



Demo and hands on with pyUNixMD + DFTB

Daeho Han
July 11, 2022

CONTACT

Excited State Phenomena Computational Chemistry Lab.

Office/Lab. 108) 805 **Tel.** +82 52 217 2918

Web. <http://skmin.unist.ac.kr/>

Setting

- Installation of pyUNIXMD

Go to your project directory, and git clone pyUNIXMD package.

```
$ cd /your-project-directory/  
$ git clone https://github.com/skmin-lab/unixmd.git
```

Activate pyunixmd conda environment.

```
$ conda activate pyunixmd
```

Time-consuming calculations such as electronic evolution in PyUNIXMD are done with C functions through Cython package. Thus, we need to compile C functions.

```
$ cd /your-project-directory/unixmd/  
$ python setup.py build_ext -b ./src/build
```

Setting

● Tutorial materials

Clone the hands-on materials by


```
$ git clone https://github.com/compchem-cybertraining/Tutorials\_pyUNIXMD.git
```

Get back to your directory and copy an environment setting (you may need to change PYTHONPATH variable in the file) and source script for using PyUNIXMD and DFTB+

```
$ cd ..  
$ cp /projects/academic/cyberwksp21/Students/dhan/set_pyunixmd_dftb.sh .  
$ source set_pyunixmd_dftb.sh
```

set_pyunixmd+dftb.sh

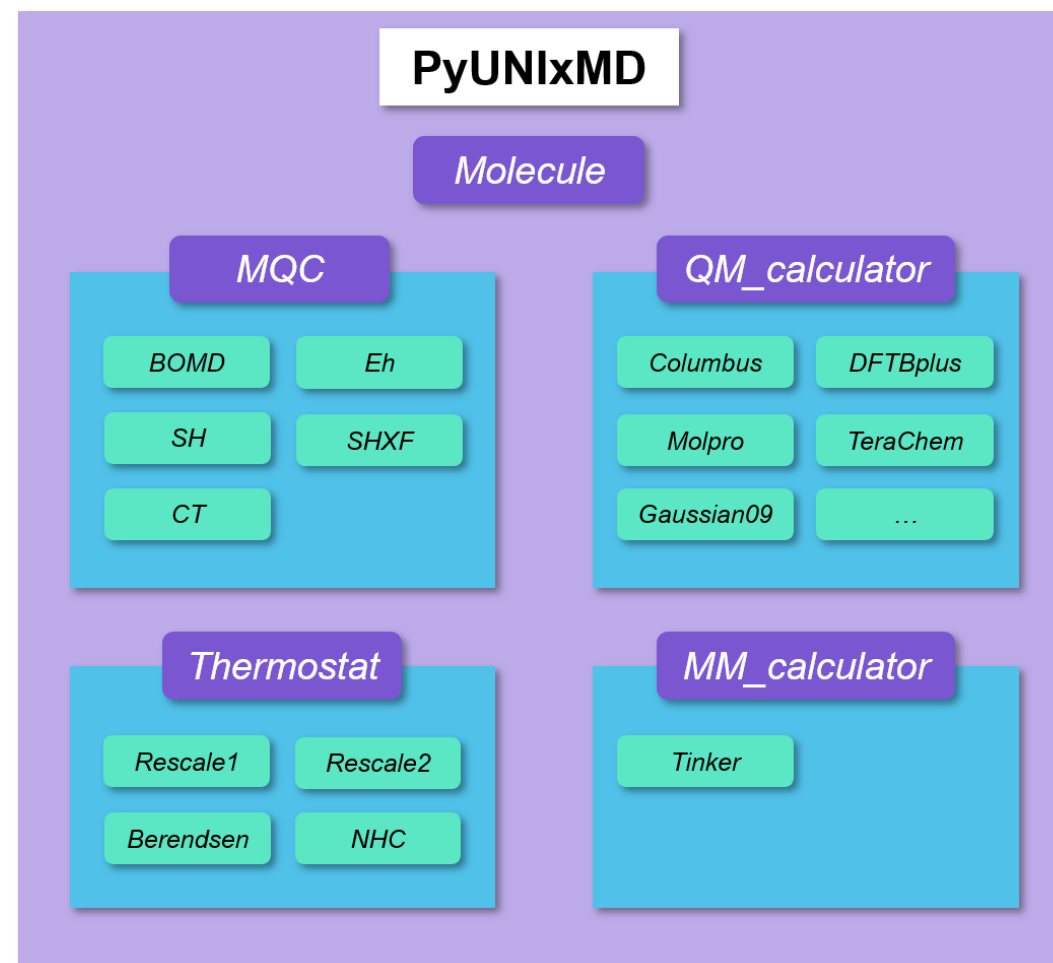
```
#!/bin/sh  
#pyUNIXMD  
export  
PYTHONPATH=$PYTHONPATH:/projects/academic/cyberwksp21/Students/$USER/unixmd/src  
  
#DFTB+  
module load intel/20.2  
module load intel-mpi/2020.2  
export I_MPI_PMI_LIBRARY=/usr/lib64/libpmi.so  
  
source /util/academic/intel/20.2/compilers_and_libraries_2020.2.254/linux/bin/compilervars.sh  
intel64  
source  
/util/academic/intel/20.2/compilers_and_libraries_2020.2.254/linux/mpi/intel64/bin/mpivars.sh
```

Change this if necessary. 

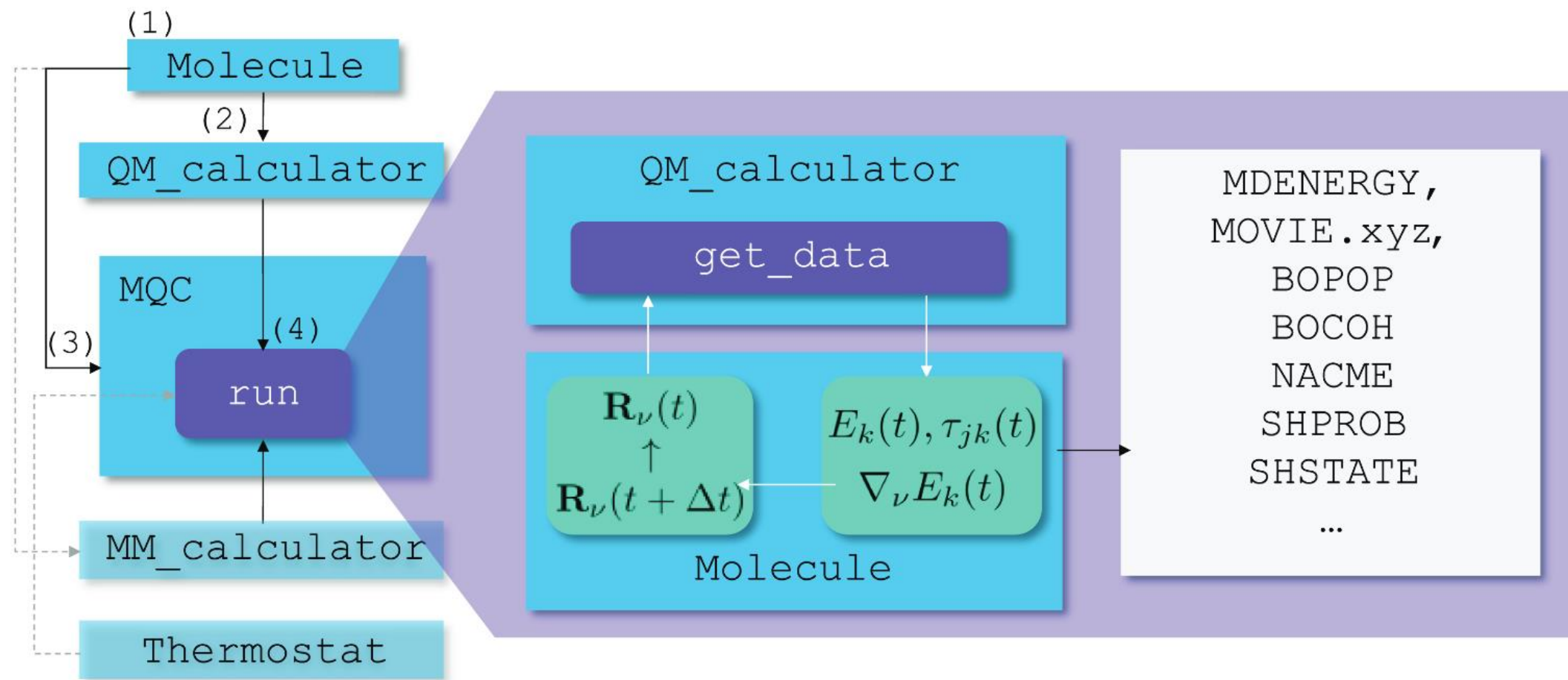
- pyUNIxMD is an object-oriented Python program for molecular dynamics simulations involving multiple electronic states. It is mainly for studying the nonadiabatic dynamics of excited molecules.
- Each ingredient of nonadiabatic dynamics is treated as a Class, which makes it easy to manage and analyze the MD quantities.



J. Comput. Chem. **2021**, 42, 1755-1766.



- A general workflow of a NAMD with pyUNixMD. Each object exchanges the data in NAMD.



- pyUNIxMD gives an NAMD interface with a lot of electronic structure programs.

Program	Method	BOMD	Ehrenfest	FSSH	SHXF	CTMQC
Columbus	CASSCF	✓	✓	✓	✓	✓
	MRCL	✓	✓	✓	✓	✓
DFTB+	TDDFTB	✓	-	✓	✓	-
	DFTB/SSR	✓	✓	✓	✓	✓
Gaussian 09	TDDFT	✓	-	✓	✓	-
Molpro	CASSCF	✓	✓	✓	✓	✓
Q-Chem	TDDFT	✓	✓	✓	✓	✓
TeraChem	SSR	✓	✓	✓	✓	✓

- DFTB+ is an open-source software package based on density functional theory tight binding (DFTB).
- This is a semi-empirical method where the KS Hamiltonian matrix represented by a minimal atomic basis and the total energy elements are parametrized by Slater-Koster parameters. This allows long-timescale simulations of large systems with reasonable accuracy while having faster computational time than usual ab initio methods.

$$\sum_{\nu}^M c_{\nu i} (H_{\mu\nu} - \varepsilon_i S_{\mu\nu}) = 0, \quad \forall \mu, i$$

$$H_{\mu\nu} = \langle \varphi_{\mu} | \hat{H}_0 | \varphi_{\nu} \rangle + \frac{1}{2} S_{\mu\nu} \sum_{\xi}^N (\gamma_{\alpha\xi} + \gamma_{\beta\xi}) \Delta q_{\xi}$$

$$= H_{\mu\nu}^0 + H_{\mu\nu}^1, \quad S_{\mu\nu} = \langle \varphi_{\mu} | \varphi_{\nu} \rangle, \quad \forall \mu \in \alpha, \quad \nu \in \beta$$



To solve a secular equation from Kohn-Sham DFT, each ingredient is parameterized with ab initio calculation of reference systems and is given a function of atomic distances.

Phys. Rev. B **1998** 58, 7260.

J. Chem. Phys. **2020**, 152, 124101.

- You can obtain Slater-Koster parameter files in <https://dftb.org/parameters/download>. Parameters used in this tutorial are already in the tutorial directory.

`/your-project-directory/Tutorials_pyunixmd/demo_pyunixmd_dftb/mio-1-1`
`/your-project-directory/Tutorials_pyunixmd/demo_pyunixmd_dftb/ob2-1-1/base`

dftb.org the DFTB website

- About DFTB
- Codes
- Parameters
 - Introduction
 - Download
- Contact
- Sitemap

Download of Slater-Koster files

Below you find the tables with the actual versions of the available Slater-Koster files. The first column of each table contains the name of the set, which should be used, when referring to it. The third column contains the list of the elements in that set. Hyphenation of elements (A-B-C) means that all possible combinations of SK files within this set are available in the repository, although not necessarily under the same tag. In this case the second column (required field) shows the location of the rest. The last column contains a short description of the set. For a more detailed description and for download click on the name of the set.

Please note, that unless explicitly indicated, the **sets are usually not compatible** to each other. Therefore, the parameter files from two sets should not be mixed (although your program may not even notice the incompatibility of the files used).

For you convenience, you may also download all publicly available SK-sets together within **one archive**.

General purpose parameter sets

These sets were created to have a good performance over a wide range of elements and over a wide range of applications. They have been thoroughly tested for a broad range of systems. However, **if you intend to use them for systems very different from those included in the tests, please evaluate the sets first.**

Name	Requires	Elements	Short description
3ob		Br-C-Ca-Cl-F-H-I-K-Mg-N-Na-O-P-S-Zn	DFTB3 files for bio and organic molecules
matsci		Al-O-H Al-Si-O-H Cu-Si-Al-Na-O-H Ti-P-O-N-C-H O-N-C-B-H Al-O-C-H Si-P-N-O-C-H	Collection of some sets used for various problems in materials science. <i>(For some of the interactions no published tests available!)</i>
mio		H - C - N - O - S - P	SCC files for bio or organic molecules
ob2		H-C-N-O	Long range corrected parameterization for bio and organic molecules
pbc		Si - F - O - N - C - H Fe	SCC files for solids and surfaces

- DFTB+ provides with excited-state calculations: TD-DFTB and REKS(spin-Restricted Ensemble Kohn-Sham)-DFTB.
- In the TD-DFTB implementation, DFTB+ uses ground-state SCF calculation as reference and solve the Casida equation constructed from parametrized variables.

Phys. Rev. B **2001**, 63, 085108.

$$\sum_{ij\sigma} [\omega_{ij}^2 \delta_{ik} \delta_{jl} \delta_{\sigma\tau} + 2 \sqrt{\omega_{ij}} K_{ij\sigma,kl\tau} \sqrt{\omega_{kl}}] F_{ij\sigma}^I = \omega_I^2 F_{kl\tau}^I$$

The response matrix is described by the reference DFTB calculations.

- The TD-DFTB implementation of DFTB+ does not provide with nonadiabatic coupling vectors (NACVs) or nonadiabatic coupling matrix elements (NACMEs), so we calculate them by evaluating wave function overlap.

$$\dot{C}_n = -\frac{i}{\hbar} E_n C_n - \sum_k \tau_{nk} C_k \quad \tau_{nk} = \langle \Psi_n | \partial_t | \Psi_k \rangle \quad | \Psi_n \rangle = \sum_{ia} C_{ia}^n | \Phi_i^a \rangle$$

$$\tau_{nk} = \sum_{ia} C_{ia}^n \partial_t C_{ia}^k + \sum_{iab} C_{ia}^n C_{ib}^k \langle \phi_a | \partial_t \phi_b \rangle - \sum_{ija} P_{ij} C_{ia}^n C_{ja}^k \langle \phi_j | \partial_t \phi_i \rangle$$

C_{ia}^n 's are excitation coefficients of CIS-like expansion in TDDFT.

