin to center the molecule in the grid
    benzene.sc.origin = [-cell_side/2]*3
    # Reduce the contained eigenstates to only the HOMO and LUMO
    # Find the index of the smallest positive eigenvalue
    idx_lumo = (es.eig > 0).nonzero()[0][0]
    es = es.sub([idx_lumo - 1, idx_lumo])
    g = sisl.Grid(0.2, sc=benzene.sc)

    #HOMO
    es.sub(0).wavefunction(g)
    integrate(g)
    g.write('HOMO.cube')
```

```
145     g.fill(0) # reset the grid values to 0
146     #LUMO
147     es.sub(1).wavefunction(g)
148     integrate(g)
149     g.write('LUMO.cube')
150 
151         # Density
152     print("\n\n>> Generating Density .cube File....")
153     density = sisl.get_sile(f"siesta_{exp_label}.nc").read_grid(name='Rho')
154     density.set_geometry(fdf.read_geometry())
155     density.write("DENSITY.cube")
156 
157     # call vmd
158     os.system("vmd HOMO.cube")
```

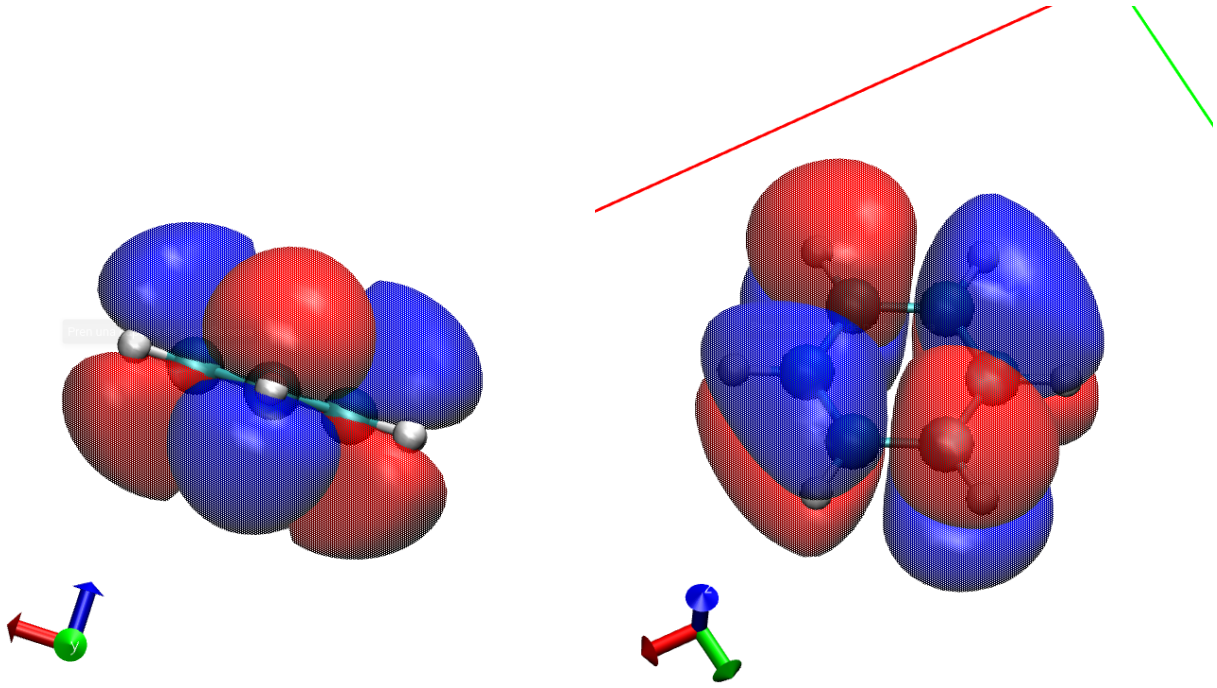
2 Example Results

As a use-case, we generated the HOMO and LUMO for Benzene as plotted in Figures 1 and 2. If one checks then Figure 5.2. of Ref. [8], (or our Figure 4), one can assert that the obtained results are in accordance with the previous bibliography. Finally, in Figure 3 we represent the obtained net electron density (the level surface at a density of 0.5). We see that the electron density leaves the center of the ring with a deficiency of electrons as compared to the rest of the molecule, which is a well known feature of benzene [8].



(a) (b)

Figure 1: Two views of the HOMO of benzene. Two surface levels of the real wavefunctions are shown for the same levels in absolute value: negative in blue and positive in red.



