Graphene and Hexagonal Boron Nitride Band Structures with SIESTA and Python

Nanometric System Simulation - Nanoscience and Nanotechnology UAB 2022/23

Xabier Oianguren Asua

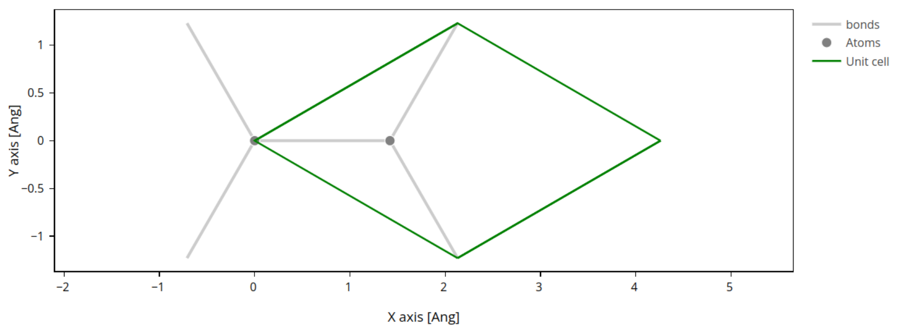
In this practice we simulate the band structures of graphene and hexagonal boron nitride (hBN) using the SIESTA software [1, 2], employing a Python script. We will first see the atom geometries we employed, then show the simulation parameter convergence for both of them and finally we will review the results and their discussion.

In the Python script we employed the `sisl` library [3] for the generation of the SIESTA inputs and part of the output parsing, the `matplotlib` [4] library was employed for plotting, and the native package `os` was used for the operative system calls. All the generated code can be found in our github repository [5]. The pseudopotential data for each atom was retrieved from Ref. [6].

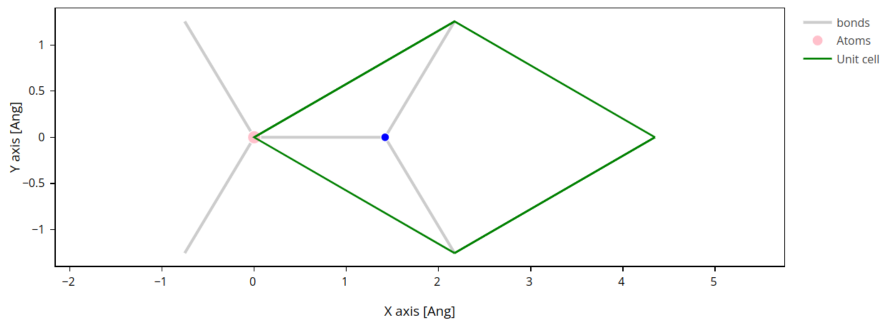
1 The Primitive Cells

The `sisl` library has a function for the direct generation of the graphene atom geometry, which is given by a 2D rhombus primitive cell (with an angle of 60 deg and sides of 2.46 Å) with two atoms, one placed in the major axis vertex and the other one along this same axis a distance of 1.42 Åaway. See Figure 1.(a). Since SIESTA performs 3D simulations, it was necessary to set a third orthogonal axis to the primitive cell, but was chosen to be 14.2Ålong, in order to isolate as much as possible the graphene layers along this undesired axis.

The geometry of hBN is the same as graphene, but with the lengths of the unit cell scaled by a factor of 2.05 % (2.51 Å) [7]. See Figure 1.(b). As an example, the generation of the geometry object in *sisl* for hBN is provided in Listing 1.



(a) Graphene



(b) hBN

Figure 1: The primitive cells for graphene (a) and hBN (b). Red depicts the B atom and N is blue.

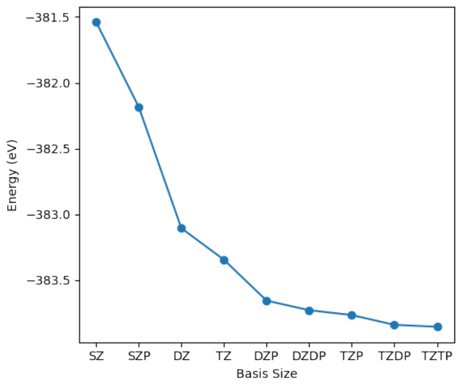
Listing 1: Generation of hBN primitive cell.

```python
import sisl
graphene = sisl.geom.graphene()
hBN = sisl.Geometry( graphene.axyz(),
                     [sisl.Atom('B'), sisl.Atom('N')],
                     graphene.sc.scale(1.0205))
# Plot it to see that it is what we wanted
hBN.plot(axes="xy")
```

