Birkan Emrem

19.11.2025

Density Functional Tight Binding (DFTB) and Applications with DFTB+

# Density Functional Tight Binding (DFTB) and Applications with DFTB+

## Introduction & Theory

- Why do we need methods faster than DFT?
- The trade-off between accuracy vs computational cost
- Where DFTB fits: „between classical force fields and full DFT“

**Ideal for:**

- Large-scale molecular dynamics
- Materials discovery and screening
- Studying reactivity in complex systems

## Theoretical Background — DFTB

Density Functional Tight-Binding (DFTB): - A computationally efficient approximation to DFT - Ideal for large systems (biomolecules, MD simulations)

$$E_{DFTB} = E_{BS} + E_{coul} + E_{rep}$$

- $E_{DFTB}$ –> DFTB total energy
- $E_{BS}$ –> Band structure energy
- $E_{coul}$–> Coulomb interaction
- $E_{rep}$–> Repulsive energy

## Input and Output Files

**Geometry File (`.gen`)**

**Input File (`dftb_in.hsd`)**

**Output Files**

**Goals of This Hands-on**

- Perform 4 types of calculations using 4 different representative systems
- Build and run all simulations from a parent Jupyter notebook
- Visualize and analyze results interactively #### Tools and dependencies
- DFTB+, ASE, NGLView, NumPy, Matplotlib, OS tools

```
import os
from ase import Atoms
from ase.visualize import view
from ase.io import read
import nglview as nv
```

## Geometry Optimization

### Triglucose substrate

```
structure_1 = read("structures/all4-1_0.gen")
w = nv.show_ase(structure_1)
w.clear_representations()
w.add_ball_and_stick(radius=0.3)
w
```

```
NGLWidget()
```

### Create Directory for the Calculations

```
! pwd
```

```
/home/di97fod/Desktop/courses/InteractiveSession
```

```
! mkdir calculations               # create the parent directory
    for the calculations
! mkdir calculations/GO            # create the GO directory for
    the geometry optimzation
```

Navigate to directory for the geometry optimization

```
%cd calculations/GO/
```

```
/home/di97fod/Desktop/courses/InteractiveSession/calculations/GO
```

```
%%writefile dftb_in.hsd

Geometry = GenFormat {
    <<< "../../structures/all4-1_0.gen"
}
Hamiltonian = DFTB {
    SCC = Yes
    ThirdOrderFull = Yes
    HubbardDerivs {
        H = -0.1857
        C = -0.1492
```

```
        O = -0.1575
    }
    Mixer = Broyden {
        MixingParameter = 0.1
    }
    MaxSCCIterations = 300
    SlaterKosterFiles = Type2FileNames {
        Prefix = "../../parameters/3ob-3-1/"
        Separator = "-"
        Suffix = ".skf"
    }
MaxAngularMomentum {
    H = s
    C = p
    O = p
}
}
Driver = GeometryOptimization {
  Optimizer = Rational {}
  MovedAtoms = 1:-1               # Move all atoms in the system
  MaxSteps = 200                  # Stop after maximal 100 steps
  OutputPrefix = "geom.out"       # Final geometry in
      geom.out.{xyz,gen}
  Convergence {GradElem = 0.001}   # Stop if maximal force below
      1E-4 H/a0
}
```

```
Writing dftb_in.hsd
```

```
# Run calculation
! dftb+ > triglucose.log
```

check the directory

```
! ls
```

```
band.out     detailed.out  dftb_pin.hsd  geom.out.xyz
charges.bin  dftb_in.hsd   geom.out.gen  triglucose.log
```

control the log file

```
! tail triglucose.log
```

```
DFTB+ running times                            cpu [s]
    wall clock [s]
-------------------------------------------------------------------------------
```

```
Global initialisation                  +       0.09 (  0.0%)
          0.00 (  0.0%)
Pre-SCC initialisation                 +       8.48 (  1.8%)
          0.38 (  1.8%)
SCC                                    +     414.90 ( 88.9%)
         19.17 ( 89.0%)
Post-SCC processing                    +      43.13 (  9.2%)
          1.97 (  9.2%)
-------------------------------------------------------------------------------
Missing                                +       0.17 (  0.0%)
          0.00 (  0.0%)
Total                                  =     466.76 (100.0%)
         21.53 (100.0%)
-------------------------------------------------------------------------------
```