J. Phys. Chem. Lett. **2015**, 6, 21, 4200–4203.

- In PyUNixMD, doubled molecule technique is used to obtain wavefunction overlap.

$$\tau_{nk} = \sum_{ia} C_{ia}^n \partial_t C_{ia}^k + \sum_{iab} C_{ia}^n C_{ib}^k \langle \phi_a | \partial_t \phi_b \rangle - \sum_{ija} P_{ij} C_{ia}^n C_{ja}^k \langle \phi_j | \partial_t \phi_i \rangle$$

$$\langle \phi_j | \partial_t \phi_i \rangle \approx \left\langle \phi_j(t) \left| \frac{\phi_i(t + \Delta t) - \phi_i(t - \Delta t)}{2\Delta t} \right. \right\rangle$$

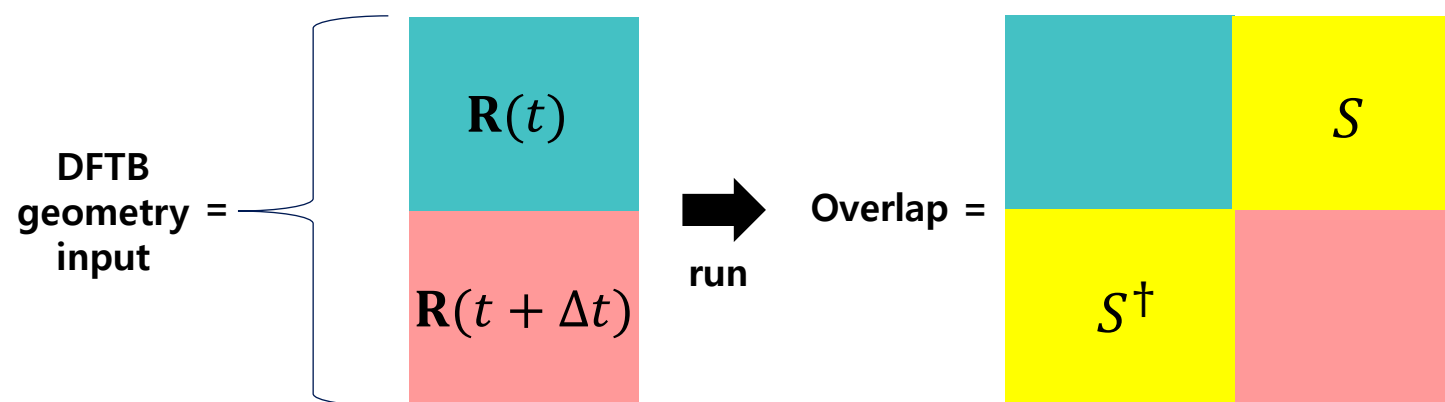
$$= \frac{1}{2\Delta t} (\langle \phi_j(t) | \phi_i(t + \Delta t) \rangle - \langle \phi_j(t) | \phi_i(t - \Delta t) \rangle)$$

$$\approx \frac{1}{2\Delta t} (\langle \phi_j(t) | \phi_i(t + \Delta t) \rangle - \langle \phi_j(t + \Delta t) | \phi_i(t) \rangle)$$

$$|\phi_i\rangle = \sum_{\mu} c_{\mu i} |\chi_{\mu}\rangle$$

$$\langle \phi_j(t) | \phi_i(t + \Delta t) \rangle = \vec{c}_j^{\dagger} S \vec{c}_i$$

$$S_{\mu\nu} = \langle \chi_{\mu}(t) | \chi_{\nu}(t + \Delta t) \rangle$$



- After DFTB+ v20.2, overlap calculation in DFTB+ checks atomic distances of inputs and halt if there is a certain pair too close each other. Thus, this checking routine is needed to be commented out if you want to perform wave function overlap calculation.

```

...
! Sort neighbours for all atom by distance
allocate(indx(maxNeighbour))
do iAtom1 = 1, nAtom
  nn1 = neigh%nNeighbour(iAtom1)
  call index_heap_sort(indx(1:nn1), neigh%neighDist2(1:nn1, iAtom1), tolSameDist2)
  neigh%iNeighbour(1:nn1, iAtom1) = neigh%iNeighbour(indx(:nn1), iAtom1)
  neigh%neighDist2(1:nn1, iAtom1) = neigh%neighDist2(indx(:nn1), iAtom1)
end do

```

```

call reallocateArrays1(img2CentCell, iCellVec, coord, nAllAtom)

```

```

! check for atoms on top of each other
do iAtom1 = 1, nAtom
  do nn1 = 1, neigh%nNeighbour(iAtom1)
    if (neigh%neighDist2(nn1, iAtom1) < minNeighDist) then
      iAtom2 = img2CentCell(neigh%iNeighbour(nn1, iAtom1))
      write (strError, "(A,I0,A,I0,A)") "Atoms ",iAtom1, " and ", iAtom2, " too close together"
      call error(strError)
    end if
  end do
end do

```

```

end subroutine updateNeighbourList

```

```

...

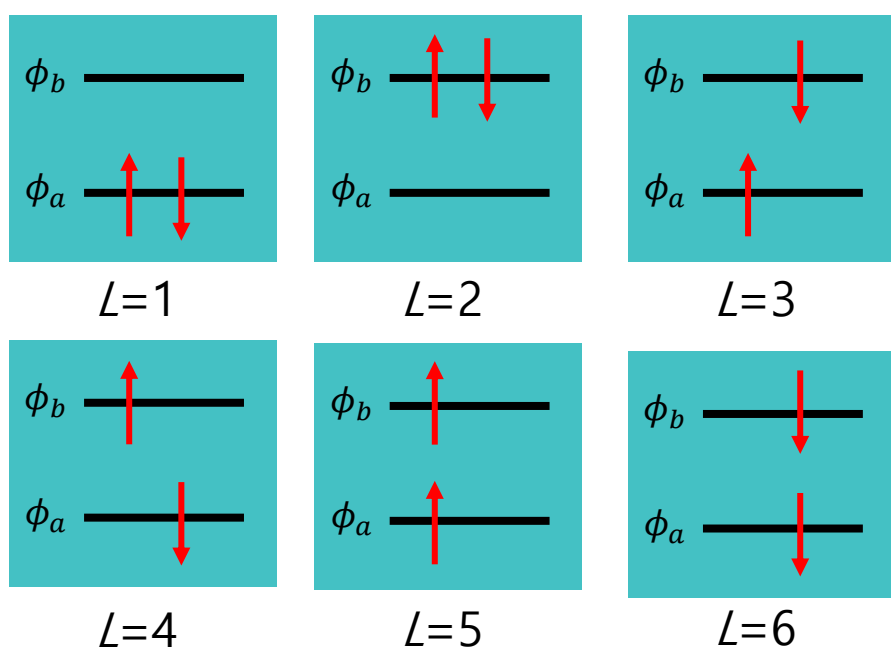
```

[dftbplus-20.2.1/prog/dftb+/lib_dftb/periodic.F90](https://github.com/dftbplus/dftbplus-20.2.1/prog/dftb+/lib_dftb/periodic.F90)

The DFTB binary used in this tutorial is compiled after commenting out this part.

- Instead of single determinants, the spin-restricted ensemble-referenced Kohn–Sham (REKS) method and its state-interaction state-averaged variant (SISA-REKS or SSR) are based on ensemble density functional theory. The semi-empirical version of SSR(2,2) is implemented in DFTB+.

- The energy of the multireference state is expanded in terms of 6 microstates with fixed integer occupations of the active orbitals ϕ_a and ϕ_b .



SA(2,2)-REKS

Perfectly spin-paired singlet state

$$E^{PPS} = \sum_L C_L^{PPS} E_L[\rho_L]$$

Open-shell singlet state

$$E^{OSS} = \sum_L C_L^{OSS} E_L[\rho_L]$$

Optimize $E^{SA} = w_0 E^{PPS} + w_1 E^{OSS}$
with respect to the coefficients and the orbitals.

SSR(2,2)-REKS

Solve the 2x2 secular equation to obtain fully adiabatic solutions.

$$\begin{pmatrix} E^{PPS} & \Delta^{SA} \\ \Delta^{SA} & E^{OSS} \end{pmatrix} \begin{pmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{pmatrix} = \begin{pmatrix} E_0^{SSR} & 0 \\ 0 & E_1^{SSR} \end{pmatrix} \begin{pmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{pmatrix}$$

- Forces and NACVs are evaluated analytically by the coupled-perturbed equation.

$$\mathbf{A}\mathbf{U}^\lambda = \mathbf{B}^\lambda \quad \sum_{k < l} A_{ij,kl} U_{kl}^\lambda = B_{ij}^\lambda \quad \frac{\partial}{\partial \lambda} C_{\mu i} = \sum_j C_{\mu j} U'_{ji}^\lambda \quad U'_{ji}^\lambda = -\frac{1}{2} \frac{\partial' S_{ji}}{\partial \lambda} + U_{ji}^\lambda$$

Tutorial #1: Thermal sampling of target molecules: CNH_4^+ , PSB3

Tutorial #2: SH(EDC) and SHXF dynamics of CNH_4^+ with TD-DFTB

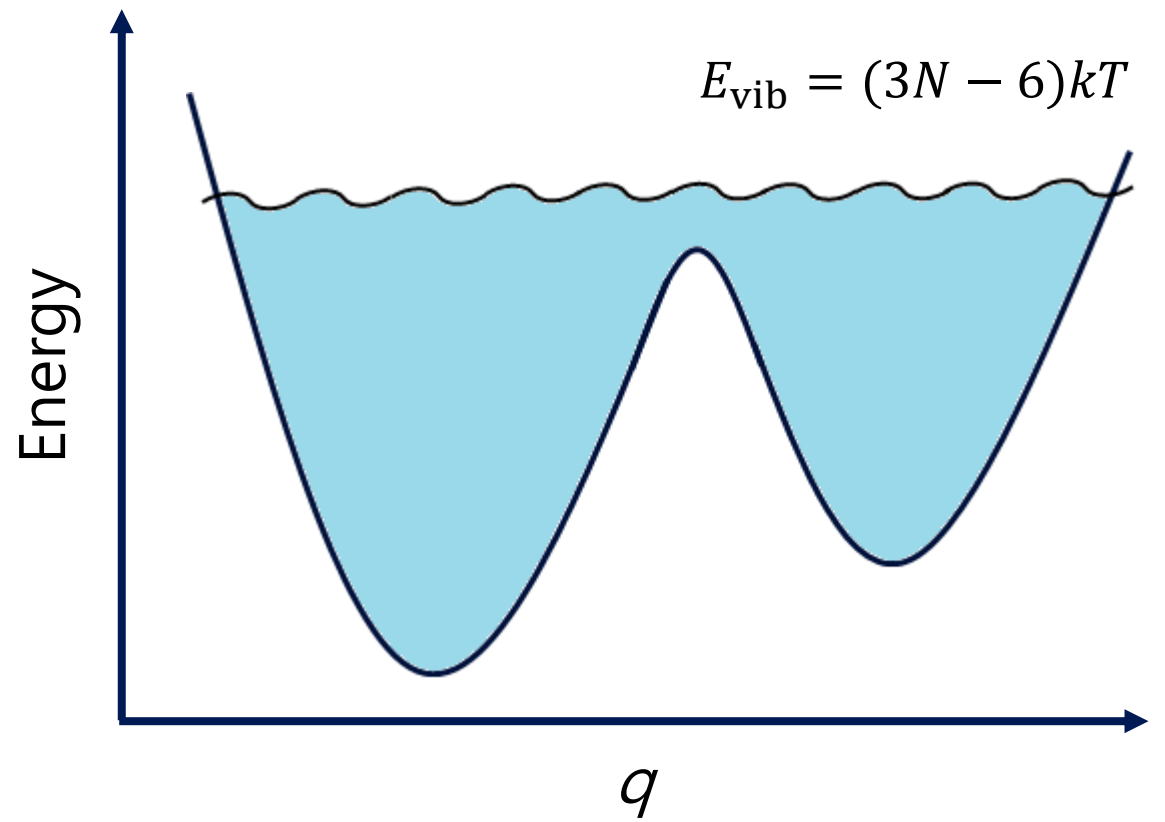
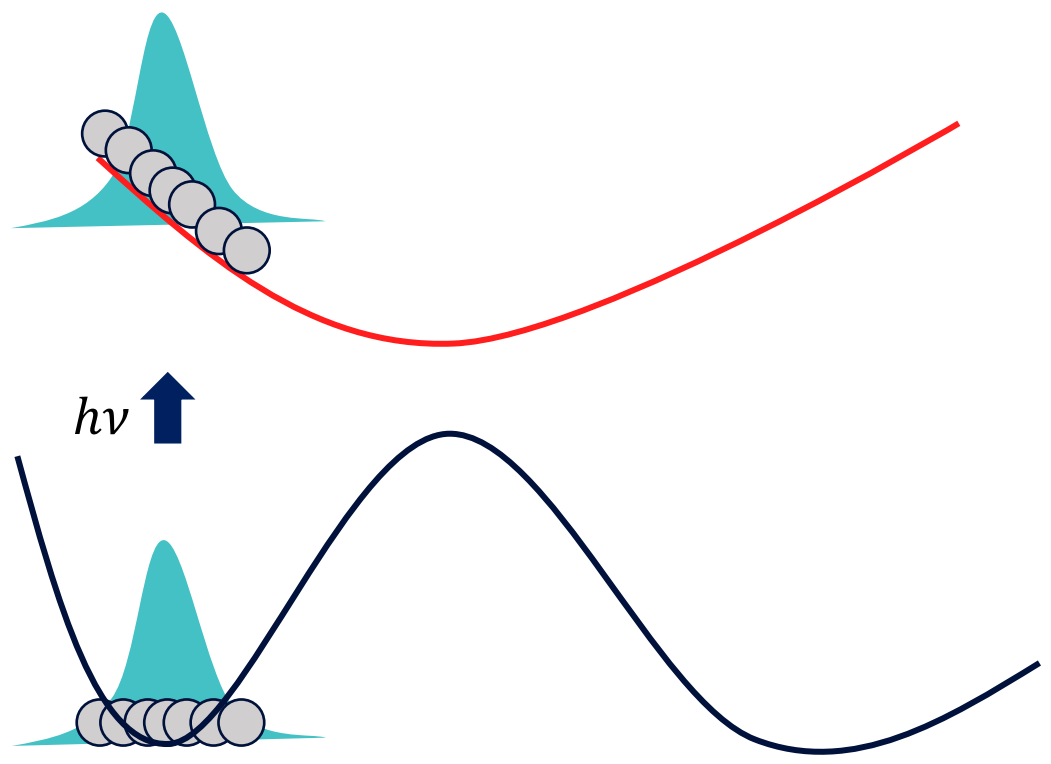
Tutorial #3: Investigating PES near a CI with TD-DFTB and DFTB/SSR

Tutorial #4: SH(EDC) and SHXF dynamics of CNH_4^+ with DFTB/SSR

Tutorial #5: SHXF dynamics of PSB3 with DFTB/SSR

Tutorial #1: Thermal sampling of target molecules: CNH_4^+ , PSB3

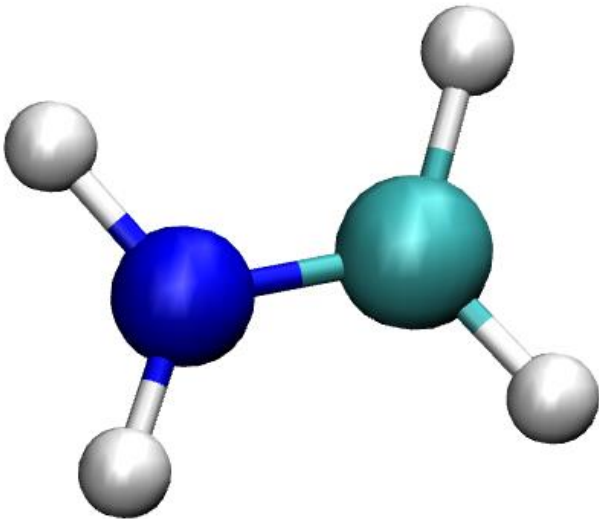
- Before running nonadiabatic dynamics, we need a proper sampling from a phase space to mimic an initial nuclear wave packet.
- Thermal sampling is to sample configurations and momenta of a system from the phase space imposing thermal equilibrium.



Tutorial #1: Thermal sampling of target molecules: CNH_4^+ , PSB3

- BOMD details for the thermal sampling

- Charge=+1; Only 1 BO state, that is, the ground state is considered.
- DFTB calculation is done to obtain E with mio-1-1 Slater-Koster (SK) parameters.
- Nosé-Hoover Chain thermostat is employed with the time coupling constant is 10.0 fs.
- BOMD dynamics of CNH_4^+ at the ground state during 25 ps; $dt=0.5$ fs.



Tutorial #1: Thermal sampling of target molecules: CNH_4^+ , PSB3

● Write a running script (run.py) for pyUNIxMD.

```
from molecule import Molecule
from thermostat import NHC
import qm, mqc
```

← Import necessary pyUNIxMD packages.

```
# Define the target system.
```

```
geom = """
6
```

```
C  -0.04474383   0.02851832   0.04519326   0.000 0.000 0.000
N  -0.04441695  -0.06449170   1.27599013   0.000 0.000 0.000
H   0.89522209  -0.03575664  -0.51205651   0.000 0.000 0.000
H   0.88084838  -0.07197904   1.86478714   0.000 0.000 0.000
H  -0.81785945   0.01033355   1.81981263   0.000 0.000 0.000
H  -1.02074371  -0.01600249  -0.55063563   0.000 0.000 0.000
"""
```

← Make a Molecule object describing your system.

```
mol = Molecule(geometry=geom, nstates=1, charge=+1.)
```

← Make a DFTB object.

```
# Set QM method.
```

```
qm = qm.dftbplus.DFTB(molecule=mol, version="20.1", install_path="/user/username/_install_dftb_20.1/", sk_path="/user/username/mio-1-1/")
```

Change the paths accordingly!

```
thermo = NHC(time_scale = 10.0, temperature=300.)
```

← Use a Nosé-Hoover Chain thermostat.

```
# Determine MD method.
```

```
md = mqc.BOMD(molecule=mol, thermostat=thermo, nsteps=50000, dt=0.5, istate=0)
```

← Make a BOMD object to run dynamics.

```
# Execute the simulation.
```

```
md.run(qm=qm)
```

← Run BOMD.