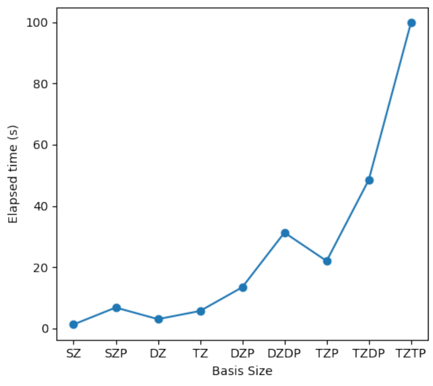
2 Choosing the Right Simulation Parameters

2.1 Basis Size - Convergence

Fixing a k-space grid of $(Nk_x, Nk_y, Nk_z) = (13, 13, 1)$ points along each direction of the reciprocal cell, we sweep the possible localized atomic orbital basis sizes in ['SZ', 'SZP', 'DZ', 'TZ', 'DZP', 'DZDP', 'TZP', 'TZDP', 'TZTP'] in the common notation that can be found in the SIESTA manual [8]. We find that in both graphene and hBN the total energy converges towards a stable value as we increase the complexity of the basis (Figures 2, 3 a), but so does the computation time required (Figures 2, 3 b). We find in both cases that the 'TZP' (triple z polarized) basis gives a converged enough result for our purposes in a reasonable computation time (since a bigger basis set requires double the computation time). The core of the code employed for the analysis can be found in Listing 2 exemplified for hBN.

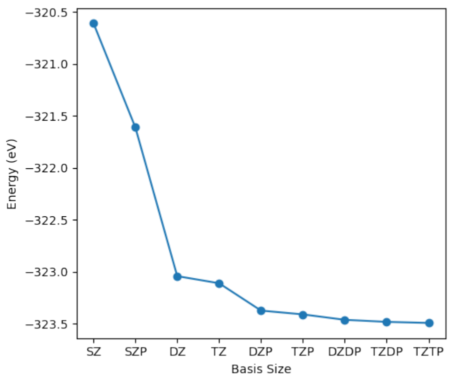


(a) Total Energy

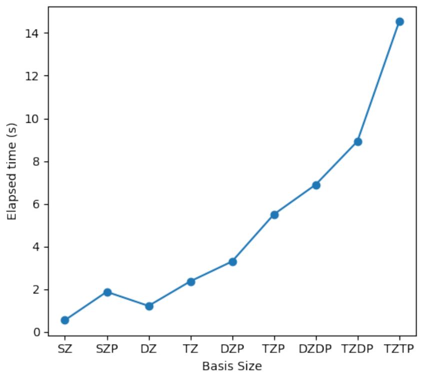


(b) Computation time

Figure 2: In (a), the total energy to which the SIESTA algorithm has converged with each basis size in graphene. In (b) the total time required for each of these calculations.



(a) Total energy



(b) Computation time

Figure 3: In (a), the total energy to which the SIESTA algorithm has converged with each basis size in hBN. In (b) the total time required for each of these calculations.

Listing 2: SIESTA simulation for each basis size. Note that we import the pseudopotential file generator functions we wrote in a separate script Pseudopotential_file_generator.py, also available in the Github repository [5].

```python
import os
from Pseudopotential_file_generator import *
# Define all the basis sizes that we want to try.
basis_sizes = ['SZ', 'SZP', 'DZ', 'TZ', 'DZP', 'DZDP', 'TZP', 'TZDP', 'TZTP']
Nkx=13
Nky=13
Nkz=1

for basis_size in basis_sizes:
    print(f"RUNNING SIMULATION FOR BASIS: {basis_size} K GRID: Nkx={Nkx} Nky={Nky} Nkz={Nkz}")
    out_dir = output_path+f"hBN_Basis_{basis_size}_Nkx={Nkx}_Nky={Nky}_Nkz={Nkz}/"
    os.makedirs(out_dir, exist_ok=True)
    generate_B_psf(out_dir+"/B.psf")
    generate_N_psf(out_dir+"/N.psf")
    hBN.write(out_dir+ "/geom.fdf")

    # Inside this directory, open a RUN.fdf file, which will be our main input file.
    with open(out_dir+ '/RUN.fdf', 'w') as f:
        # We include the fdf file that contains the geometry, with the %include statement
        # Note the \n, which means "new line" in text files.
        f.write("%include geom.fdf \n")
        # We now include the basis size input.
        f.write(f"PAO.BasisSize {basis_size}\n")
        #f.write("TS.HS.Save true\n") # save Hamiltonian for band structure analysis
        #f.write("SaveRho true\n") # save density of electrons in geometry
        k_grid=f"%block kgrid.MonkhorstPack\n{Nkx} 0 0  0\n 0 {Nky} 0  0\n   0 0 {Nkz}  0\n%
    endblock kgrid.MonkhorstPack\n"
        f.write(k_grid)

    # We have the directory set up, we can run the siesta calculation
    os.system(f"cd {out_dir}; {siesta_path} RUN.fdf > RUN.out")
```