extract total energies and save into `energies.txt`

```
! grep "Total Energy" triglucose.log | awk '{print $3}' >
    energies.txt
```

control the `energies.txt` file

```
! head energies.txt
```

```
-87.9747853308
-88.1581573048
-88.3235456743
-88.4316353471
-88.5013513975
-88.5482347574
-88.5625706362
-88.5789474435
-88.5968777061
-88.6102477586
```

plot the total and relative energy with respect to geometry steps

```
import matplotlib.pyplot as plt
import pandas as pd

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5))

# Read the log file
with open("triglucose.log", "r") as f:
    lines = f.readlines()

#energy values
```

```
energies = pd.read_csv("energies.txt", header=None)
energy_diff = [i - min(energies[0]) for i in energies[0]]


# Plotting
ax1.plot(range(1, len(energies) + 1), energies, marker='*',
    color='crimson', markersize=10, linestyle='--')
ax1.set_xlabel("Geometry Step")
ax1.set_ylabel("Total Energy (Hartree)", fontsize=14)
ax1.set_title("Total Energy")

ax2.plot(range(1, len(energies) + 1), energy_diff, marker='o',
    color='goldenrod', markersize=10, linestyle='--')
ax2.set_xlabel("Geometry Step")
ax2.set_ylabel("Relative Energy (Hartree)", fontsize=14)
ax2.set_title("Relative Energy")

_ = fig.suptitle("Energy Convergence During Geometry
    Optimization")
```

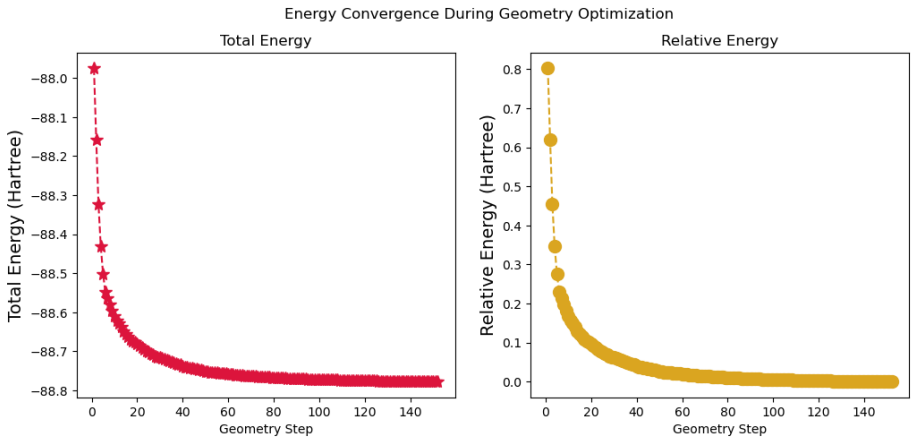


Figure 1: png

Now let's look at how the optimized geometry looks

```
structure_1_opt = read("geom.out.gen")
w2 = nv.show_ase(structure_1_opt)
w2.clear_representations()
w2.add_ball_and_stick(radius=0.3)
w2
```

```
NGLWidget()
```

## HOMO-LUMO GAP

### Triangular Finite $Mo_aS_b$ Nanoparticle

In this section we are going to investigate finite $Mo_{28}S_{72}$ nanoparticle.

```
structure_2 = read("../../structures/MoS2_triangle.gen")
w = nv.show_ase(structure_2)
w.clear_representations()
w.add_ball_and_stick(radius=0.5)
w
```

```
NGLWidget()
```

first lets go back to main directory

```
%cd ../..
```

```
/home/di97fod/Desktop/courses/InteractiveSession
```

```
! ls
```

```
calculations  dftb.ipynb  dos.py  Figures  parameters  structures
```

and let's create a directory named `homo_lumo_gap` where we will perform this calculation

```
! mkdir calculations/homo_lumo_gap     # create the
    homo-lumo-gap directory
```

```
%cd calculations/homo_lumo_gap
```

```
/home/di97fod/Desktop/courses/InteractiveSession/calculations/homo_lumo_gap
```