- nsteps = the # of MD steps
- dt = nuclear time step
- istate = the target adiabatic state (0: ground)

Tutorial #1: Thermal sampling of target molecules: CNH_4^+ , PSB3

- Run the dynamics. The standard output gives an outline of the dynamics and information of each MD step.

```
$ python run.py >log
```

```
-----  
PyUNIxMD version 20.1
```

```
< Developers >
```

```
Seung Kyu Min, In Seong Lee, Jong-Kwon Ha, Daeho Han,  
Kicheol Kim, Tae In Kim, Sung Wook Moon
```

```
-----  
Please cite PyUNIxMD as follows:
```

```
I. S. Lee, J.-K. Ha, D. Han, T. I. Kim, S. W. Moon, & S. K. Min.  
PyUNIxMD: A Python-based excited state molecular dynamics package.  
Journal of Computational Chemistry, 42:1755-1766. 2021
```

```
PyUNIxMD begins on 2022-06-22 22:05:50
```

```
-----  
Initial Coordinate (au)
```

```
-----  
      X           Y           Z           Mass  
C      -0.08455358    0.05389181    0.08540288    21894.16674  
N      -0.08393586   -0.12187164    2.41127171    25532.72508  
H       1.69172445   -0.06757025   -0.96764649    1837.36222  
H       1.66456208   -0.13602066    3.52393672    1837.36222  
H      -1.54553026    0.01952758    3.43894722    1837.36222  
H      -1.92892592   -0.03024032   -1.04055046    1837.36222
```

```
•  
•  
•
```

```
-----  
Dynamics Information
```

```
QM Program      =      dftbplus  
QM Method       =      DFTB
```

```
MQC Method      =      BOMD  
Time Interval (fs) =      0.500000  
Initial State (0:GS) =      0  
Nuclear Step    =      50000
```

```
Output Frequency =      1  
Verbosity Level  =      0
```

```
-----  
Thermostat Information
```

```
Thermostat      =      Nose-Hoover chain  
Target Temperature (K) =      300.000  
Time Scale (fs) =      10.000  
Chain Length     =      3  
Order            =      3  
Integrator Steps =      1
```

```
-----  
Start Dynamics
```

```
-----  
#INFO  STEP  State  Kinetic(H)  Potential(H)  Total(H)  Temperature(K)  
INFO   0     0     0.00000000  -5.26476310  -5.26476310  0.000000  
INFO   1     0     0.00136572  -5.26617565  -5.26480993  71.876421  
INFO   2     0     0.00477962  -5.26969234  -5.26491272  251.547257  
INFO   3     0     0.00864259  -5.27364892  -5.26500633  454.851833  
INFO   4     0     0.01137980  -5.27643681  -5.26505701  598.908506  
INFO   5     0     0.01213073  -5.27720031  -5.26506958  638.429526
```

Continued

Tutorial #1: Thermal sampling of target molecules: CNH_4^+ , PSB3

- In the md/ directory, there are a movie file (MOVIE.xyz) and energy profile (MDENERGY) of the dynamics.

MDENERGY

#	Step	Kinetic(H)	Potential(H)	Total(H)	E(0)(H)
	0	0.00000000	-5.26476310	-5.26476310	-5.26476310
	1	0.00136572	-5.26617565	-5.26480993	-5.26617565
	2	0.00477962	-5.26969234	-5.26491272	-5.26969234
	3	0.00864259	-5.27364892	-5.26500633	-5.27364892
	4	0.01137980	-5.27643681	-5.26505701	-5.27643681
	5	0.01213073	-5.27720031	-5.26506958	-5.27720031
	6	0.01090941	-5.27596079	-5.26505138	-5.27596079
	7	0.00842216	-5.27343345	-5.26501129	-5.27343345
	8	0.00576658	-5.27072789	-5.26496131	-5.27072789
	9	0.00404730	-5.26897788	-5.26493058	-5.26897788
	10	0.00395230	-5.26889641	-5.26494411	-5.26889641
	11	0.00547636	-5.27048207	-5.26500570	-5.27048207
	12	0.00795132	-5.27305155	-5.26510023	-5.27305155
	13	0.01034960	-5.27555888	-5.26520929	-5.27555888
	14	0.01167268	-5.27699140	-5.26531872	-5.27699140
	15	0.01127518	-5.27667887	-5.26540369	-5.27667887
	16	0.00912401	-5.27455954	-5.26543553	-5.27455954
	17	0.00595008	-5.27134913	-5.26539905	-5.27134913
	18	0.00310808	-5.26843828	-5.26533020	-5.26843828
	19	0.00197998	-5.26728305	-5.26530307	-5.26728305
	20	0.00314779	-5.26851440	-5.26536661	-5.26851440
	21	0.00599259	-5.27149558	-5.26550299	-5.27149558
	22	0.00909822	-5.27475747	-5.26565925	-5.27475747
	23	0.01109985	-5.27689395	-5.26579410	-5.27689395
	24	0.01132049	-5.27721435	-5.26589386	-5.27721435
	25	0.00991230	-5.27586492	-5.26595262	-5.27586492
	26	0.00763470	-5.27361391	-5.26597922	-5.27361391
	27	0.00547580	-5.27146920	-5.26599340	-5.27146920
	28	0.00423587	-5.27026157	-5.26602571	-5.27026157

⋮

MOVIE.xyz

6	Step:	0	Position(A)	Velocity(au)
C	-0.04474383	0.02851832	0.04519326	0.00000000
N	-0.04441695	-0.06449170	1.27599013	0.00000000
H	0.89522209	-0.03575664	-0.51205651	0.00000000
H	0.88084838	-0.07197904	1.86478714	0.00000000
H	-0.81785945	0.01033355	1.81981263	0.00000000
H	-1.02074371	-0.01600249	-0.55063563	0.00000000
6	Step:	1	Position(A)	Velocity(au)
C	-0.04485029	0.02845637	0.04476312	-0.00001910
N	-0.04393518	-0.06448518	1.27620180	0.00008504
H	0.89556274	-0.03555676	-0.51231794	0.00006118
H	0.87875747	-0.07206956	1.86326786	-0.00037395
H	-0.82264140	0.01045389	1.82287369	-0.00084000
H	-1.01963776	-0.01558464	-0.54973191	0.00019865
6	Step:	2	Position(A)	Velocity(au)
C	-0.04516162	0.02827061	0.04348857	-0.00003674
N	-0.04255682	-0.06446022	1.27685559	0.00015861
H	0.89656032	-0.03495741	-0.51308885	0.00011800
H	0.87266846	-0.07234145	1.85883038	-0.00071431
H	-0.83623379	0.01074659	1.83152739	-0.00155296
H	-1.01639830	-0.01433809	-0.54707488	0.00038296
6	Step:	3	Position(A)	Velocity(au)
C	-0.04565373	0.02796121	0.04141735	-0.00005142
N	-0.04046629	-0.06440236	1.27799527	0.00021167
H	0.89814358	-0.03395980	-0.51432972	0.00016637
H	0.86313459	-0.07279595	1.85183844	-0.00098664

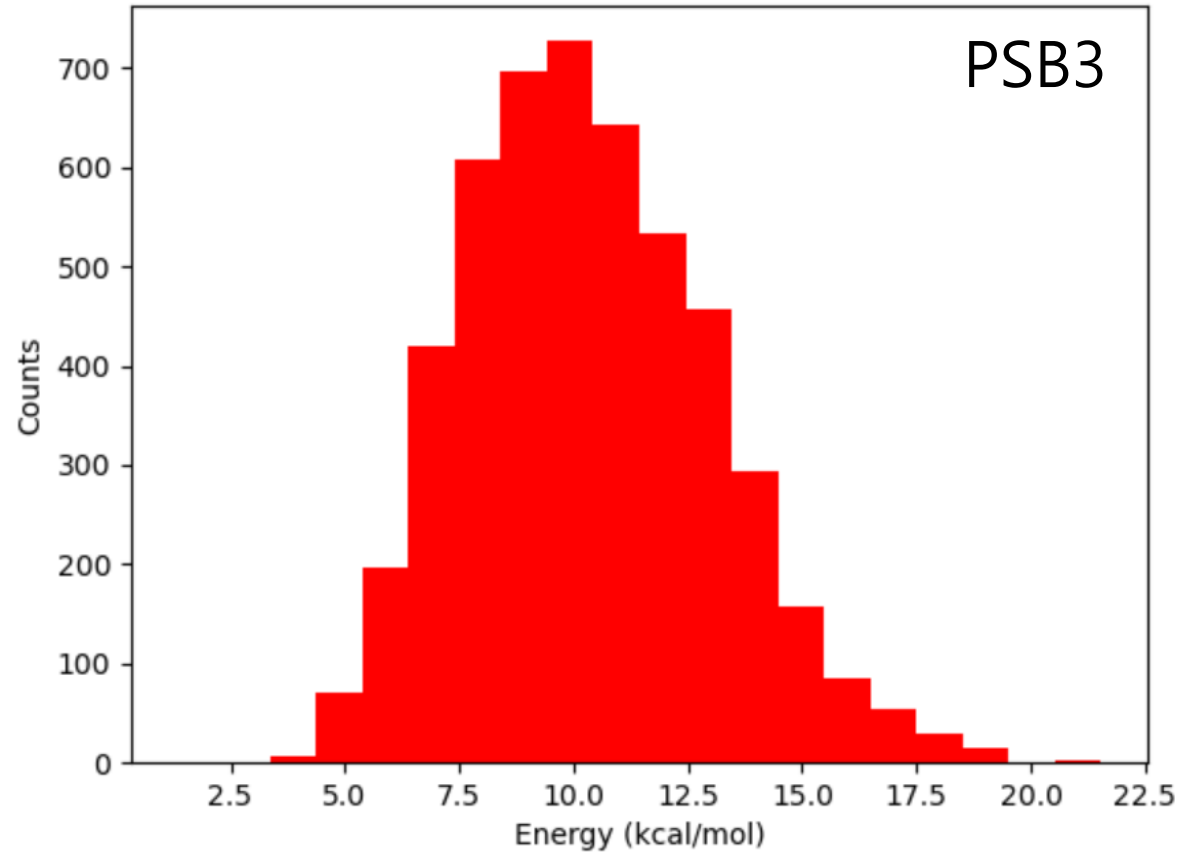
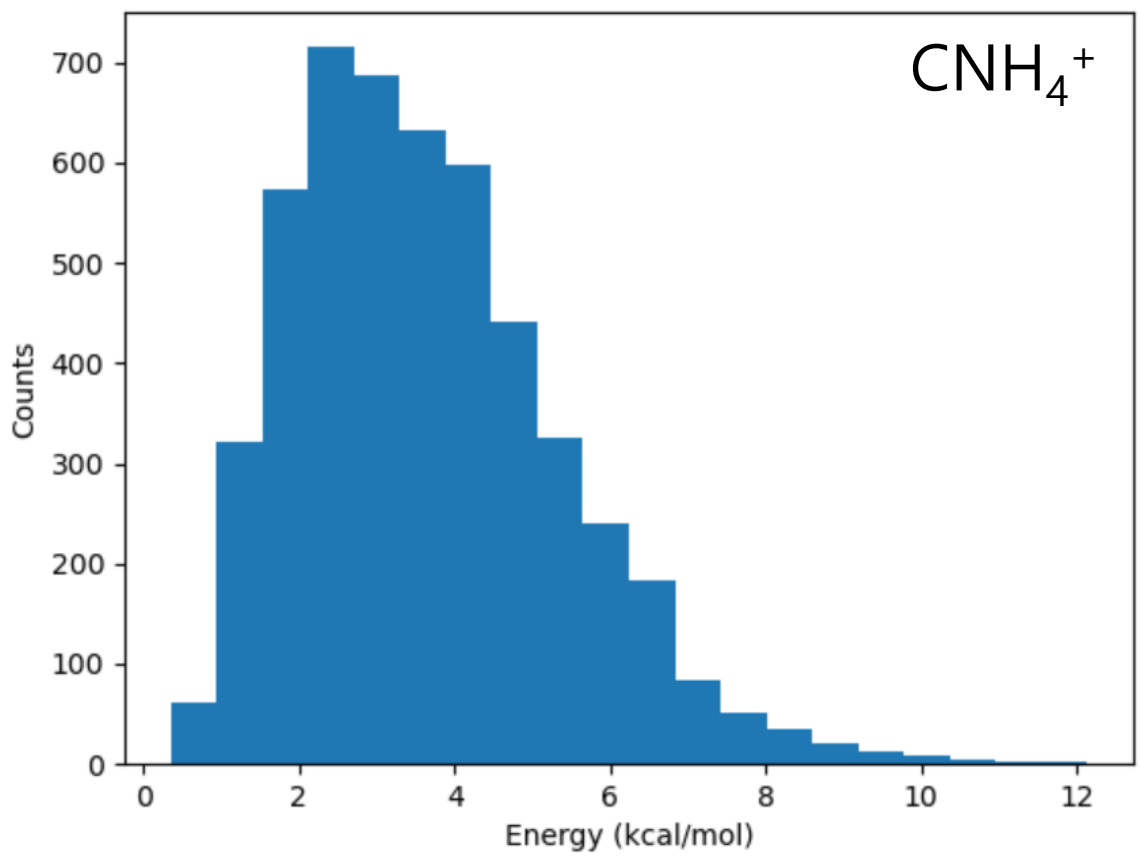
⋮

Tutorial #1: Thermal sampling of target molecules: CNH_4^+ , PSB3

- Employing the configurations and momenta from the MD trajectories, we can investigate the distribution of kinetic energies and instantaneous temperatures.

$$\sum_v \frac{1}{2} M_v \mathbf{V}_v^2(t) = \frac{3N - 6}{2} kT(t)$$

From the last 5,000 MD frames,



● SH(EDC) dynamics details

- Upto 3 electronic states (S_0, S_1, S_2) are considered; $S_2 \rightarrow S_1$ dynamics of CNH₄⁺ is interested.
- TDDFTB calculation is done to obtain E, \mathbf{F} , NACME with mio-1-1 Slater-Koster (SK) parameters.
- Nosé-Hoover Chain thermostat is employed to proceed the dynamics w/o dissociation of CNH₄⁺.
- SH(EDC) dynamics of CNH₄⁺ is run during 30 fs; dt=0.25 fs.
- Energy-based decoherence correction (EDC) can be used if you want.

$$\dot{C}_n = -\frac{i}{\hbar} E_n C_n - \sum_k \tau_{nk} C_k + \xi_n \quad \xi_{n \neq a}^{\text{SHEDC}} = -\frac{|E_n - E_a|}{\hbar} \left(1 + \frac{\alpha}{T}\right)^{-1} C_{n \neq a}$$

J. Chem. Phys. **2010**, 133, 134111.

Tutorial #2: SH(EDC) and SHXF dynamics of CNH4+ with TD-DFTB

● Prepare running scripts (run.py).

```
from molecule import Molecule
from thermostat import NHC
import qm, mqc
```

```
# Define the target system.
with open("geom.xyz", "r") as fp:
    geom = fp.read()
mol = Molecule(geometry=geom, nstates=3, charge=+1)
```

← Upto 3 electronic states is considered. ⇒ nstates=3

```
# Set QM method.
qm = qm.dftbplus.DFTB(molecule=mol, version="20.1", install_path="/user/username/_install_dftb_20.1/", sk_path="/user/username/mio-1-1/")
```

```
thermo = NHC(time_scale = 3.0, temperature=300.)
```

Change the paths accordingly!

```
# Determine MD method.
md = mqc.SH(molecule=mol, thermostat=thermo, nsteps=120, nesteps=1000, dt=0.25, istate=2, elec_object="density", hop_rescale="energy", hop_reject="keep")
```

```
# Execute the simulation.
```

```
md.run(qm=qm)
```

- The initial state is the 2nd excited state (S_2). ⇒ istates=2
- Electronic propagation is done w/ interpolation. ⇒ nestep=1000
- The density matrix is propagated by electronic propagation routine. ⇒ elec_object="density"
- After a hop, nuclear velocities are rescaled according to the transition energy. ⇒ hop_rescale="energy"
- When a hop is classically frustrated, nuclear velocities are not changed. ⇒ hop_reject="keep"
- If you want to use the energy-based decoherence, add the following argument. ⇒ dec_correction="edc"

Tutorial #2: SH(EDC) and SHXF dynamics of CNH4+ with TD-DFTB

- Make MD inputs with `input_gen.py`.

```
$ python input_gen.py -d /your-tut-dir/TUT1/01-dftb-BOMD-cnh4/sampling -f run.py -n 200
```

Usage:

`input_gen.py` -d the path of sampled xyz files -f the path of a running script (run.py) -n the number of trajectories

This script provides with MD trajectory directories (TRAJ_001, TRAJ_002, ...) where there are the running script (run.py) and an xyz file for the initial geometry/velocity (geom.xyz) copied from sampled xyz files (sample_001.xyz, sample_002.xyz ...).

- Run the dynamics.

```
$ python submit_all.py
```

- You can obtain average population, coherence, etc. with `statistical_analysis.py`.