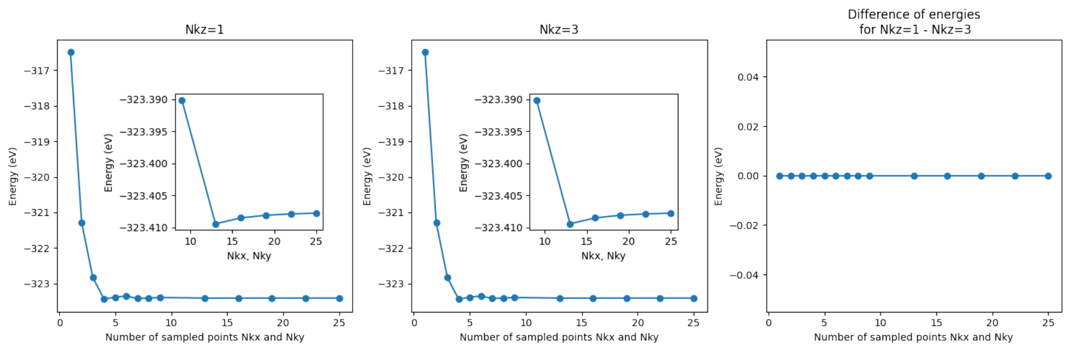
2.2 Number of Sampled Points in k-space - Convergence

Given we fixed the basis size at 'TZP', we now sweeped over a range of different k-space grids and made a similar convergence analysis. Since graphene is a 2D material and we placed a big vacuum range between different 2D layers, we should have enough with a number of k in the z axis equal to 1 ($Nk_z = 1$). Yet, just in case, we tried the analysis with $Nk_z = 1$ and $Nk_z = 3$ to see if there was any difference. With respect to the in-plane sampling, since the cell is symmetric for both x and y in both materials, we decided to increase the k sampling symmetrically, along a grid that went from $(Nk_x, Nk_y) = (1, 1)$ till $(Nk_x, Nk_y) = (27, 27)$. The resulting metrics are plotted in Figure 4 for graphene and Figure 5 for hBN.

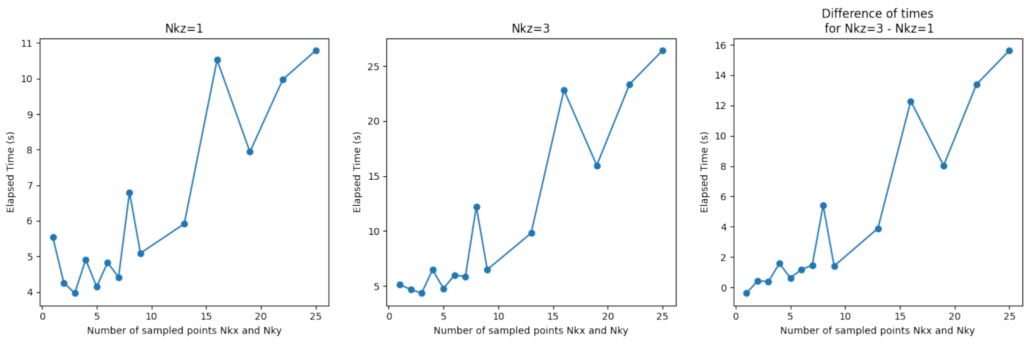
On the one hand, we can see that in the case of graphene, $Nk_x, Nk_y = 13$ is enough for a precision within the first decimal place (see the insets in Figures 4.(a)), which is quite convenient since the computational time after that increases to more than double. In the case of hBN the convergence is even finer, since from $Nk_x, Nk_y = 9$ on, the results only change in the third decimal place. However, this is at the cost of computational time, which is bigger in all cases for hBN. All in all, we choose $Nk_x, Nk_y = 9$ as a sufficient grid for hBN, while $Nk_x, Nk_y = 13$ is the most convenient one for graphene.

On the other hand, we see that changing Nk_z from 1 to 3 does not change the resulting energies at all, while the computational time is vastly worsen. Thus, $Nk_z = 1$ is the way to go.

The core of the code employed for the simulations can be found in Listing 3 following the example of hBN. The parsing of the output file is exemplified in Listing 4 for hBN and the present analysis.