now we write our input file and save as `dftb_in.hsd`

```
%%writefile dftb_in.hsd

Geometry = GenFormat {
    <<< "../../structures/MoS2_triangle.gen"
}

Hamiltonian = DFTB {
    SCC = Yes
    SCCTolerance = 1e-5
    MaxSCCIterations = 1000
    SlaterKosterFiles = Type2FileNames {
        Prefix = "../../parameters/ptbp/"
        Separator = "-"
```

```
        Suffix = ".skf"
    }
MaxAngularMomentum {
    Mo = "d"
    S = "p"
}
Charge = 0
Filling = Fermi {
    Temperature [Kelvin] = 300
}
}

Analysis = {
    WriteEigenvectors = Yes
}
Options {
    WriteDetailedXml = Yes
    WriteResultsTag = Yes
}

ParserOptions {
    ParserVersion = 14
}
```

```
Writing dftb_in.hsd
```

We run the calculation

```
! dftb+ > MoS2_triangle.log
```

Now we use simple python script to extract HOMO, LUMO and Gap values in eV

```
homo = None
lumo = None

with open("band.out") as f:
    for line in f:
        parts = line.strip().split()
        if len(parts) != 3:
            continue # skip bad lines
        ev = float(parts[1])
        occ = float(parts[2])
        if occ > 1e-3:
            homo = ev # keep updating until last occupied
        elif lumo is None:
            lumo = ev # first unoccupied eigenvalue
```

```

if homo is not None and lumo is not None:
    print(f"HOMO: {homo:.3f} eV")
    print(f"LUMO: {lumo:.3f} eV")
    print(f"Gap : {lumo - homo:.3f} eV")
else:
    print("Could not find HOMO or LUMO.")
```

```
HOMO: -6.155 eV
LUMO: -5.551 eV
Gap : 0.604 eV
```

## Density of States and Bandstructure

### Graphene: 4x4x1 Supercell

In this section we are going to investigate density of states and electronic band structure of 4x4 supercell graphene monolayer Let's first look how our monolayer looks

```
structure_3 = read("../../structures/graphene_441.gen")
w = nv.show_ase(structure_3)
w.clear_representations()
w.add_ball_and_stick(radius=0.35)
w
```

```
NGLWidget()
```

we go one directory back and create a directory for this section

```
%cd ..
```

```
/home/di97fod/Desktop/courses/InteractiveSession/calculations
```

```
! mkdir dos_bs
```

```
%cd dos_bs/
```

```
/home/di97fod/Desktop/courses/InteractiveSession/calculations/dos_bs
```

Firstly we are going to optimize the monolayer, and let's create a directory for that

```
!mkdir GO
%cd GO
```

```
/home/di97fod/Desktop/courses/InteractiveSession/calculations/dos_bs/GO
```

```
%%writefile dftb_in.hsd

Geometry = GenFormat {
    <<< "../../../structures/graphene_441.gen"
}

Hamiltonian = DFTB {
    SCC = Yes
    MaxSCCIterations = 300
    SlaterKosterFiles = Type2FileNames {
        Prefix = "../../../parameters/mio-1-1/"
        Separator = "-"
        Suffix = ".skf"
    }
MaxAngularMomentum {
    C = "p"
}
KPointsAndWeights = SupercellFolding {
  24 0 0
  0 24 0
  0 0 1
  0.0 0.0 0.0
}
Filling = Fermi {
  Temperature [Kelvin] = 100
}
}

Driver = GeometryOptimization {
  Optimizer = Rational {}
  LatticeOpt = Yes
  Isotropic = Yes
  MovedAtoms = 1:-1                # Move all atoms in the system
  MaxSteps = 200                   # Stop after maximal 100 steps
  OutputPrefix = "geom.out"        # Final geometry in
      geom.out.{xyz,gen}
  Convergence {GradElem = 0.001}   # Stop if maximal force below
      1E-4 H/a0
}
```

```
Writing dftb_in.hsd
```

```
! dftb+ > graphene_opt.log
```

now since our geometry is optimized we can make DOS calculation. For that purpose, we create a subdirectory called dos

```
! mkdir dos
%cd dos
```

```
/home/di97fod/Desktop/courses/InteractiveSession/calculations/dos_bs/GO/dos
```