Tutorial #2: SH(EDC) and SHXF dynamics of CNH4+ with TD-DFTB

● BOPOP, SHSTATE:

BOPOP

ρ_{00}

ρ_{11}

ρ_{22}

#	Density Matrix: population Re; see the manual for detail orders		
0	0.00000000	0.00000000	1.00000000
1	0.00000001	0.00000041	0.99999958
2	0.00000005	0.00000405	0.99999589
3	0.00000014	0.00001526	0.99998460
4	0.00000021	0.00004100	0.99995879
5	0.00000033	0.00008206	0.99991761
6	0.00000041	0.00014425	0.99985534
7	0.00000053	0.00025982	0.99973966
8	0.00000058	0.00051854	0.99948088
9	0.00000065	0.00112715	0.99887220
10	0.00000065	0.00258845	0.99741090
11	0.00000069	0.00636965	0.99362966
12	0.00000065	0.01778148	0.98221786
13	0.00000066	0.06065432	0.93934502
14	0.00000051	0.23668625	0.76331324
15	0.00000021	0.54971721	0.45028257
16	0.00000019	0.52392547	0.47607434
17	0.00000057	0.33771029	0.66228914
18	0.00000046	0.21805087	0.78194867
19	0.00000067	0.15419903	0.84580030
20	0.00000057	0.12318203	0.87681740
21	0.00000047	0.11085732	0.88914221
22	0.00000063	0.10874634	0.89125303
23	0.00000039	0.11175457	0.88824504
24	0.00000042	0.11687767	0.88312191
25	0.00000047	0.12244470	0.87755483
26	0.00000024	0.12760522	0.87239454
27	0.00000035	0.13197230	0.86802735
28	0.00000033	0.13544401	0.86455566
29	0.00000017	0.13806261	0.86193721
30	0.00000030	0.13993400	0.86006570

⋮

SHSTATE

#	Step	Running State
	0	2
	1	2
	2	2
	3	2
	4	2
	5	2
	6	2
	7	2
	8	2
	9	2
	10	2
	11	2
	12	2
	13	2
	14	2
	15	2
	16	2
	17	2
	18	2
	19	2
	20	2
	21	2
	22	2
	23	2
	24	2
	25	2
	26	2
	27	2
	28	2
	29	2
	30	2

31	2
32	2
33	2
34	2
35	2
36	2
37	2
38	2
39	2
40	2
41	2
42	2
43	2
44	2
45	2
46	2
47	2
48	2
49	1
50	1
51	1
52	1
53	1
54	1
55	1
56	1
57	1
58	1
59	1
60	1
61	1

⋮

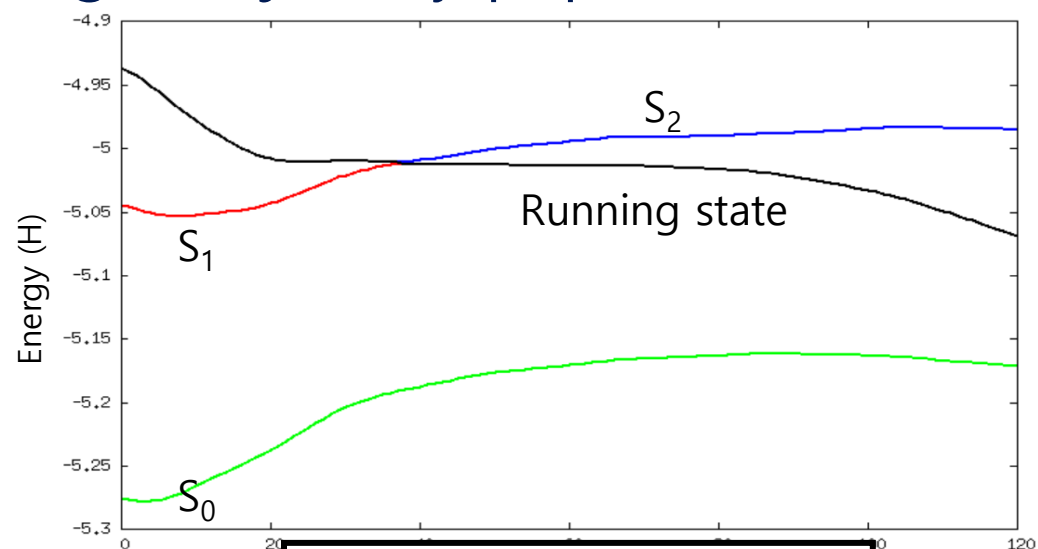
Tutorial #2: SH(EDC) and SHXF dynamics of CNH4+ with TD-DFTB

BOCOH, NACMD:

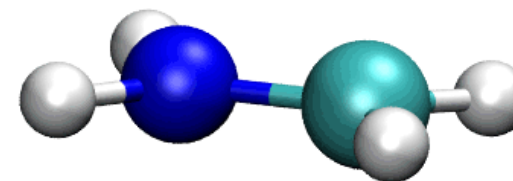
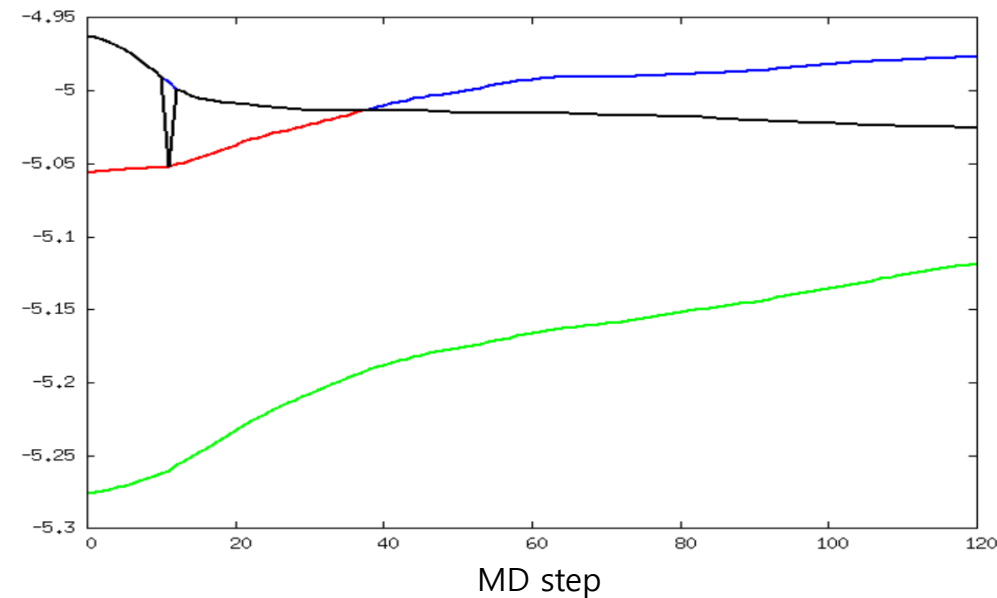
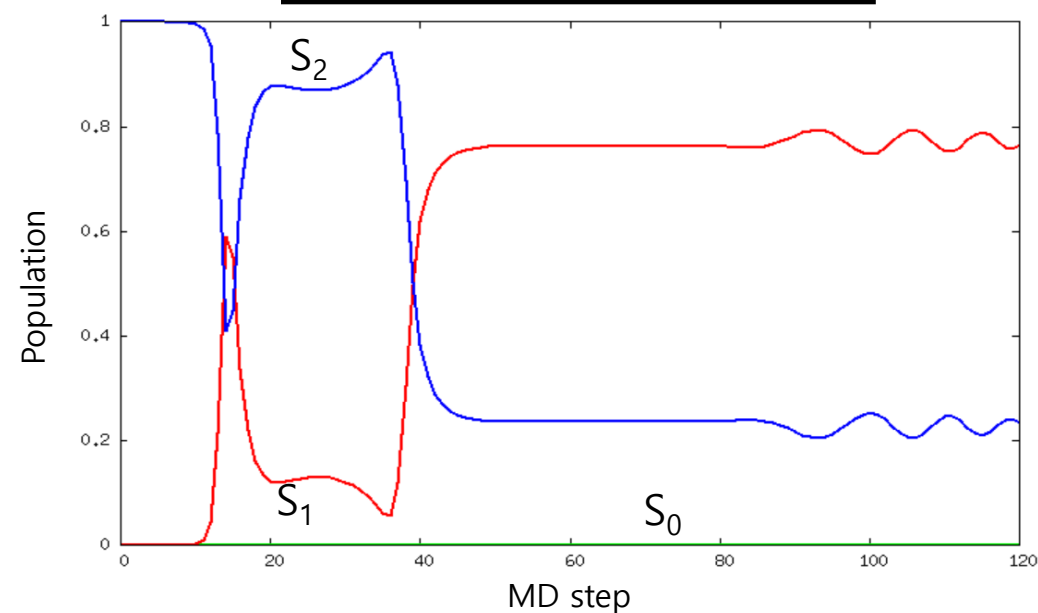
BOCOH								NACME			
#	Density	Re ρ_{01}	Im ρ_{01}	Re ρ_{02}	Im ρ_{02}	Re ρ_{12}	Im ρ_{12}	#	τ_{01}	τ_{02}	τ_{12}
0	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0	0.00000000	0.00000000	0.00000000
1	0.00000005	-0.00000005	-0.00005135	0.00010125	-0.00059743	0.00022971	0.000140499	1	0.00000491	0.00003119	0.00012830
2	0.00000037	-0.00000028	-0.00003695	0.00022761	-0.00144225	0.00328735	0.00583058	2	0.00000620	0.00008031	0.00019202
3	0.00000129	-0.00000068	-0.00003072	0.00037200	-0.00211101	0.01154360	0.02116625	3	0.00000866	0.00011995	0.00041379
4	0.00000274	-0.00000101	-0.00003363	0.00045535	-0.00264597	0.03050034	0.04519629	4	0.00001158	0.00015243	0.00060618
5	0.00000498	-0.00000141	-0.00002256	0.00057124	-0.00280635	0.06871634	0.10878191	5	0.00001449	0.00018104	0.00084356
6	0.00000749	-0.00000175	-0.00003187	0.00063958	-0.00331243	0.18331082	0.32199262	6	0.00001710	0.00020483	0.00115732
7	0.00001117	-0.00000344	-0.00001188	0.00072485	-0.00499541	0.47454098	0.7454098	7	0.00001913	0.00022230	0.00158551
8	0.00001629	-0.00000590	-0.00002242	0.00076029	-0.00838246	1.0985457	1.10985457	8	0.00002040	0.00023303	0.00219190
9	0.00002467	-0.00001124	-0.00000182	0.00080698	-0.01398607	2.24378673	2.24378673	9	0.00002081	0.00023771	0.00309556
10	0.00003687	-0.00001817	-0.00001345	0.00080679	-0.02321727	3.7316117	3.7316117	10	0.00002030	0.00023790	0.00454298
11	0.00005733	-0.00003349	0.00000047	0.00082919	-0.04008911	5.45394042	5.45394042	11	0.00001873	0.00023543	0.00710194
12	0.00008931	-0.00006057	-0.00000637	0.00080197	-0.07504523	7.29053846	7.29053846	12	0.00001540	0.00023199	0.01224448
13	0.00015240	-0.00012959	0.00000751	0.00078721	-0.15288059	10.47454098	10.47454098	13	0.00000796	0.00022821	0.02422582
14	0.00023901	-0.00024985	0.00005972	0.00061805	-0.27746442	14.45394042	14.45394042	14	-0.00000863	0.00022266	0.05186830
15	0.00028233	-0.00019544	0.00009195	0.00029686	-0.14946216	19.29053846	19.29053846	15	-0.00001965	0.00021578	0.07037152
16	0.00023652	0.00020718	-0.00008549	0.00028727	0.20824401	25.29053846	25.29053846	16	-0.00000345	0.00021397	0.04150730
17	0.00025537	0.00035758	-0.00002546	0.00061482	0.37316117	33.29053846	33.29053846	17	0.00000800	0.00020853	0.01923728
18	0.00007005	0.00030731	-0.00000018	0.00059687	0.40256741	43.09189163	43.09189163	18	0.00001165	0.00020137	0.01008334
19	-0.00009817	0.00030609	-0.00004859	0.00075127	0.35028717	55.08786628	55.08786628	19	0.00001230	0.00019369	0.00598453
20	-0.00015077	0.00021790	0.00009149	0.00070101	0.24378673	70.22040004	70.22040004	20	0.00001176	0.00018620	0.00387578
21	-0.00021781	0.00006756	-0.00003661	0.00064479	0.10985457	89.29410864	89.29410864	21	0.00001069	0.00017941	0.00266898
22	-0.00026075	-0.00001203	0.00003067	0.00074663	-0.02710598	112.31013831	112.31013831	22	0.00000956	0.00017405	0.00192276
23	-0.00019505	-0.00007364	0.00007577	0.00058288	-0.14835102	141.27795218	141.27795218	23	0.00000808	0.00016362	0.00143436
24	-0.00012817	-0.00018060	-0.00006069	0.00060571	-0.24215524	177.21113520	177.21113520	24	0.00000633	0.00015660	0.00109746
25	-0.00011376	-0.00021125	0.00006820	0.00063870	-0.30348499	222.12389025	222.12389025	25	0.00000470	0.00014600	0.00086545
26	-0.00002686	-0.00017435	0.00003054	0.00046023	-0.33239997	280.02885051	280.02885051	26	0.00000281	0.00013632	0.00069466
27	0.00006516	-0.00020591	-0.00006426	0.00055014	-0.33235728	355.06398594	355.06398594	27	0.00000092	0.00012781	0.00056808
28	0.00006056	-0.00020334	0.00008349	0.00052951	-0.30874587	451.14756311	451.14756311	28	-0.00000097	0.00012012	0.00047299
29	0.00009713	-0.00012081	0.00000208	0.00038732	-0.26768536	575.21759102	575.21759102	29	-0.00000287	0.00011311	0.00040066
30	0.00017052	-0.00011091	-0.00004636	0.00050216	-0.21508549	735.27219602	735.27219602	30	-0.00000480	0.00010665	0.00034512

- New files are generated as compared to BOMD relating to nonadiabatic transition:
 - BO population, diagonal elements of density matrix (BOPOP)
 - BO coherence, off-diagonal elements of density matrix (BOCOH)
 - Nonadiabatic coupling matrix elements (NACME)
 - Running states (SHSTATE)
 - Hopping probabilities (SHPROB).

● Single-trajectory population and molecular motion:



Single trajectory results



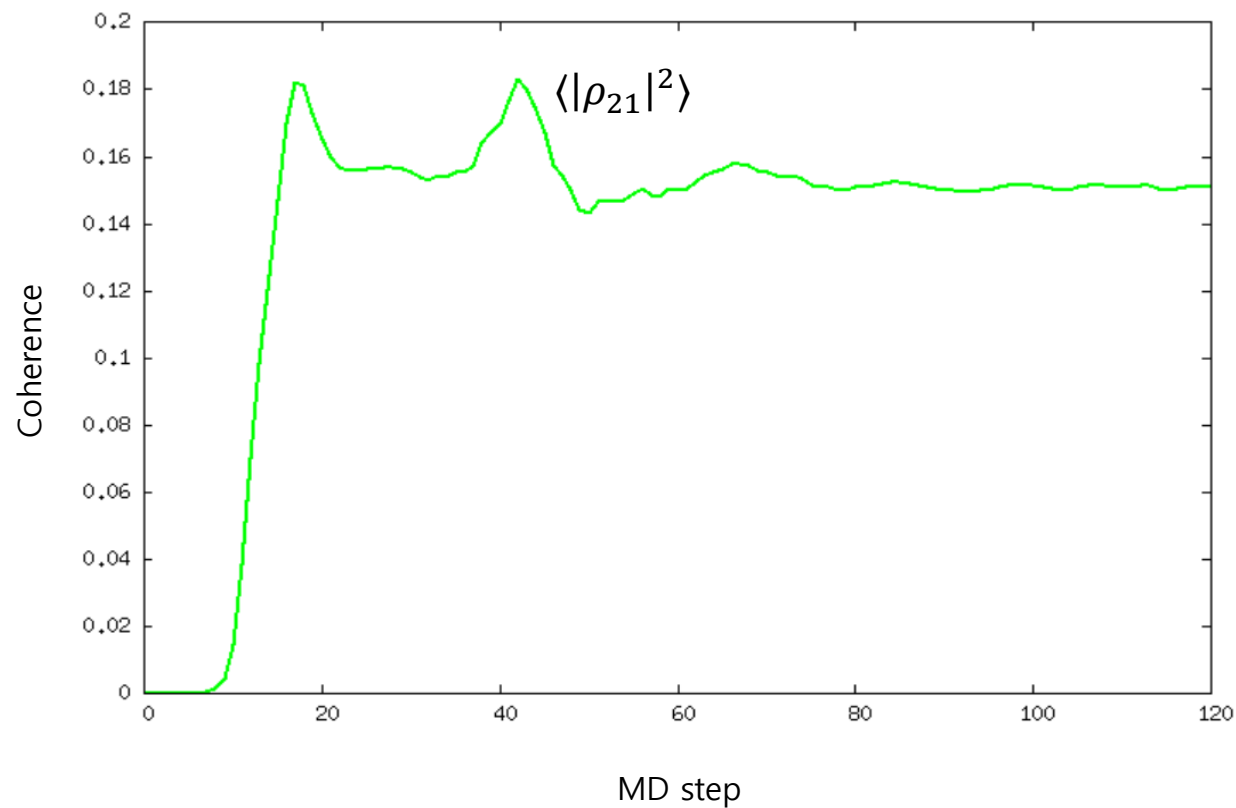
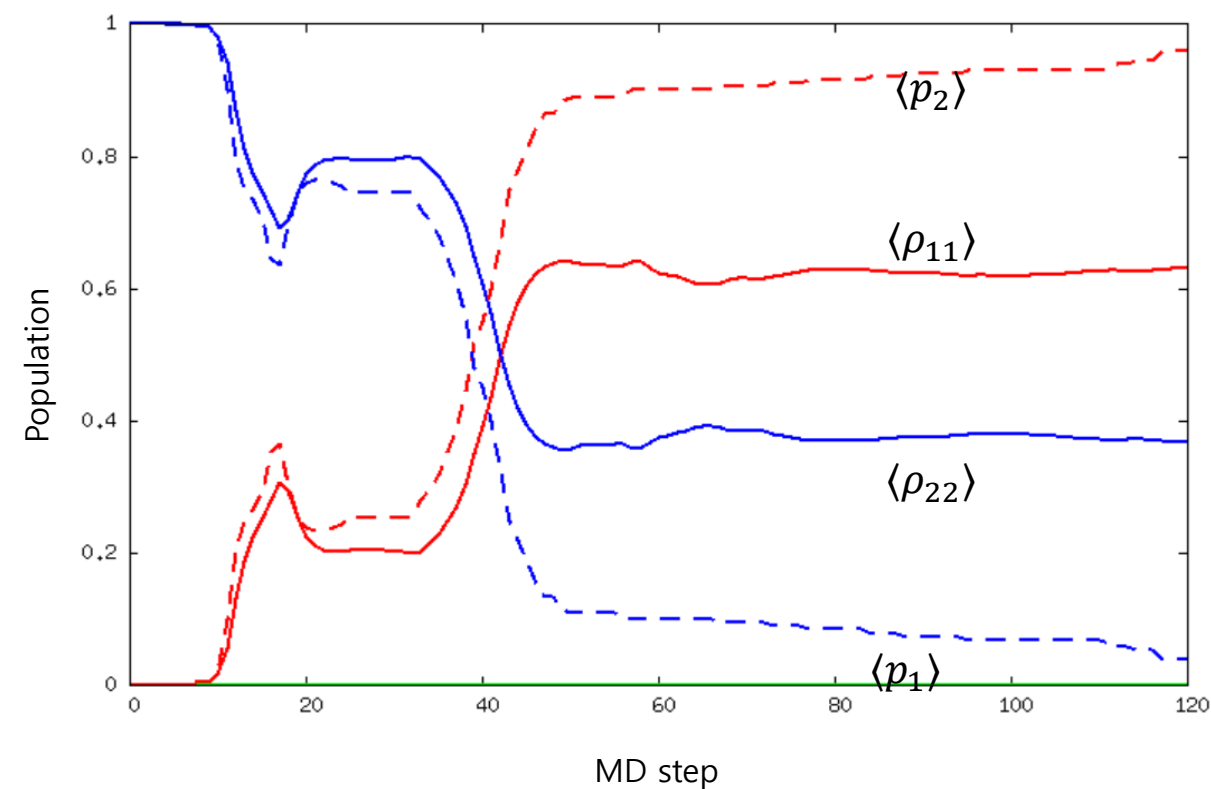
The C=N bond Elongation \rightarrow pyramidalization