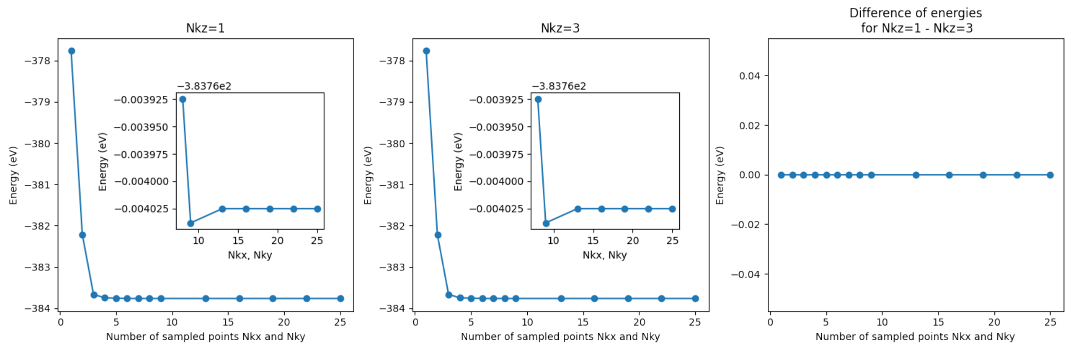


(a) Total Energy

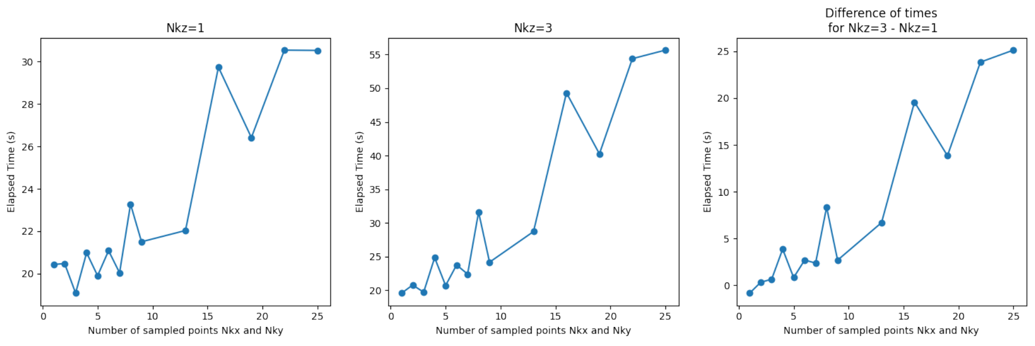


(b) Computation time

Figure 4: Graphene. In (a), the total energy to which the SIESTA algorithm has converged with each k grid size. In (b) the total time required for each of these calculations. In (a) and (b), from left to right, the curve for $Nk_z = 1$, the one for $Nk_z = 3$ and the difference in obtained energy or times in both cases.



(a) Total Energy



(b) Computation time

Figure 5: hBN. In (a), the total energy to which the SIESTA algorithm has converged with each k grid size. In (b) the total time required for each of these calculations. In (a) and (b), from left to right, the curve for $Nk_z = 1$, the one for $Nk_z = 3$ and the difference in obtained energy or times in both cases.

Listing 3: SIESTA simulation for each k-grid. Note that we import the pseudopotential file generator functions we wrote in a separate script Pseudopotential_file_generator.py, also available in the Github repository [5].

```python
# Define the grid of parameters to try
basis_size = 'TZP'
max_Nk = 27
Nkxs = 2*(list(range(1,10))+list(range(13, max_Nk, 3)))
Nkys = 2*(list(range(1,10))+list(range(13, max_Nk, 3)))
Nkzs = [1]*len(Nkxs)+[3]*len(Nkxs)
for Nkx, Nky, Nkz in zip(Nkxs, Nkys, Nkzs):
    print(f"RUNNING SIMULATION FOR BASIS: {basis_size} K GRID: Nkx={Nkx} Nky={Nky} Nkz={Nkz}")
    out_dir = output_path+f"hBN_Basis_{basis_size}_Nkx={Nkx}_Nky={Nky}_Nkz={Nkz}/"
    os.makedirs(out_dir, exist_ok=True)
    generate_N_psf(out_dir+"/N.psf")
    generate_B_psf(out_dir+"/B.psf")
    hBN.write(out_dir+ "/geom.fdf")
    # Inside this directory, open a RUN.fdf file, which will be our main input file.
    with open(out_dir+ '/RUN.fdf', 'w') as f:
        # We include the fdf file that contains the geometry, with the %include statement
        # Note the \n, which means "new line" in text files.
        f.write("%include geom.fdf \n")
        # We now include the basis size input.
        f.write(f"PAO.BasisSize {basis_size}\n")
        f.write("TS.HS.Save true\n") # save Hamiltonian for band structure analysis
        f.write("SaveRho true\n") # save density of electrons in geometry
        k_grid=f"%block kgrid.MonkhorstPack\n{Nkx} 0 0  0\n 0 {Nky} 0  0\n   0 0 {Nkz}  0\n%
    endblock kgrid.MonkhorstPack\n"
        f.write(k_grid)

    # We have the directory set up, we can run the siesta calculation
    os.system(f"cd {out_dir}; {siesta_path} RUN.fdf > RUN.out")
```