For DOS calculation, we are going to use the optimized geometry. Let's copy it to that folder.

```
! cp ../geom.out.gen ./opt.gen
```

Now we create our input file for the DOS calculation.

```
%%writefile dftb_in.hsd

Geometry = GenFormat {
    <<< "opt.gen"
}

Hamiltonian = DFTB {
  Scc = Yes
  SccTolerance = 1e-5
  SlaterKosterFiles = Type2FileNames {
    Prefix = "../../../../parameters/mio-1-1/"
    Separator = "-"
    Suffix = ".skf"
  }
MaxAngularMomentum {
    C = "p"
}
  KPointsAndWeights = SupercellFolding {
    48 0 0
    0 48 0
    0 0  1
    0 0 0
  }
}

Analysis {
  ProjectStates {
    Region {
      Atoms = C
      ShellResolved = Yes
      Label = "dos_C"
    }
  }
}

```

```
ParserOptions {
  ParserVersion = 12
}
```

```
Writing dftb_in.hsd
```

And we run

```
! dftb+ > graphene_dos.log
```

In order to visualize the density of states and the partial density of states, we should convert the corresponding human readable files (with prefix .out) to XY-format data

```
%%bash

dp_dos band.out dos_total.dat
dp_dos -w dos_C.1.out dos_C.1.dat
dp_dos -w dos_C.2.out dos_C.2.dat
```

now we copy `dos.py` from parent folder to here to use it to plot DOS of graphene.

```
! cp ../../../../dos.py .
```

```
from dos import plot_dos
```

```
plot_dos(".", show_total_only=False)
```

```
<Figure size 640x480 with 0 Axes>
```

Now we are going to continue for the electronic band structure calculation. First lets create a directory for the band structure calculation.

```
! mkdir bs
%cd bs
```

```
/home/di97fod/Desktop/courses/InteractiveSession/calculations/dos_bs/GO/dos/bs
```

For the electronic band structure calculation, we need `charges.bin` and optimized geometry file

```
! cp ../charges.bin .
! cp ../opt.gen .
```

```
%%writefile dftb_in.hsd

Geometry = GenFormat {
    <<< "opt.gen"
}
```

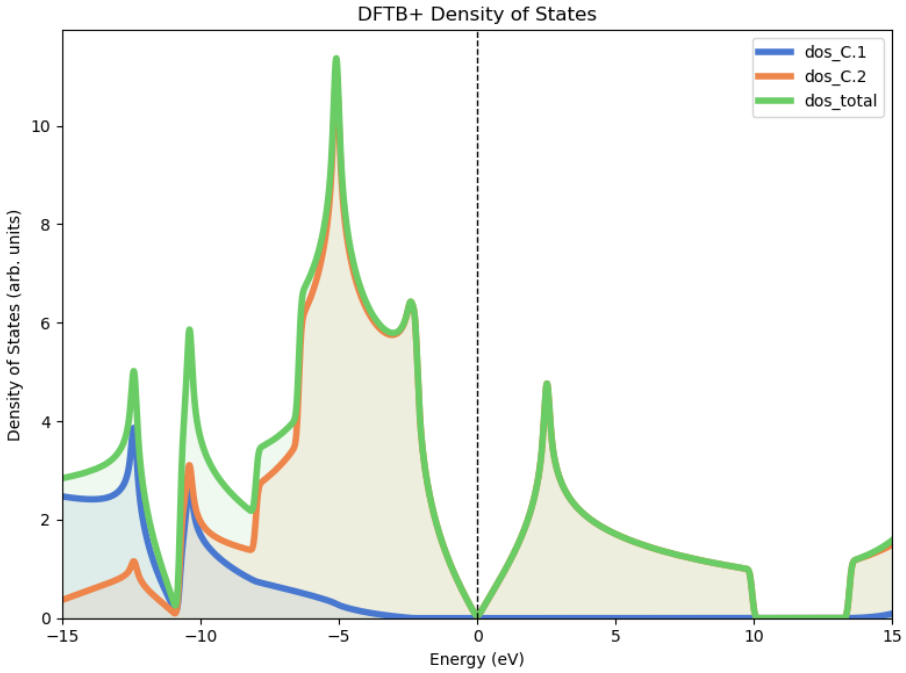
DFTB+ Density of States
dos_C.1
dos_C.2
dos_total
Density of States (arb. units)
Energy (eV)
0
2
4
6
8
10
−15
−10
−5
0
5
10
15