Tutorial #2: SH(EDC) and SHXF dynamics of CNH4+ with TD-DFTB

- Without the decoherence, the dynamics suffers from the "overcoherence" problem.

Averaged results with 200 trajectories

$$\langle \rho_{ij} \rangle = \frac{1}{N_{\text{traj}}} \sum_I \rho_{ij}^I \quad \langle p_i \rangle = \frac{1}{N_{\text{traj}}} \sum_I N_i^I$$



● SHXF dynamics details

- SHXF dynamics of CNH₄⁺ is run during 30 fs; dt=0.25 fs.
- The width of a nuclear wave packet, $\sigma = 0.07$ a.u.

$$\dot{C}_n = -\frac{i}{\hbar} E_n C_n - \sum_k \tau_{nk} C_k + \xi_n$$

$$\xi_n^{\text{SHXF}} = \sum_k \sum_\nu \frac{1}{M_\nu} \frac{\nabla_\nu |\chi|}{|\chi|} \Big|_{\mathbf{R}(t)} \cdot (\mathbf{f}_{k\nu} - \mathbf{f}_{n\nu}) |C_k|^2 C_n$$

Tutorial #2: SH(EDC) and SHXF dynamics of CNH4+ with TD-DFTB

- Prepare running scripts (run.py).

```
from molecule import Molecule
from thermostat import NHC
import qm, mqc

# Define the target system.
with open("geom.xyz", "r") as fp:
    geom = fp.read()
mol = Molecule(geometry=geom, nstates=3, charge=+1)

# Set QM method.
qm = qm.dftbplus.DFTB(molecule=mol, version="20.1", install_path="/user/username/_install_dftb_20.1/",
sk_path="/user/username/mio-1-1/")

thermo = NHC(time_scale = 3.0, temperature=300.)

# Determine MD method.
md = mqc.SHXF(molecule=mol, thermostat=thermo, nsteps=120, nesteps=1000, dt=0.25, istate=2, elec_object="density",
hop_rescale="energy", hop_reject="keep", sigma=0.07)

# Execute the simulation.
md.run(qm=qm)
```

Change the paths accordingly!

Use an SHXF object instead of SH.

The width of a nuclear wave packet is set to be 0.07. \Rightarrow sigma=0.07

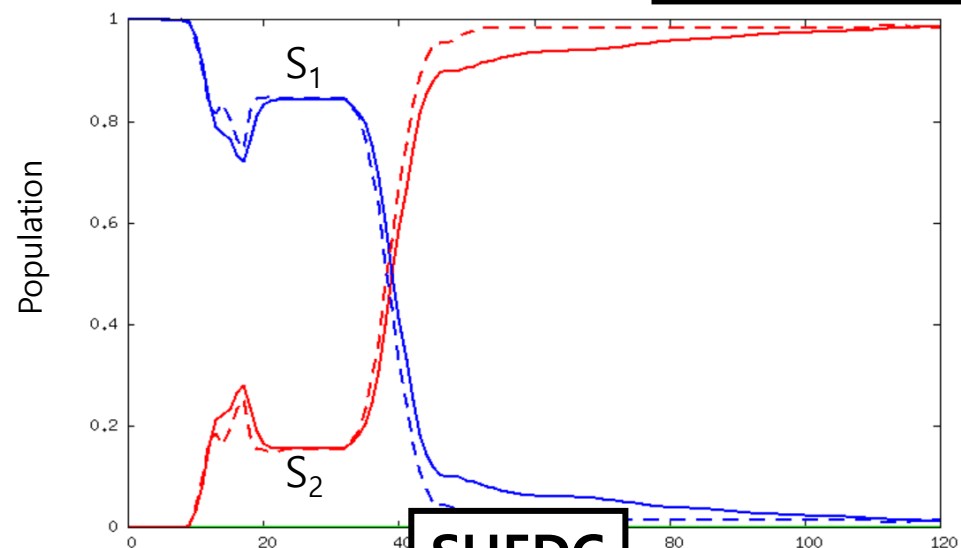
- Make MD inputs and execute them as earlier.

Tutorial #2: SH(EDC) and SHXF dynamics of CNH_4^+ with TD-DFTB

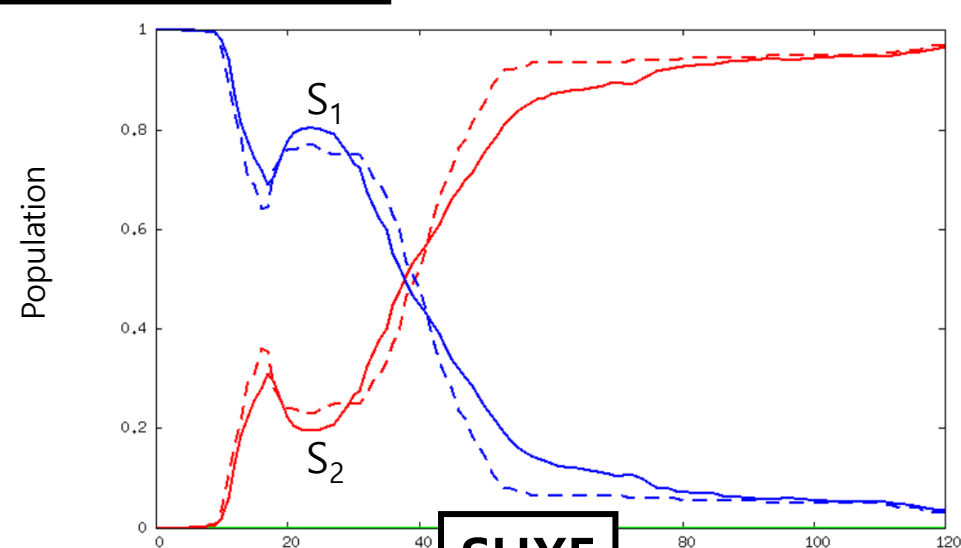
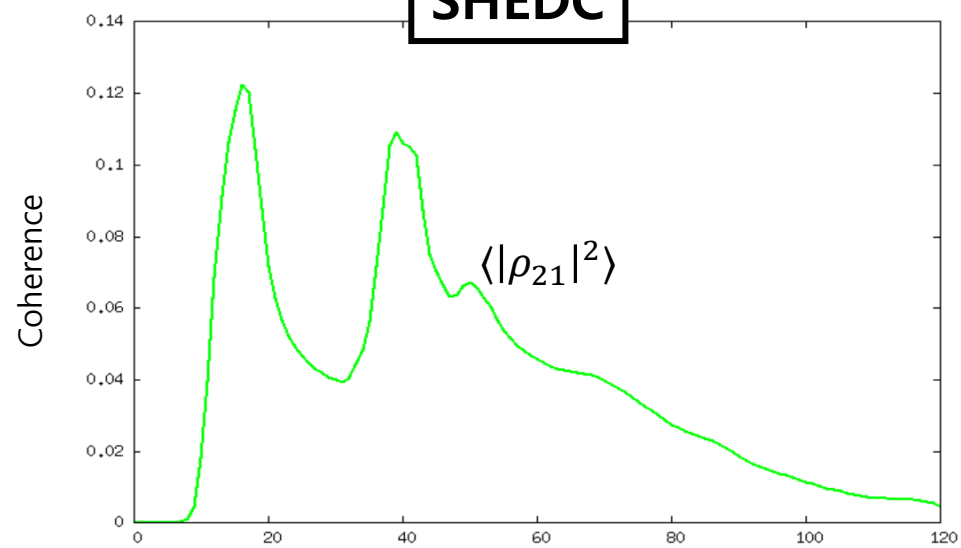
- Population/coherence dynamics in CNH_4^+ w/ decoherence

Averaged results with 200 trajectories

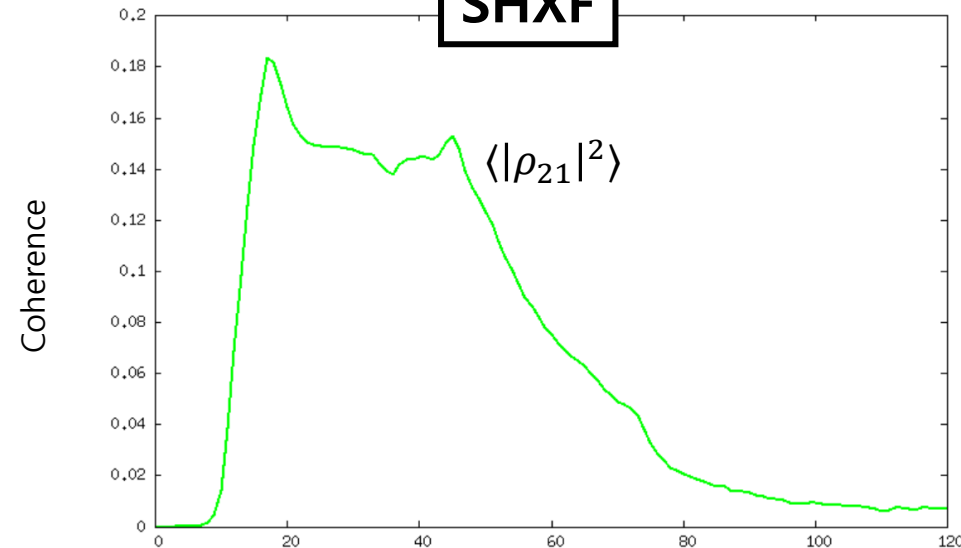
— $\langle \rho_{jj} \rangle$
- - - $\langle p_j \rangle$



SHEDC



SHXF

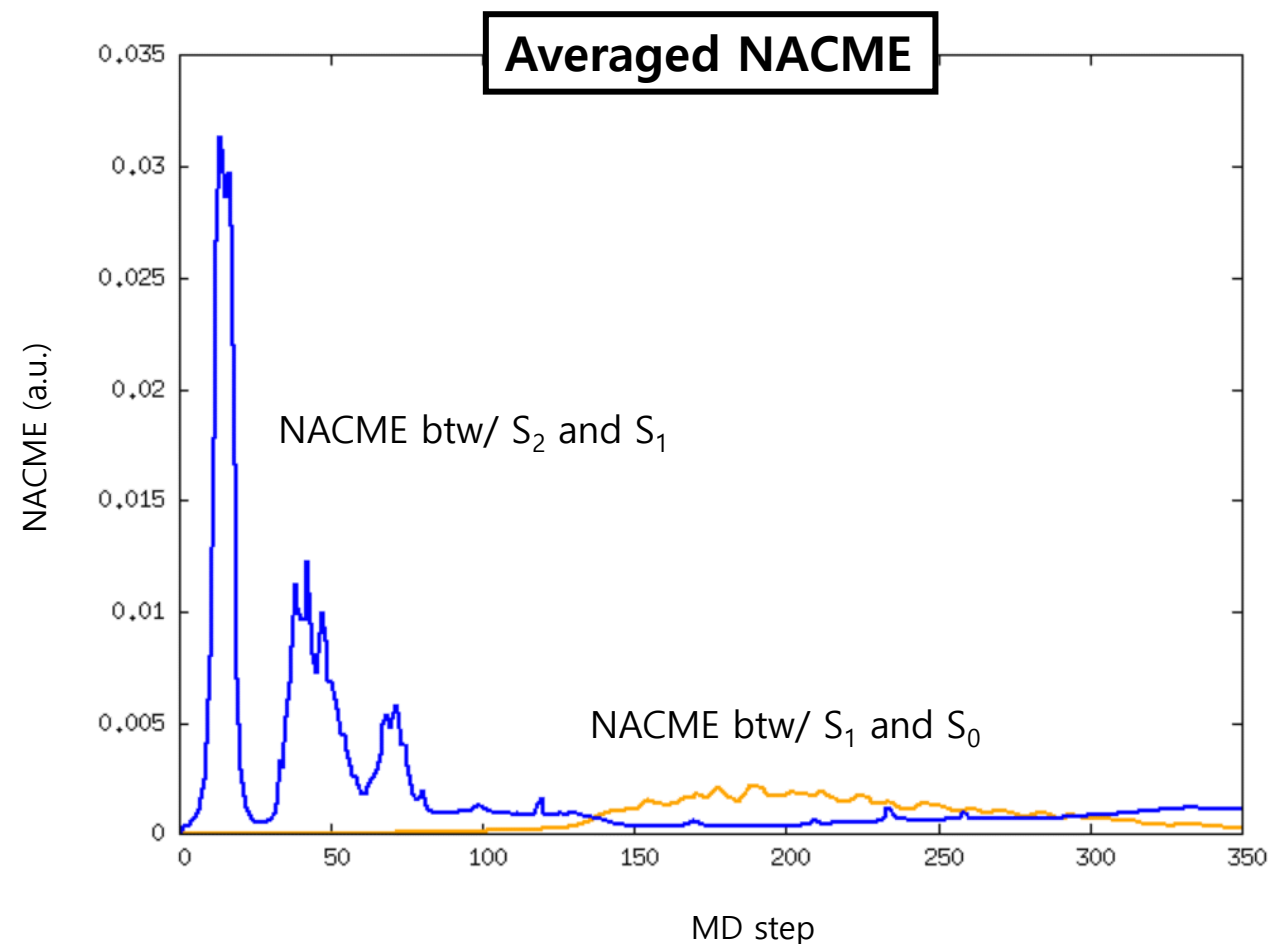
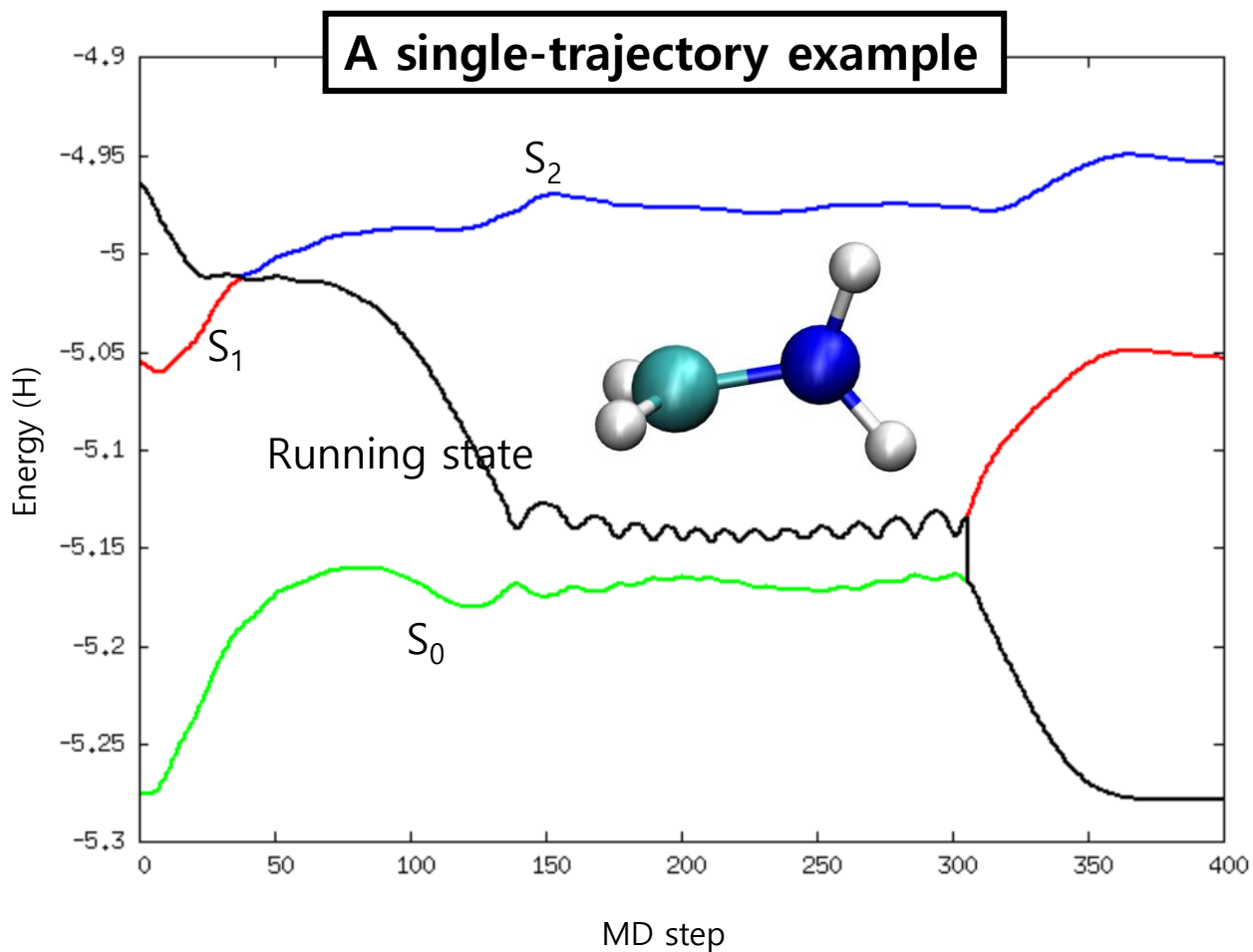