Listing 4: Gathering of the total energy and time of each simulation after Listing 3.

```python
# Initialize an empty list to store the energies.
Es, times = [], []
# Loop over Nks
for Nkx, Nky, Nkz in zip(Nkxs, Nkys, Nkzs):
    # Define the directory where this basis size calculation has run.
    out_dir = output_path+f"hBN_Basis_{basis_size}_Nkx={Nkx}_Nky={Nky}_Nkz={Nkz}/"
    # Initialize a parser for the output file (RUN.out)
    out_file = sisl.get_sile(out_dir + "/RUN.out")
    energies = out_file.read_energy()
    Etot = energies['total']
    Es.append(Etot)
    # look for elapsed time
    with open(out_dir + "/RUN.out", "r") as f:
        for line in f:
            if "Elapsed wall time (sec)" in line:
                times.append(float(line.split("time (sec) = ")[-1]))
```

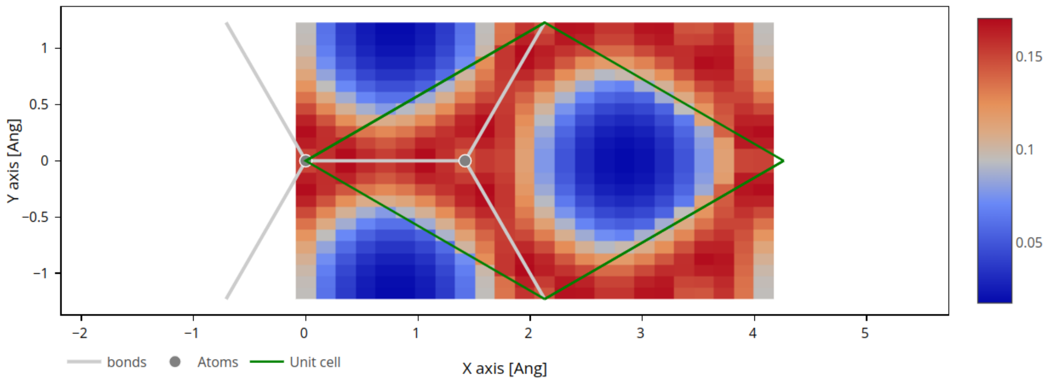
Listing 5: Taking the Hamiltonian generated with SIESTA, we can compute the band structure and plot the different graphics shown in Section 3 as follows.

```python
out_dir = output_path+f"hBN_Basis_{basis_size}_Nkx={13}_Nky={13}_Nkz={1}/"
H = sisl.get_sile(f"{out_dir}/RUN.fdf").read_hamiltonian()
# We need to define a path of k points
band_struct = sisl.BandStructure(H, points=[[0, 0, 0], [2/3, 1/3, 0], [1/2, 0, 0]],
    divisions=100, names=[r"\Gamma", "M", "K"])
# Then we can plot the bands
band_struct.plot()
# Get the fatbands plot and split the contributions by the n, l,m
fatbands = band_struct.plot.fatbands()
fatbands.split_groups(on="n+l+m", scale=15)
# Get the PDOS plot
pdos_plot = H.plot.pdos( kgrid=[40,40,1], nE=1000, Erange=[-10, 10],
    distribution={"method": "gaussian", "smearing": 0.1})
pdos_plot.split_DOS(on="n+l", name="Atom $atoms")
# Get the electronic density
rho = sisl.get_sile(f"{out_dir}/RUN.fdf").read_grid("RHO")
rho.plot(axes="xy", plot_geom=True)
```

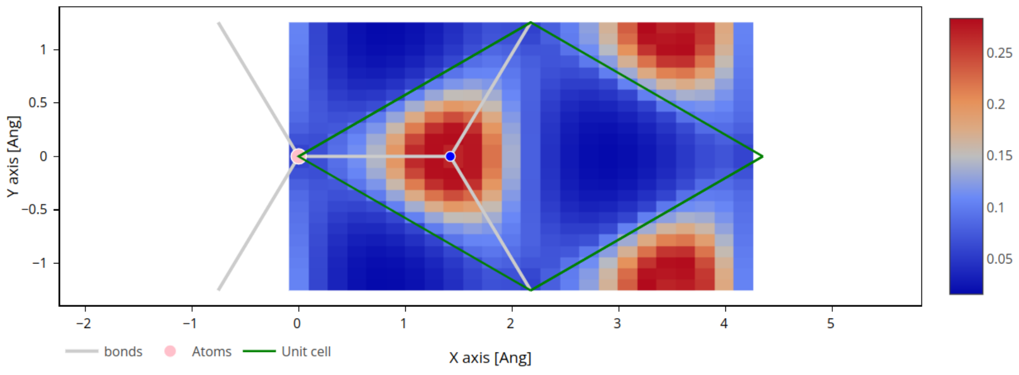
3 Band Structures - Results and Discussion

In the definitive SIESTA simulations we also dumped the converged Hamiltonians. Taking them with the `sisl` library, we can compute the band structure and plot the density of states and other interesting results from Python, as done in Listing 5 as an example for hBN.

