

PROCEDURE OVERVIEW

We will perform any non-adiabatic calculation in four steps:

1. Ground State Optimization
2. Ground State Born-Oppenheimer Molecular Dynamics (Sample Nuclear Phase Space)
3. Calculate the Charge-Localized Wavefunction (Initial Electronic Condition for each Nuclear Geometry)
4. Non-adiabatic Molecular Dynamics

INSTALL DFTB+ AND MODIFIED DFTB+

If you already have DFTB+ installed locally or via the cluster through a module, then you do not need to install it again. We can simply point to that installation when we start running the codes.

First, we need to install both the non-modified DFTB+ package.

I suggest a simple directory structure like or something similar:

DFTB/

DFTB/ORIGINAL

DFTB/MODIFIED_CT

You may need some intel compilers to be loaded. For me, I use our cluster intel through “module load intel” which loads “intel/2020.4”

Something like:

```
$ cd DFTB/ORIGINAL
```

```
$ git clone https://github.com/dftbplus/dftbplus
```

Follow installation instructions found in “INSTALL.rst”.

Something like:

```
$ make -j4
```

Similarly, for the modified version.

Something like:

```
$ cd DFTB/MODIFIED_CT
```

```
$ git clone https://github.com/bradenmweight/DFTBplus_Charge_Transfer
```

Probably install the same way as you did for the unmodified version.

Something like:

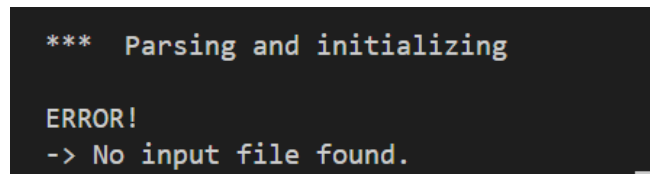
```
$ make -j4
```

TEST BOTH DFTB+ INSTALLATIONS

DFTB/ORIGINAL/_install/bin/dftb+ (or via module on the cluster)

DFTB/MODIFIED_CT/_install/bin/dftb+

If you run the executable for each (in a blank new folder), you should hopefully be able to recover to following error (see Figure 1 below). This means that the DFTB+ distribution is working correctly to some extent. 😊 Different versions of DFTB+ will give slightly different errors here, but as long as it says that it could not find an input file, you should be good.

A terminal window with a dark background and light-colored text. The text reads:

```
*** Parsing and initializing  
  
ERROR!  
-> No input file found.
```

Figure 1: Run the executable “DFTB/ORIGINAL/_install/bin/dftb+”

If you instead find the following error:

.../DFTB/ORIGINAL/_install/bin/dftb+: error while loading shared libraries: libmkl_intel_lp64.so: cannot open shared object file: No such file or directory

You will need to make sure that the intel library is correctly loaded.

EXAMPLE CALCULATION – TWO FULLERENES

I have uploaded a new folder called “example” in my modified FSSH_DFTB+ repository, which simulates the charge transfer dynamics between two fullerene molecules (C60/C60).

1. Optimize Geometry

Note: This is probably not necessary if you already performed static calculations with a different electronic structure software, i.e., Gaussian16, Q-CHEM, etc.

Put initial geometry in the 1_GS_OPT folder as geometry_input.xyz in the standard XYZ format. The submission script is located in Template/opt.sbatch.tem. Modify as needed; make sure the number of cores in optimize.py and the sbatch file are the same.

To run the optimization:

```
$ python3 optimize.py geometry_input.xyz
```

IMPORTANT NOTE: Put the intended DONOR species at the beginning of the XYZ file.

2. Canonical Ensemble (*i.e.*, NVT) Dynamics

At fixed temperature (according to the Langevin thermostat), we perform picoseconds of dynamics on the ground state potential energy surface such that we can sample the nuclear coordinates and momenta at regular intervals to use as initial conditions for excited state dynamics. Modify parameters in nvt.py as needed. For publishable data, probably run 50 ps of dynamics (simTime) or so at 1 fs timestep (timeStep) for 1000 trajectories (grabGeometries) sampled every 50 fs (simTime / grabGeometries). The batch script is located at Template/nvt.sbatch.tem. This job may take some time if you have a large system.

To run the NVT dynamics:

```
$ python3 nvt.py ../1_GS_OPT/optimized.xyz (or whatever file you want to start with)
```

3. Create initial, charge-localized wavefunctions

For each initial nuclear conditions (sampled from NVT dynamics), we will compose an initial wavefunction that is localized to one part of the system. This condition is chosen to be one of the conduction orbitals for the DONOR species. For example, if we choose to use the LUMO of the DONOR species, then we simply get the expansion coefficients of the LUMO from DONOR in the basis of the SYSTEM molecular orbitals. This script is overly complicated and should be split into more than one step, but it does work. The script “psiEigen.py” is the main workhorse here. “step” = “grabGeometries” from NVT job. Here, only use a single core for this job since many short jobs will be submitted during the process. “LUMO” indicates which conduction orbital on the DONOR species is to be chosen as the initial electronic condition. LUMO = 0 is the LUMO of the DONOR. LUMO = 1 is LUMO+1 on the DONOR, etc. “donorAtoms” indicates the range of the DONOR atoms. The DONOR needs to be the first chunk of atoms in the XYZ scripts, as noted in the first step; else, everything will go wrong here. “yourId” should be your cluster’s user ID, usually the same as what is printed with “echo \$USER” at the command line. There are two batch scripts here: Template/createPsi.sbatch.tem and Template/dftb.sbatch.tem. Modify as needed. Everything should only need one core. For very large systems, each main job may take up to ~3 hours, which is controlled by

dftb.sbatch.tem. createPsi.sbatch.tem should be given a while as well, maybe ~3+ hours, too, if all main jobs start right away.

To run:

```
$ python3 psiEigen.py
```

4. Non-Adiabatic Molecular Dynamics (NAMD)

Here, we perform the NAMD calculations for each initial condition in the classical path approximation (CPA) where the nuclei are moving only according to the ground state forces and do not respond to the motion of the electronic system. namd.py is the main script here. “cores” should be set to 1 since we intend to run a number of trajectories simultaneously. “nuclearTraj” = “grabGeometries” from the NVT dynamics step. “timeStep” should probably be less than 1.0 fs to get accurate electronic dynamics. “simTime” is however long you want to run the non-adiabatic dynamics in fs units. “elecSteps” controls the number of electronic propagation steps per nuclear timestep. “activeSpace” controls the number of conduction orbitals (of the total system) included in the non-adiabatic dynamics. For example, if LUMO = 0 in the previous step, probably activeSpace = 10 is good enough. If LUMO = 10 in the previous step, then maybe activeSpace = 100 may be needed. To check this, you can explicitly look in the Psi.txt files created in the previous step to see where the large contributions to the initial wavefunction are in the system’s MO basis. At the top of namd.py, be sure to change the location of your modified dftb+ code executable file. The batch file is Template/namd.sbatch.tem. Modify as needed.

To run the NAMD jobs:

```
$ python3 namd.py
```

To acquire the results, run:

```
$ python3 get_average_DFTBplus.py
```

“adiabatic_pop.txt” reports the adiabatic MO populations over time.

“estimator{1,2,3}.txt” reports the charge local population on the two fragments via three different methods of rotating the adiabatic population to the diabatic one. Probably “estimator3.txt” is the best one. See our recent paper on the FSSH QD for polaritons. [X] We explain this in detail in Appendix C.

[X] Hu, Mandal, Weight, and Huo, J. Chem. Phys. 157, 194109 (2022)