MD step

MD step

Tutorial #2: SH(EDC) and SHXF dynamics of CNH4+ with TD-DFTB

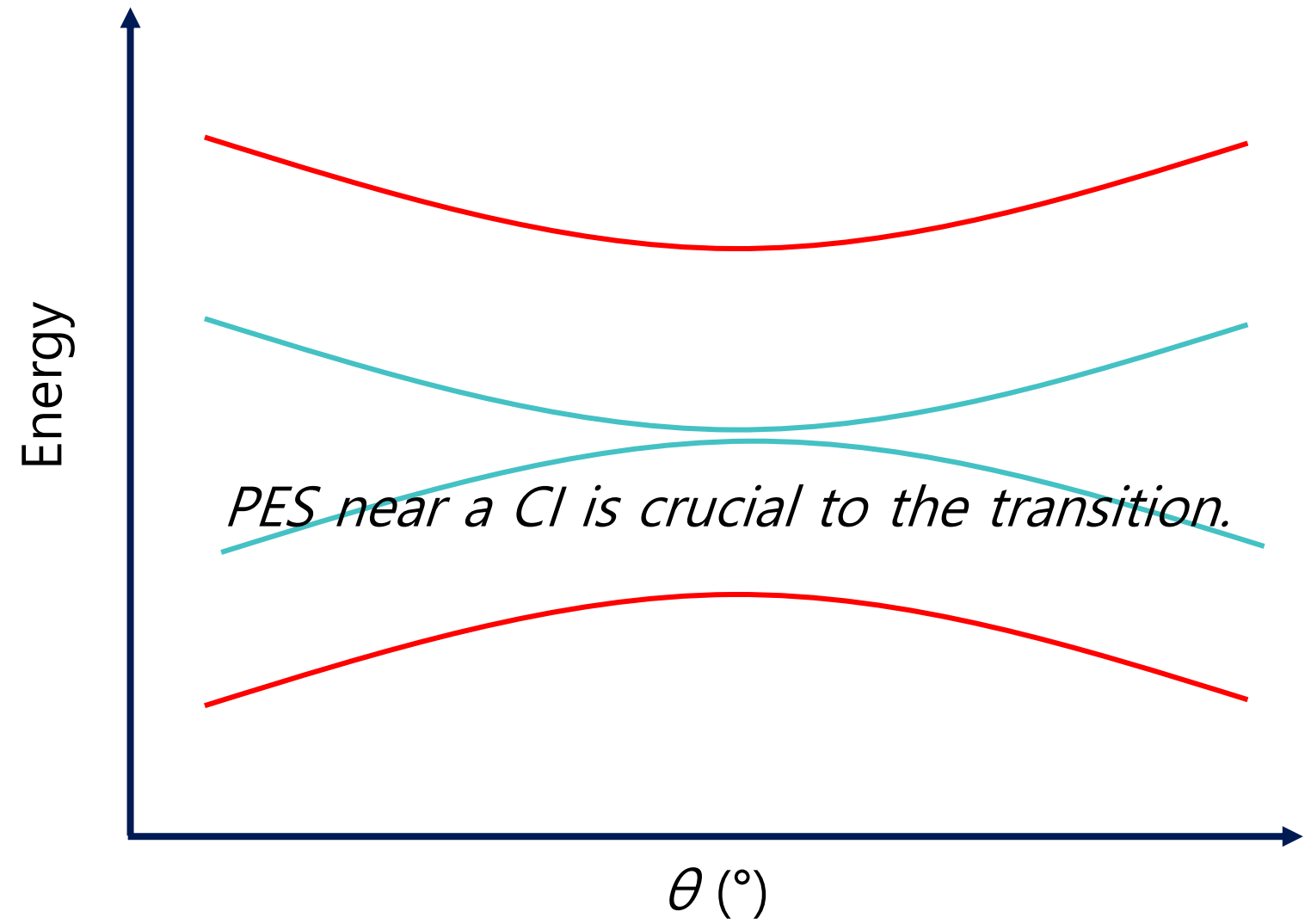
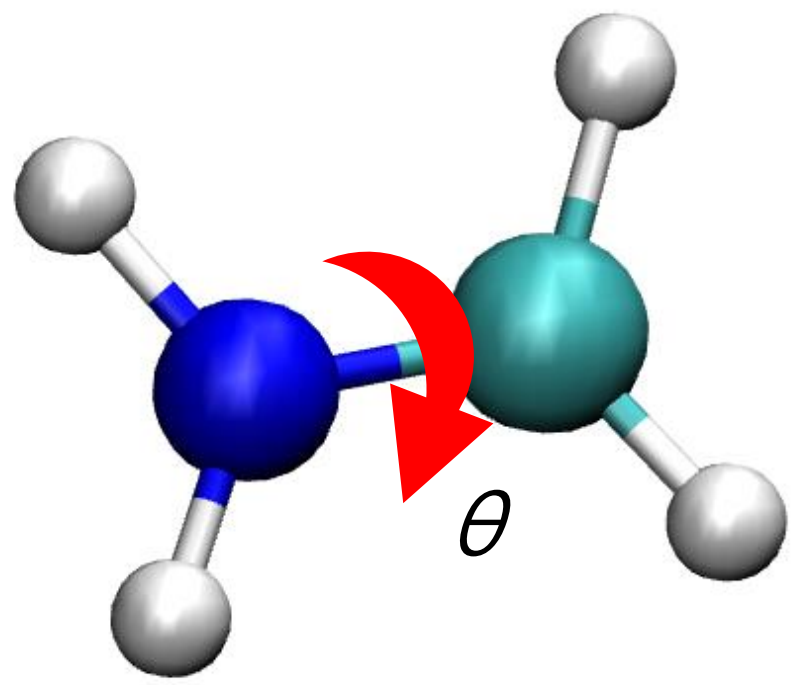
- TD-DFTB cannot describe S_1/S_0 CI well.



The dynamics is struggling with hopping from S_1 to S_0 .

Tutorial #3: Investigating PES near a CI with TD-DFTB and DFTB/SSR

- Investigate PES in terms of torsional-angle coordinates w/ TD-DFTB and DFTB/SSR.



Tutorial #3: Investigating PES near a CI with TD-DFTB and DFTB/SSR

● Prepare a running script (1/2).

```
from molecule import Molecule
import qm
```

```
install_path = "/user/username/_install_dftb_20.1/"
mio_path = "/user/username/mio-1-1/"
ob2_path = "/user/username/ob2-1-1/base/"
```

Change the paths accordingly!

```
# Define the target system.
```

```
geom = """
```

```
6
```

```
planar CNH4
```

C	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
N	0.000000	0.000000	1.335000	0.000000	0.000000	0.000000
H	0.943102	0.000000	-0.544500	0.000000	0.000000	0.000000
H	0.943102	0.000000	1.879500	0.000000	0.000000	0.000000
H	-0.943102	0.000000	1.879500	0.000000	0.000000	0.000000
H	-0.926647	0.000000	-0.535000	0.000000	0.000000	0.000000

```
"""
```

```
mol = Molecule(geometry=geom, nstates=2, charge=+1)
```

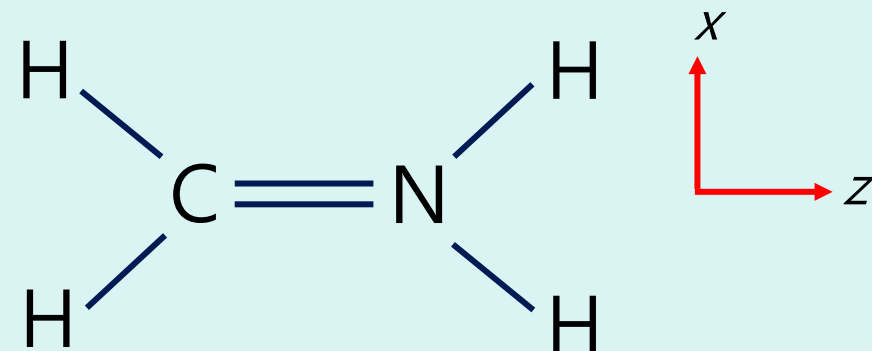
```
# Functions to perform dftb+ and print energies
```

```
def calc_dftb(qm, mol):
```

```
    qm.write_xyz(mol); qm.get_input(mol, -1, [0], False)
    qm.run_QM(mol, "./", -1, [0], False); qm.extract_QM(mol, "./", -1, [0], 1., False)
    print(f"{mol.states[0].energy:13.8f} {mol.states[1].energy:13.8f}", end="")
```

```
def calc_ssr(qm, mol):
```

```
    qm.write_xyz(mol); qm.get_input(mol, -1, [0], False)
    qm.run_QM("./", -1, [0]); qm.extract_QM(mol, [0], False)
    print(f"{mol.states[0].energy:13.8f} {mol.states[1].energy:13.8f}", end="")
```



← A planar CNH_4^+ structure, that is, the starting point of rotations S_0, S_1 are considered. \Rightarrow nstates=2

Functions having `pyUNIxMD` methods to perform DFTB+ and print energies

Continued

Tutorial #3: Investigating PES near a CI with TD-DFTB and DFTB/SSR

● Prepare a running script (2/2).

```
# Investigate S0, S1 PES by rotating the molecule
```

```
import numpy as np
def rmat_z(theta):
    Rz = np.array([np.cos(theta), -np.sin(theta), 0., np.sin(theta), np.cos(theta), 0., 0., 0., 1.])
    Rz = Rz.reshape((3,3))
    return Rz
deg2rad = np.pi/180; da = 10.; rz = rmat_z(da*deg2rad)
```

Define a rotation matrix function. We rotate the molecule by 10° repetitively.

```
for ir in range(18):
    print(f"{ir*da} ", end="")
```

Scan PES from 0° to 170°

```
# Set QM method.
```

```
qm1 = qm.dftbplus.DFTB(molecule=mol, version="20.1", install_path=install_path, sk_path=mio_path); qm1.calc_coupling=False
qm2 = qm.dftbplus.SSR(molecule=mol, version="20.1", install_path=install_path, sk_path=ob2_path, l_range_sep=True, tuning = [3.0, 3.0, 3.0]);
qm2.calc_coupling=False
calc_dftb(qm1, mol)
calc_ssr(qm2, mol)
print("")
```

Define a TDDFTB and LC-DFTB/SSR object, respectively.

l_range_sep = Whether to use range-separated DFTB

```
# Rotating the molecule
```

```
mol.pos[3] = np.dot(rz, mol.pos[3])
mol.pos[4] = np.dot(rz, mol.pos[4])
```

Update mol by multiply the rotation matrix to H's of CNH_4^+

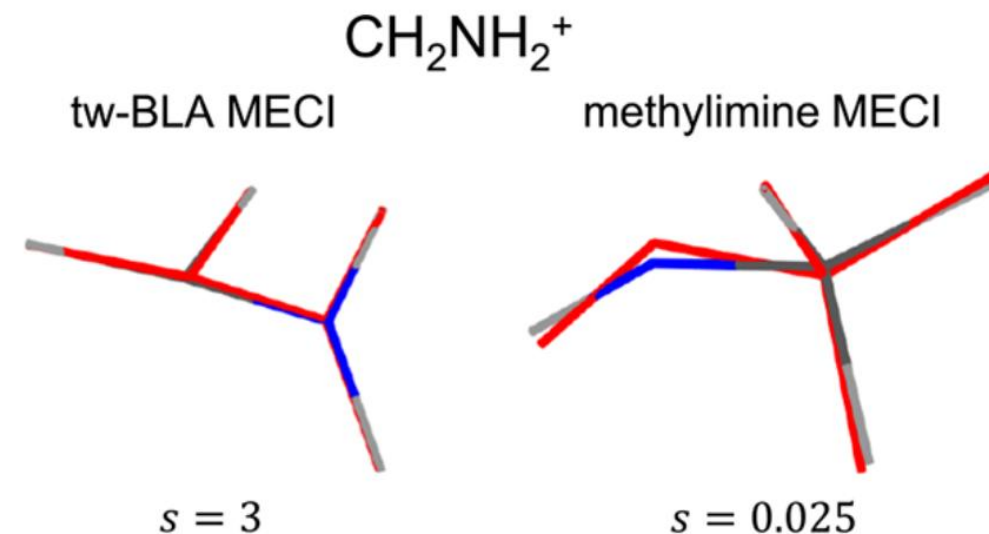
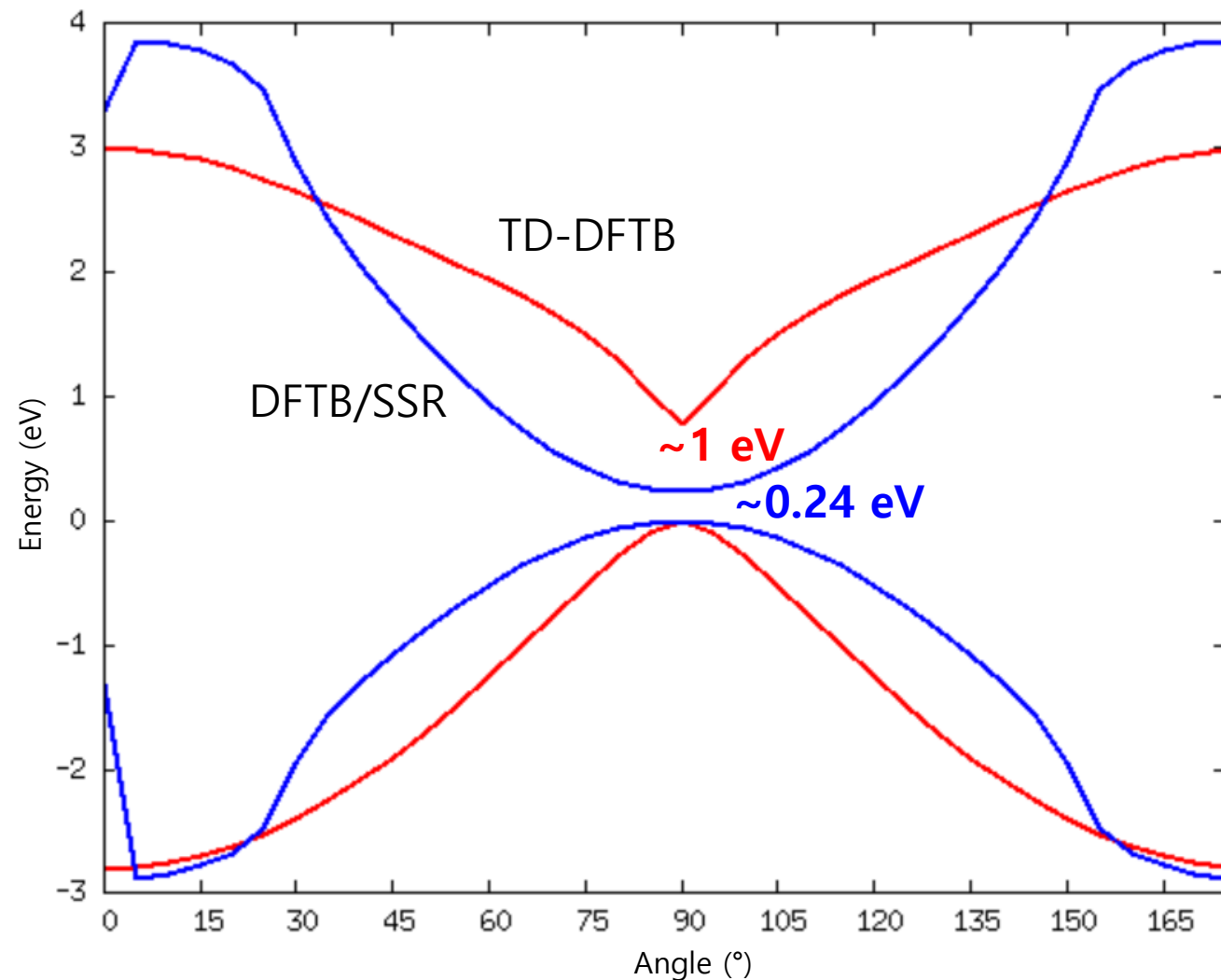
Tutorial #3: Investigating PES near a CI with TD-DFTB and DFTB/SSR

J. Chem. Theory Comput. **2019**, 15, 5, 3021–3032.

- Execute it.

```
$ sbatch submit.sh
```

Minimum energy conical intersection (MECI) optimized w/ DFTB/SSR



	SSR / ω PBEh	TD- DFTB	DFTB/ SSR	LC-DFTB/ SSR	LC- DFTB(W) /SSR
Methyliminium					
FC point	8.02	6.13	6.47	6.86	7.11(6.69)
tw-BLA MECI	4.23	n/a	n/a	5.03	3.70
met MECI	4.90	n/a	n/a	n/a	(4.72)

DFTB/SSR is able to find the S_1/S_0 MECI structure.