In Figure 6 we plot the electronic density obtained for graphene and hBN. We see that while graphene offers a continous electron density connecting all sites along the lattice, in hBN, the electron density is mainly concentrated around the nitrogen atoms (which was expected since B and N have a difference in electronegativity roughly equal to the one hydrogen and oxygen have in water -a highly polarized molecule-). This localized density already tells us that electrons will have a hard time to freely move across the lattice.



(a) Graphene

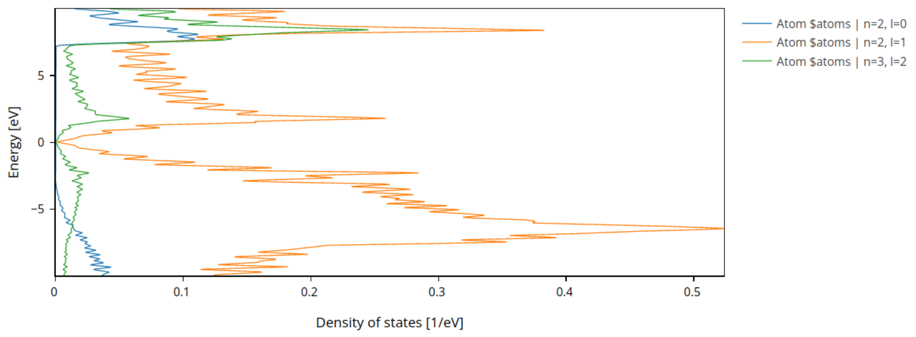


(b) hBN

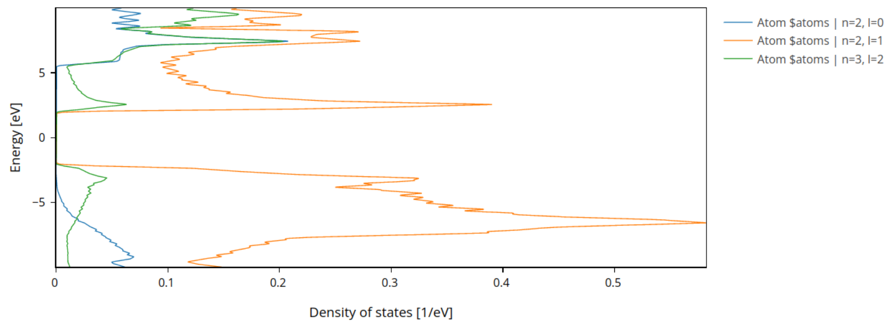
Figure 6: Electron density computed with *sisl* employing the Hamiltonian obtained with SIESTA.

Indeed we can confirm this if we plot the density of states around the Fermi level (set at 0 eV), as done in Figure 7. What we see is that while graphene shows a continuity of states in all the energy range, with the exception of an infinitesimal step at the Fermi level with no states (the Dirac cone's vertex), hBN has a banned energy gap from around $-2eV$ to $2eV$. This indicates that while graphene is a conductor, hBN is an insulator, with a band gap of more than 4 eV (confirmed in Ref. [7]) and cannot conduct electricity except for very high temperatures, where it might behave as a semiconductor. It must be noted that in the plotted energy ranges most of the density of states is due to the p orbitals $(l = 1)$ in $n = 2$, for both materials.

If we plot the unfolded band structure as done in Figure 8, we find that effectively, around the Fermi energy, graphene has a diabolical band crossing with the vertex in the Fermi energy, that joins the occupied and unoccupied states at $T = 0k$. This is the so-called Dirac cone [9]. On the contrary, even if the surrounding bands seem to be very similar, in the case of hBN, around the Fermi level there is no available state in the plotted path of k. This is why they have so different electronic properties.