(a) **(b)**

Figure 2: Two views of the LUMO of benzene. Two surface levels of the real wavefunctions are shown for the same levels in absolute value: negative in blue and positive in red.

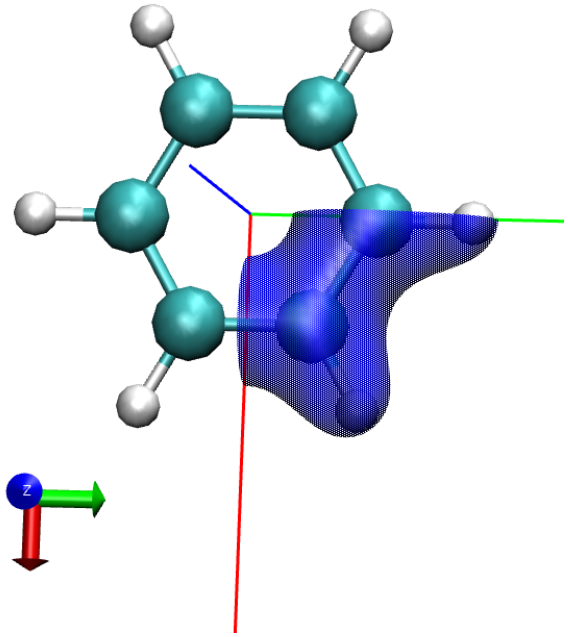


Figure 3: Integrated electron density for Benzene shown in a single quadrant. The 0.5 density surface level is shown.

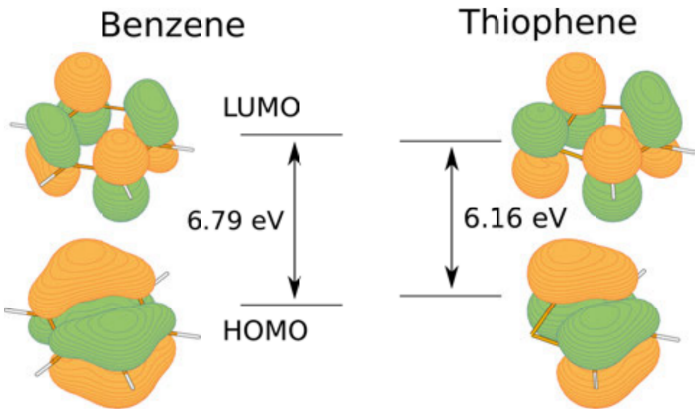


Figure 4: Figure taken from Ref. [8]. We can see the electron density of HOMO and LUMO of benzene and thiophene and their HOMO-LUMO gap, calculated through Gaussian 16, with B3LYP and 6-31G(d,p). Benzene is in great agreement with our result.

References

[1] N. Papior, "sisl," 2022.

[2] A. García, N. Papior, A. Akhtar, E. Artacho, V. Blum, E. Bosoni, P. Brandimarte, M. Brandbyge, J. I. Cerdá, F. Corsetti, R. Cuadrado, V. Dikan, J. Ferrer, J. Gale, P. García-Fernández, V. M. García-Suárez, S. García, G. Huhs, S. Illera, R. Korytár, P. Koval, I. Lebedeva, L. Lin, P. López-Tarifa, S. G. Mayo, S. Mohr, P. Ordejón, A. Postnikov, Y. Pouillon, M. Pruneda, R. Robles, D. Sánchez-Portal, J. M. Soler, R. Ullah, V. W.-z. Yu, and J. Junquera, "Siesta: Recent developments and applications," *The Journal of Chemical Physics*, vol. 152, no. 20, p. 204108, 2020.

[3] W. Humphrey, A. Dalke, and K. Schulten, "VMD – Visual Molecular Dynamics," *Journal of Molecular Graphics*, vol. 14, pp. 33–38, 1996.

[4] "Tutorial given in the `sisl` documentation." https://sisl.readthedocs.io/en/latest/tutorials/tutorial_siesta_1.html.

[5] "Github repository with the python scripts and notebook generated for the report." https://github.com/Oiangu9/_Miscellaneous/tree/main/SSN.

[6] "Offial siesta pseudopotential valut." https://departments.icmab.es/leem/SIESTA_MATERIAL/Databases/Pseudopotentials/periodictable-gga-abinit.html.

[7] "Wikipedia entry of the benzene moleculee." https://en.wikipedia.org/wiki/Benzene.

[8] T. Zhang, *Synchrotron Radiation Studies of Molecular Building Blocks for Functional Materials*. PhD thesis, Acta Universitatis Upsaliensis, 2018.