● SH(EDC) dynamics details

- $S_1 \rightarrow S_0$ dynamics for CNH₄⁺ are interesting. We checked that DFTB/SSR gives the S_1/S_0 CI successfully.
- DFTB/SSR calculation is done to obtain E , \mathbf{F} , NACV with ob2-1-1 Slater-Koster (SK) parameters, which is designed for range-separated DFTB calculations.
- Nosé-Hoover Chain thermostat is employed to proceed the dynamics w/o dissociation.
- SH(EDC) dynamics of CNH₄⁺ is run during 75 fs; dt=0.25 fs.
- Energy-based decoherence correction (EDC) can be used if you want.

Tutorial #4: SH(EDC) and SHXF dynamics of CNH4+ with DFTB/SSR

- Prepare a running script.

```
from molecule import Molecule
from thermostat import NHC
import qm, mqc
```

```
# Define the target system.
```

```
with open("geom.xyz", "r") as fp:
    geom = fp.read()
```

```
mol = Molecule(geometry=geom, nstates=2, charge=+1)
```

```
# Set QM method.
```

```
qm = qm.dftbplus.SSR(molecule=mol, tuning = [3.0, 3.0, 3.0], l_range_sep=True, l_state_interactions=True, version="20.1",
install_path="/user/username/_install_dftb_20.1/", sk_path="/user/username/ob2-1-1/base/")
```

Define a DFTB/SSR object.



```
thermo = NHC(time_scale = 3.0, temperature=300.)
```

Change the paths accordingly!

```
# Determine MD method.
```

```
md = mqc.SH(molecule=mol, thermostat=thermo, nsteps=300, nesteps=1000, dt=0.25, istate=1, elec_object="density")
```

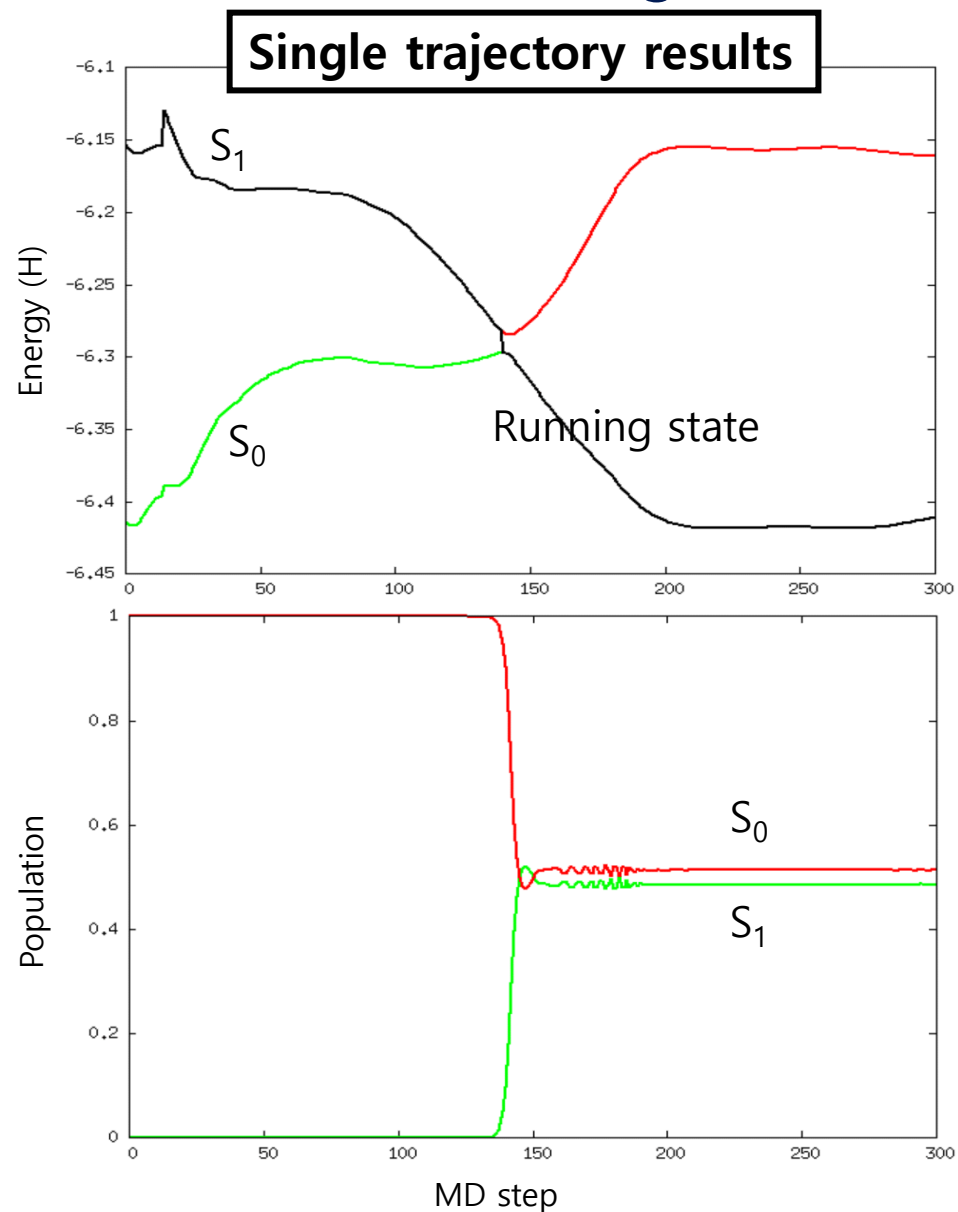
```
# Execute the simulation.
```

```
md.run(qm=qm)
```

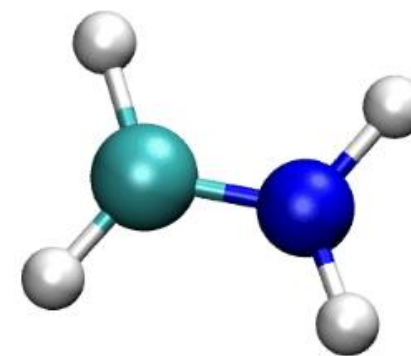
- Make MD inputs and execute them as earlier.

Tutorial #4: SH(EDC) and SHXF dynamics of CNH4+ with DFTB/SSR

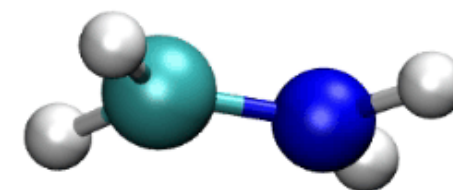
- Now, the rotation during a relaxation from S_1 to S_0 can be captured well.



Reflection



Rotation

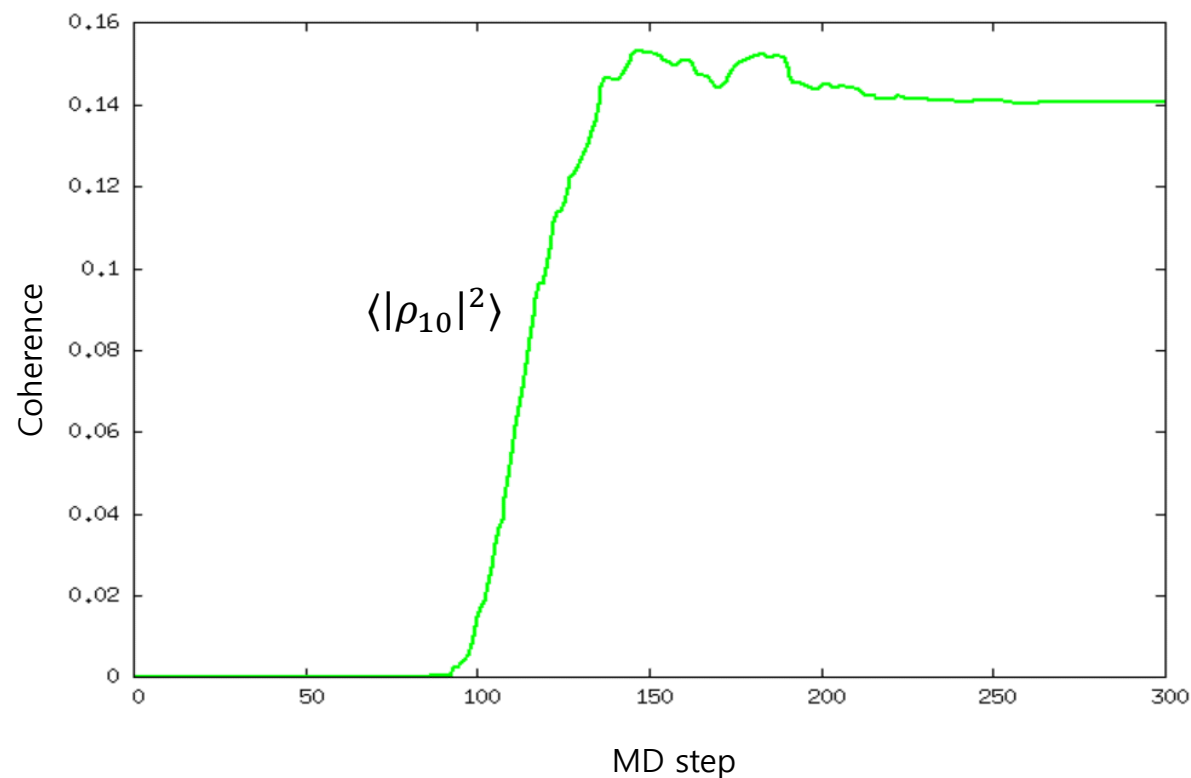
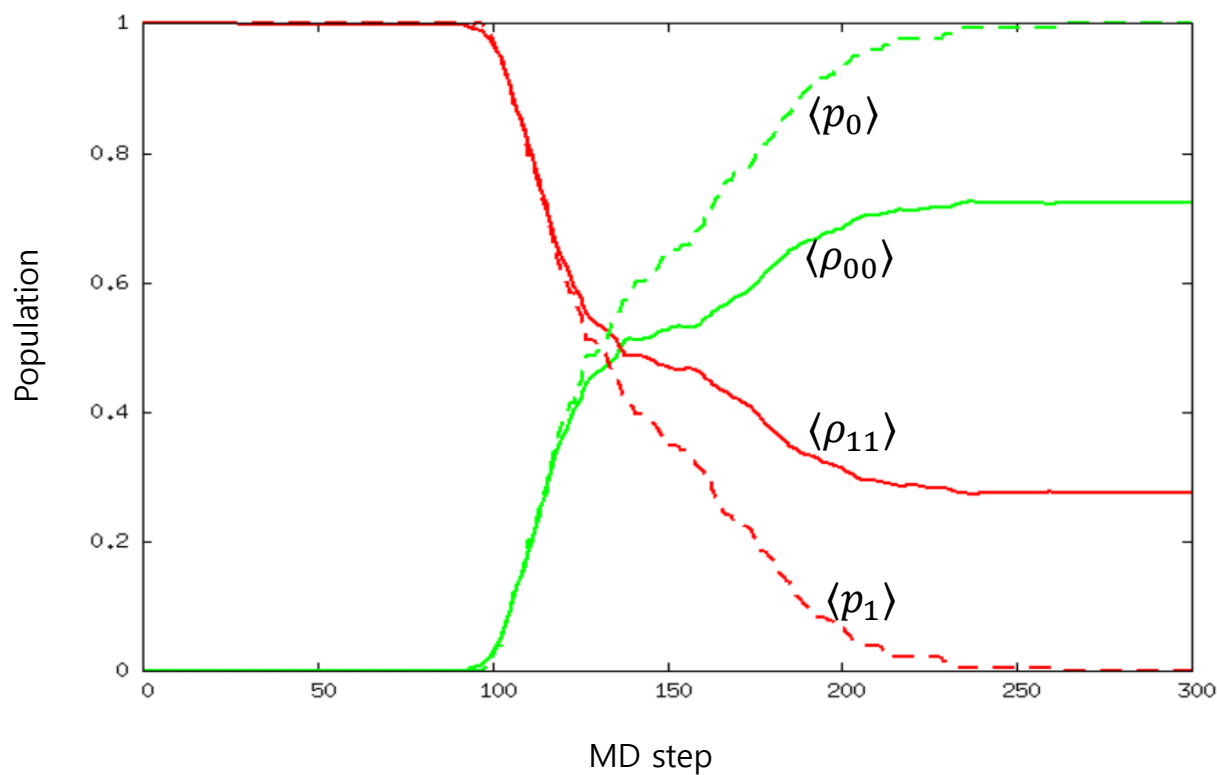


Tutorial #4: SH(EDC) and SHXF dynamics of CNH4+ with DFTB/SSR

- $S_1 \rightarrow S_0$ transition is observed with DFTB/SSR method, but the dynamics is still lack of decoherence.

Averaged results with 200 trajectories

$$\langle \rho_{ij} \rangle = \frac{1}{N_{\text{traj}}} \sum_I \rho_{ij}^I \quad \langle p_i \rangle = \frac{1}{N_{\text{traj}}} \sum_I N_i^I$$



- SHXF dynamics details

- DFTB/SSR provides with NACVs, so nuclear velocities are rescaled along the NACV corresponding to the transition.
- The width of a nuclear wave packet is $\sigma = 0.07$ a.u.

Tutorial #4: SH(EDC) and SHXF dynamics of CNH4+ with DFTB/SSR

- Prepare a running script.

```
from molecule import Molecule
from thermostat import NHC
import qm, mqc

# Define the target system.
with open("geom.xyz", "r") as fp:
    geom = fp.read()
mol = Molecule(geometry=geom, nstates=2, charge=+1)

# Set QM method.
qm = qm.dftbplus.SSR(molecule=mol, tuning=[3.0, 3.0, 3.0], l_range_sep=True, l_state_interactions=True, version="20.1",
install_path="/user/username/_install_dftb_20.1/", sk_path=
"/user/username/ob2-1-1/base/")

thermo = NHC(time_scale=3.0, temperature=300.)

# Determine MD method.
md = mqc.SHXF(molecule=mol, thermostat=thermo, nsteps=300, nesteps=1000, dt=0.25, istate=1, elec_object="density", sigma=0.1)

# Execute the simulation.
md.run(qm=qm)
```

Change the paths accordingly!