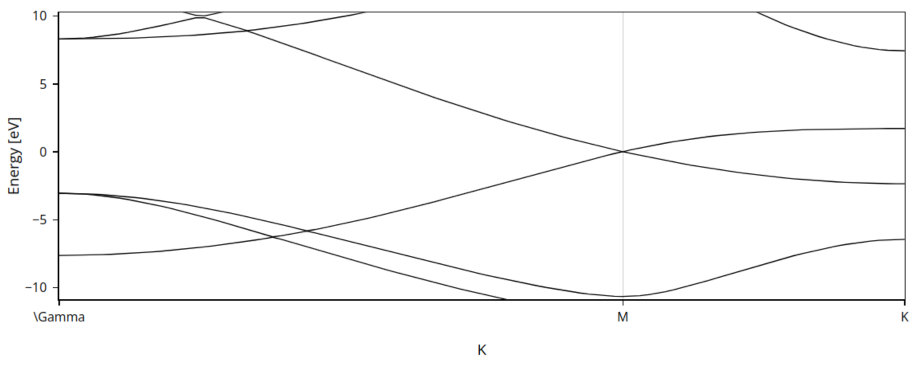


(a) Graphene

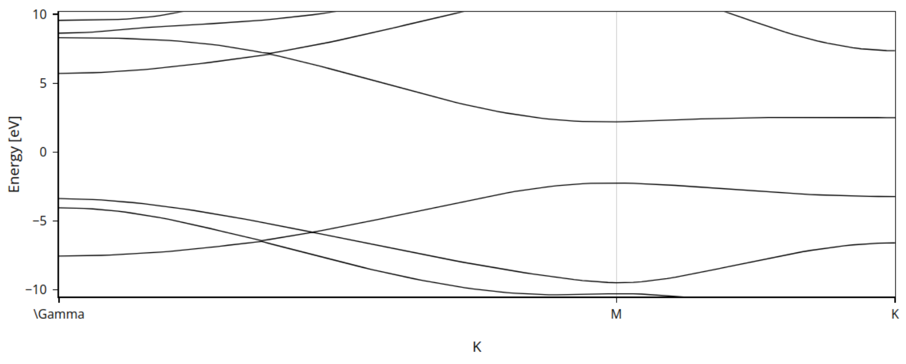


(b) hBN

Figure 7: Density of states in $E \in [-10, 10]$ eV computed with *sisl* employing the Hamiltonian obtained with SIESTA. Fermi level at $E = 0eV$. The density of states is plotted per contributions by the different orbitals employed in the simulation. Clearly, most of the states around the Fermi level are made of p orbitals ($l = 1$) of the $n = 2$ level in both compounds, which is the last level following Aufbau's principle for either B, C, N.



(a) Graphene

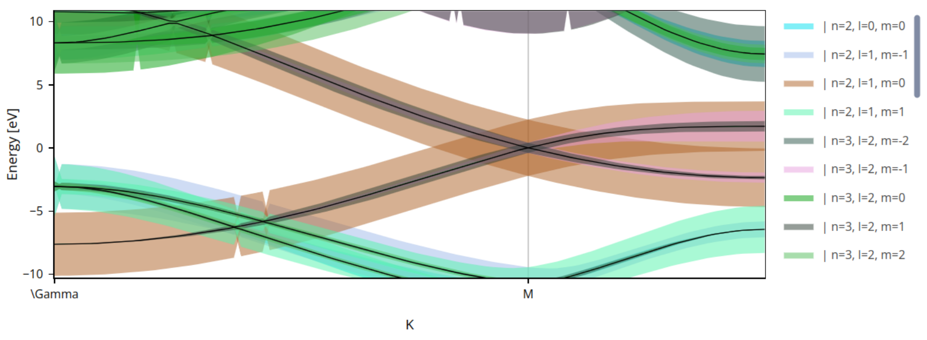


(b) hBN

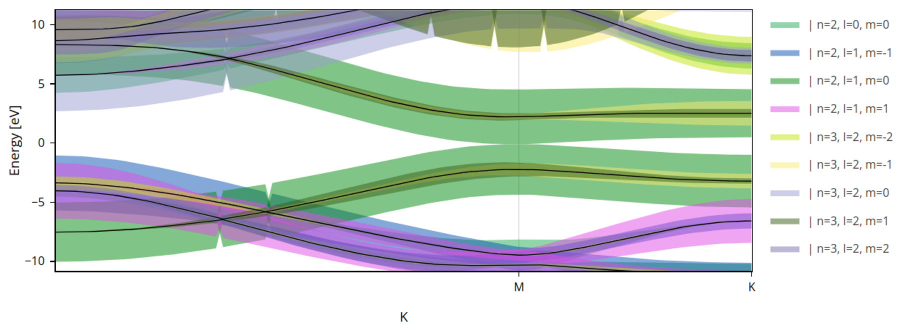
Figure 8: Band structure in $E \in [-10, 10]$ eV computed with *sisl* employing the Hamiltonian obtained with SIESTA. Fermi level at $E = 0eV$. Note the similarity of the diagrams except for the band crossing around the Fermi level that lacks hBN.