Figure 2: png

```

Hamiltonian = DFTB {
  Scc = Yes
  ReadInitialCharges = Yes
  MaxSCCIterations = 1
  SccTolerance = 1e-5
  SlaterKosterFiles = Type2FileNames {
    Prefix = "../../../../../parameters/mio-1-1/"
    Separator = "-"
    Suffix = ".skf"
  }
MaxAngularMomentum {
    C = "p"
}
KPointsAndWeights = KLines {
 1    0.0  0.0  0.0                  # Gamma
 15    0.33333333  0.66666666 0.0    # K
 15    0.5  0.0  0.0                  # M
 15    0.0  0.0  0.0                  # Gamma
}
}


ParserOptions {
  ParserVersion = 12
}
```

```
Writing dftb_in.hsd
```

Let's run the calculation

```
! dftb+ > graphene_bs.log
```

We again convert `band.out` into `.dat` file in order to post process

```
! dp_bands band.out band
```

And now let's plot the band structure of 4x4 monolayer graphene at path Gamma-K-M-Gamma

```
import numpy as np
import matplotlib.pyplot as plt

data = np.loadtxt("band_tot.dat")
k_points = data[:, 0]
bands = data[:, 1:] + 4.67

```

```
fig, ax = plt.subplots(1, 1, figsize=(8, 6))

for i in range(bands.shape[1]):
    ax.plot(k_points, bands[:, i], color='darkblue', linewidth=2)

ax.axhline(0, color="crimson", linestyle="--", linewidth=2)

for i in [16, 31]:
    plt.plot([i, i], [-3, 3], color='gray', linestyle='--')

ax.set_ylim(-3, 3)
ax.set_xlim(1, 46)

_ = ax.set_xticks([1, 16, 31, 46])
_ = ax.set_xticklabels(['\Gamma', 'K', 'M', '\Gamma'],
    fontsize=13)

_ = ax.set_ylabel('Energy (eV)', fontsize=16)
```

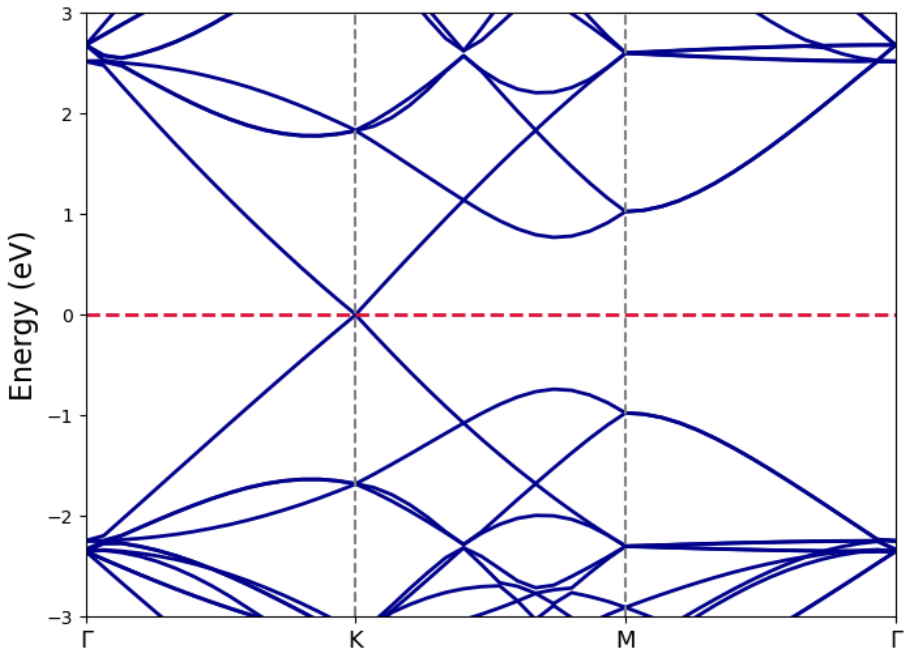


Figure 3: png