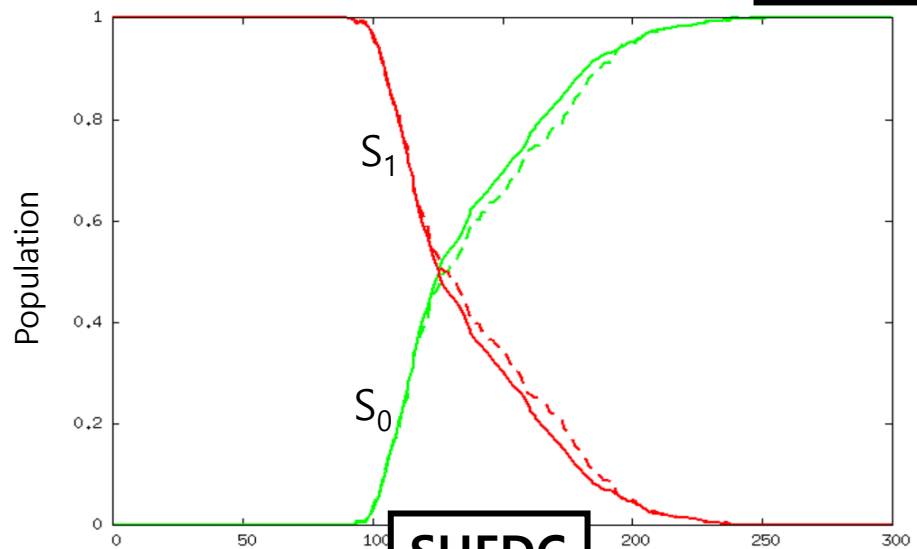
- Make MD inputs and execute them as earlier.

Tutorial #4: SH(EDC) and SHXF dynamics of CNH_4^+ with DFTB/SSR

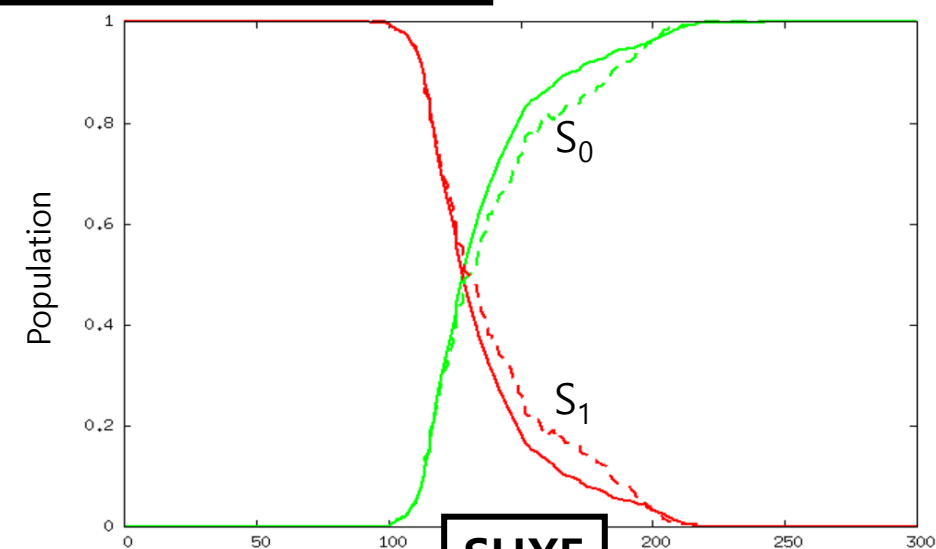
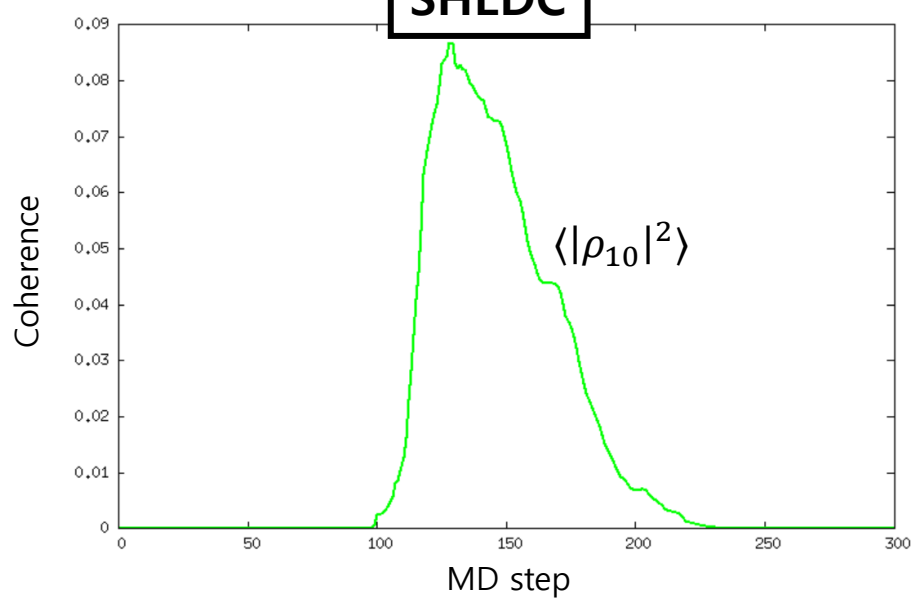
- Population/coherence dynamics in CNH_4^+ w/ decoherence

Averaged results with 200 trajectories

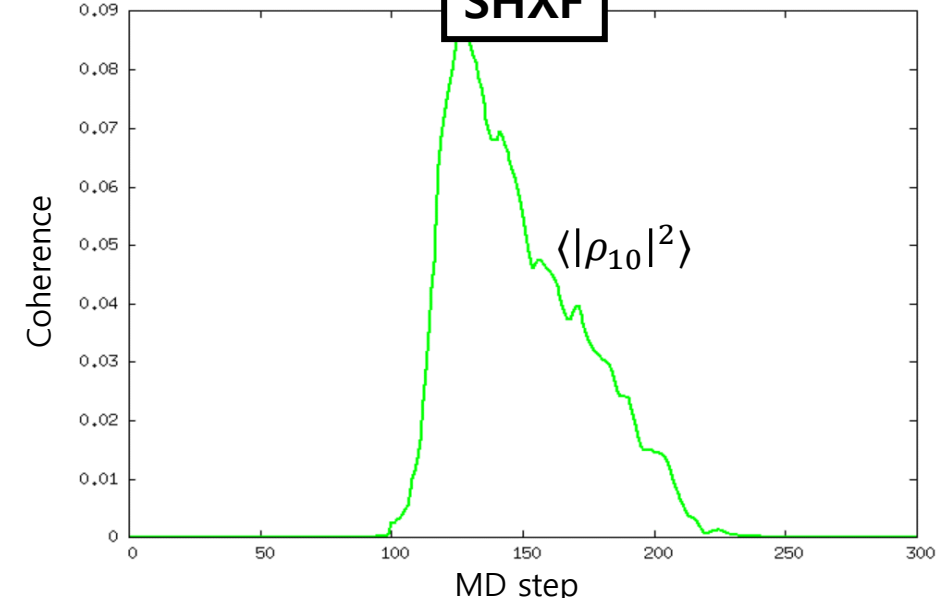
— $\langle \rho_{jj} \rangle$
- - - $\langle p_j \rangle$



SHEDC

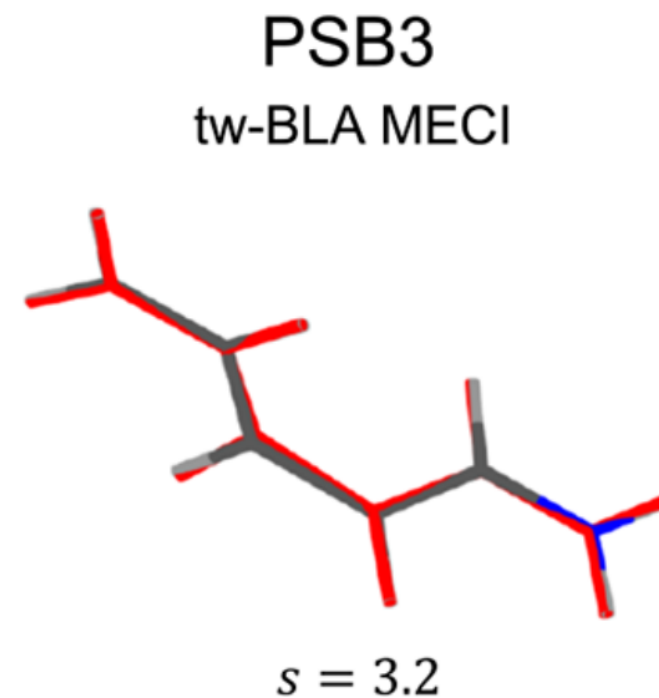


SHXF



● SHXF dynamics details

- $S_1 \rightarrow S_0$ dynamics for PSB3 are interested.
- The scaling constants for atomic spin constants has been changed to 3.2. A proper value is different according to the systems and conical intersections in question.
- SHXF dynamics of PSB3 is run during 300 fs; $dt=0.5$ fs.
- The width of a nuclear wave packet is $\sigma = 0.1$ a.u.



Tutorial #5: SHXF dynamics of PSB3 with DFTB/SSR

- Prepare a running script.

```
from molecule import Molecule
import qm, mqc

# Define the target system.
with open("geom.xyz", "r") as fp:
    geom = fp.read()
mol = Molecule(geometry=geom, nstates=2, charge=+1)

# Set QM method.
qm = qm.dftbplus.SSR(molecule=mol, tuning = [3.2, 3.2, 3.2], l_range_sep=True, l_state_interactions=True,
version="20.1",
install_path="/user/username/_install_dftb_20.1/", sk_path="/user/username/ob2-1-1/base/")

# Determine MD method.
md = mqc.SHXF(molecule=mol, nsteps=600, nesteps=1000, dt=0.5, istate=1, elec_object="density", sigma=0.1)

# Execute the simulation.
md.run(qm=qm)
```

Change the paths accordingly!

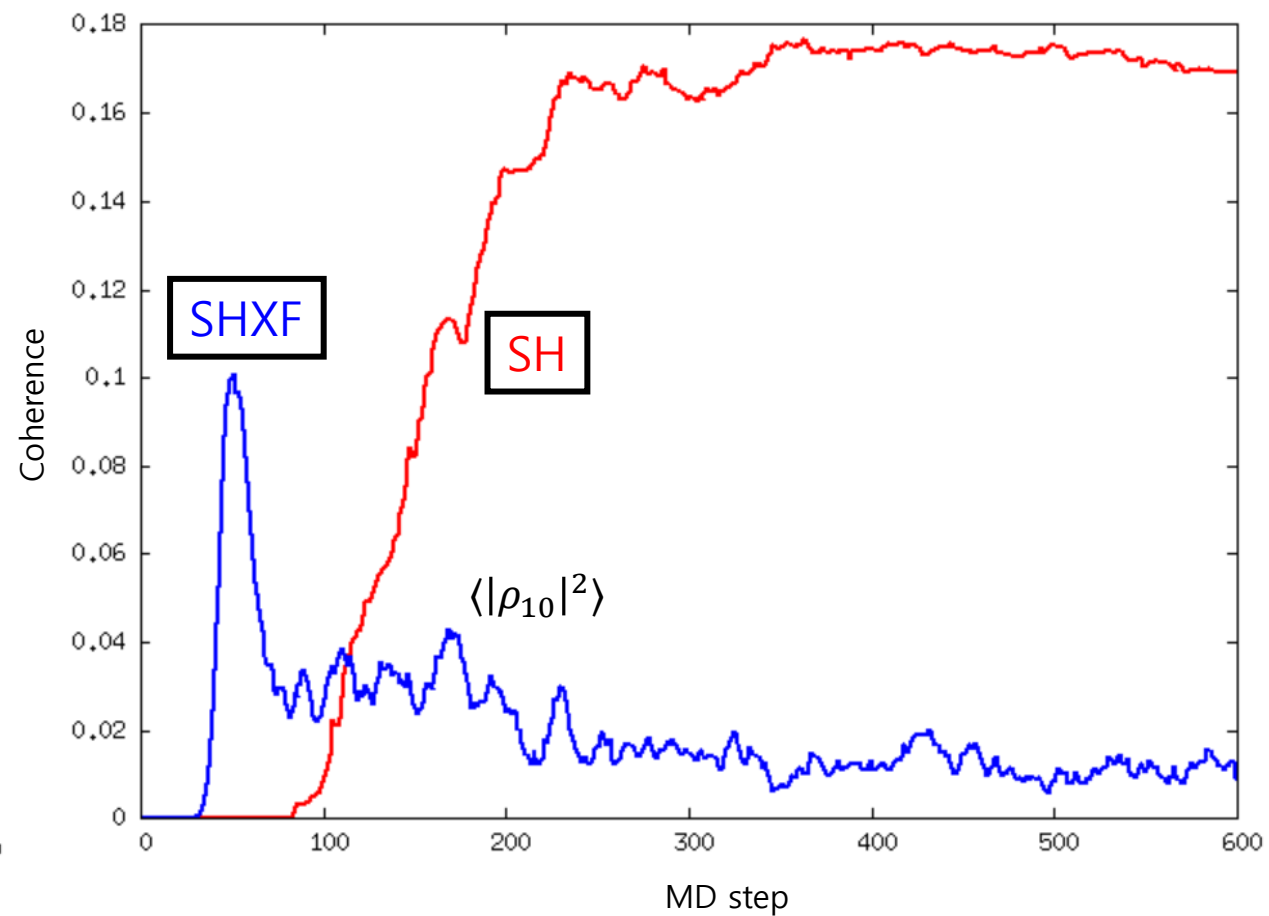
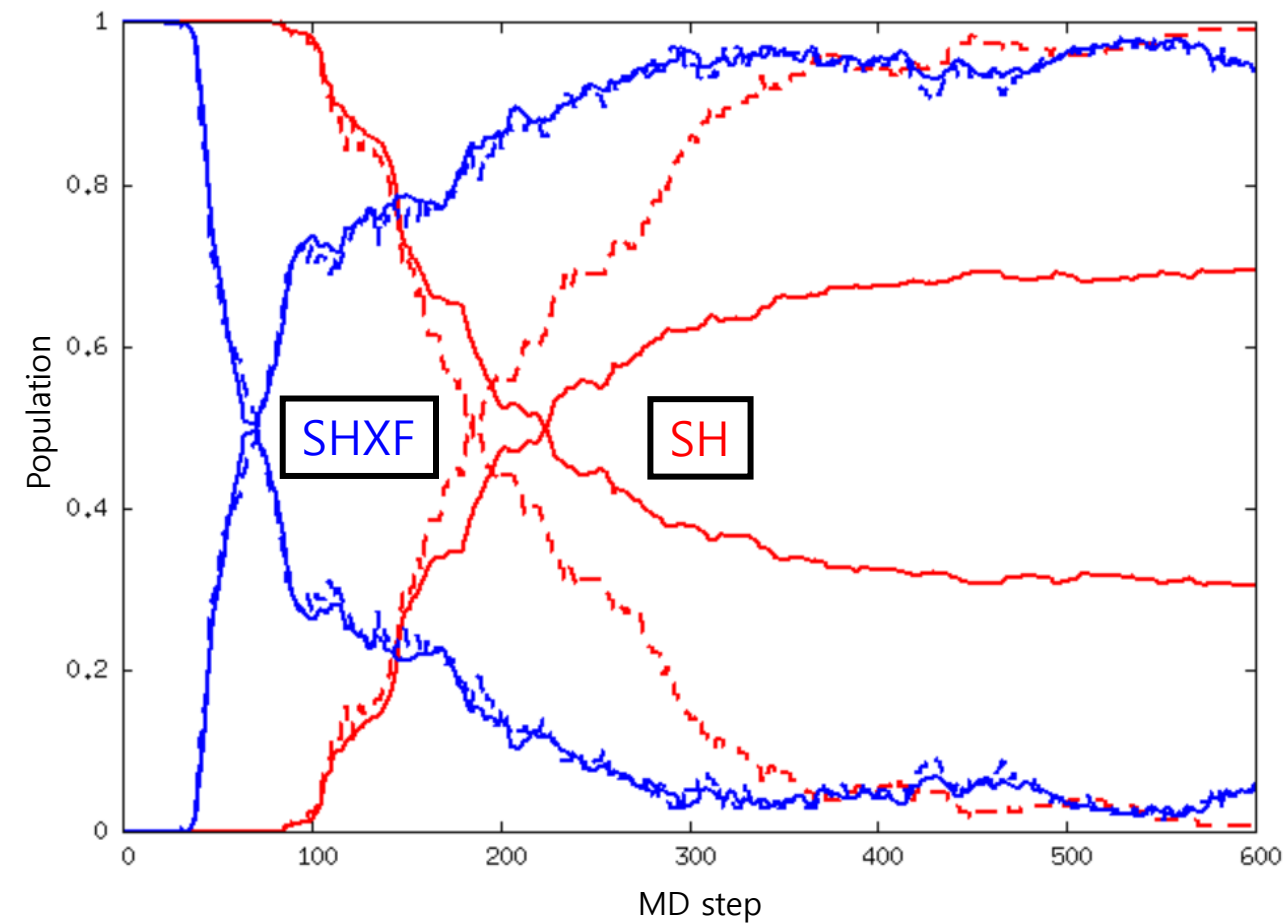
- Make MD inputs and execute them.

```
$ python input_gen.py -d /your-tut-dir/TUT1/02-dftb-BOMD-psb3/sampling -f run.py -n 200
$ python submit_all.py
```

Tutorial #5: SHXF dynamics of PSB3 with DFTB/SSR

- Population/coherence dynamics in PSB3 w/ decoherence

— $\langle \rho_{jj} \rangle$
- - - $\langle p_j \rangle$





Excited State Phenomena Computational Chemistry Lab.