Finally, we can acknowledge that regarding the orbital contributions, graphene and hBN are not that different (meaning other physical-chemical properties might be similar). To see this, first look at Figure 7, where we see that the orbital contributions to the density of states follow the same patterns per orbital type. Moreover, by plotting the fat-bands of the band structure as done in Figure 9, we see that the bands that cross in graphene and the analogue ones of hBN are mainly composed of $n = 2$, $l = 1$, $m = 0$ orbitals (the $2p_z$), which are the ones that are commonly attributed to the aromatic rings. Then, the residual contributions are given mainly by the same $n = 3$, $l = 2$, $m = \pm 1$ orbitals in both materials. The rest of the bands as well seem to be composed in relative weight by the same orbitals.

Last but not least, it is interesting to see that if we plot the fat-bands as a function of the atoms in the unit cell, as done in Figure 10, while in graphene the contribution to the levels around the Fermi energy is equally weighted for both carbon atoms, in hBN, the levels with electrons below the Fermi energy, are mainly localized on the N atom (blue), while the unoccupied levels are mainly around the B atom (green). This is in complete accordance with the electronic density plotted in Figure 6, since typically the levels below the Fermi level are the most populated ones at the temperature scales of interest.

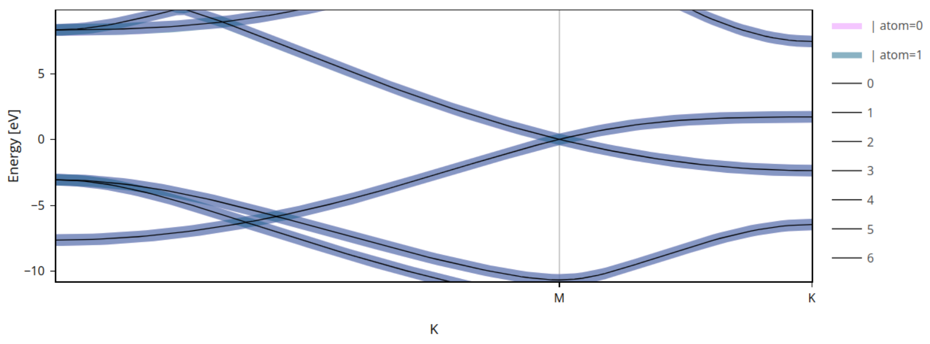


(a) Graphene

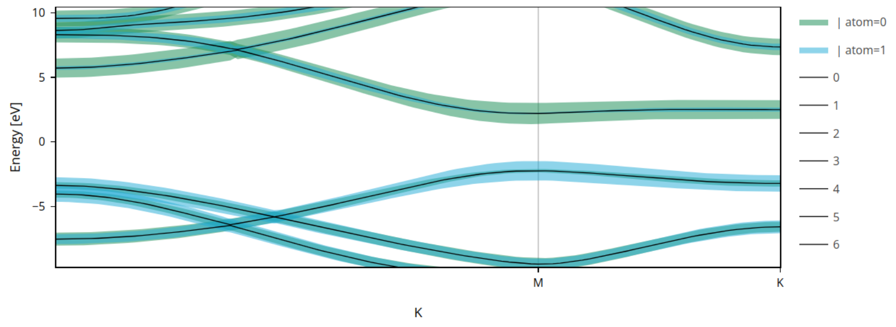


(b) hBN

Figure 9: Band structure in $E \in [-10, 10]$ eV computed with *sisl* employing the Hamiltonian obtained with SIESTA. Fermi level at $E = 0eV$. The colored bands plot the relative contributions by the different orbitals to each state. Note the similarity of the diagrams except for the band crossing around the Fermi level that lacks hBN.



(a) Graphene



(b) hBN

Figure 10: Band structure in $E \in [-10, 10]$ eV computed with *sisl* employing the Hamiltonian obtained with SIESTA. Fermi level at $E = 0eV$. The colored bands plot the relative contributions by each of the atoms in the lattice. In the case of hBN, atom 0 is B and 1 is N.

References

[1] J. M. Soler, E. Artacho, J. D. Gale, A. García, J. Junquera, P. Ordejón, and D. Sánchez-Portal, "The siesta method for ab initio order-n materials simulation," *Journal of Physics: Condensed Matter*, vol. 14, no. 11, p. 2745, 2002.

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

[4] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[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] T. M. Project, "Materials data on bn by materials project," 7 2020.

[8] "Siesta manuals." https://siesta-project.org/SIESTA_MATERIAL/Docs/Manuals/manuals.html.

[9] "Wikipedia entry on the driac cone." https://en.wikipedia.org/wiki/Dirac_cone.