Non-adiabatic Molecular Dynamics using Hefei-NAMD

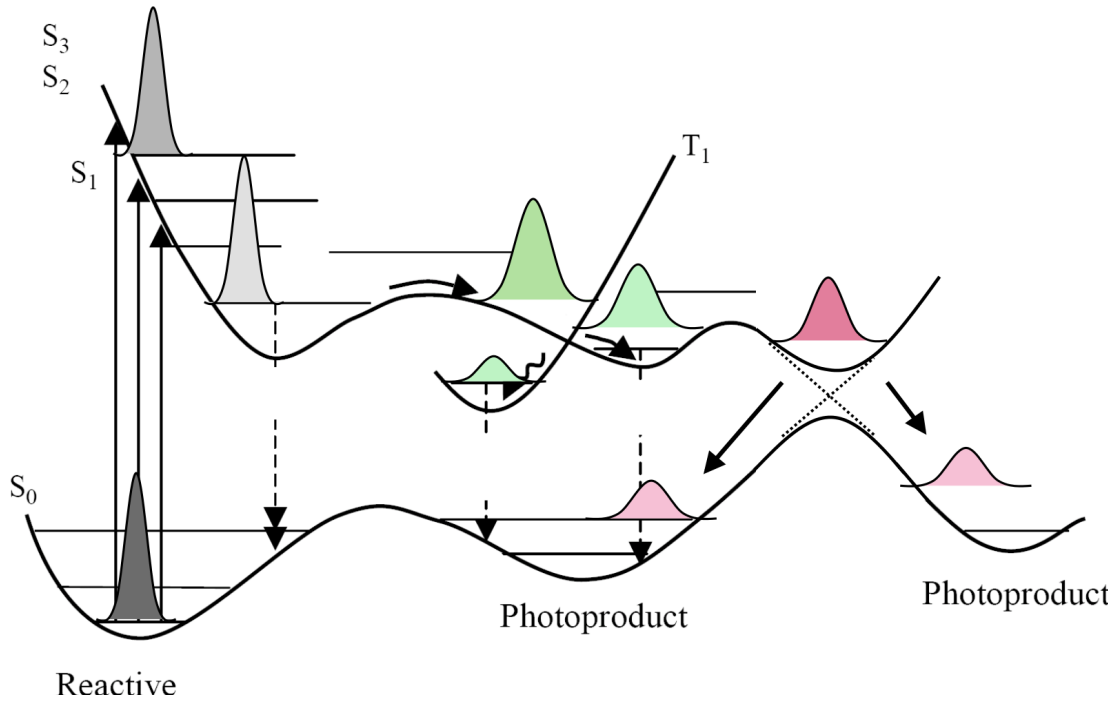
Qijing Zheng

Hefei National Laboratory for Physical Sciences at the Microscale
University of Science and Technology of China

Jun 10, 2018

Beyond Born-Oppenheimer Approximation

$$\Psi(\mathbf{r},\mathbf{R},t) = \Omega_j(\mathbf{R},t)\Phi_j(\mathbf{r};\mathbf{R}); \qquad \hat{\mathcal{H}}_{el}(\mathbf{r};\mathbf{R})\Phi_j(\mathbf{r};\mathbf{R}) = E_j(\mathbf{R})\Phi_j(\mathbf{r};\mathbf{R})$$



Method beyond BO approximation

Quantum Dynamics

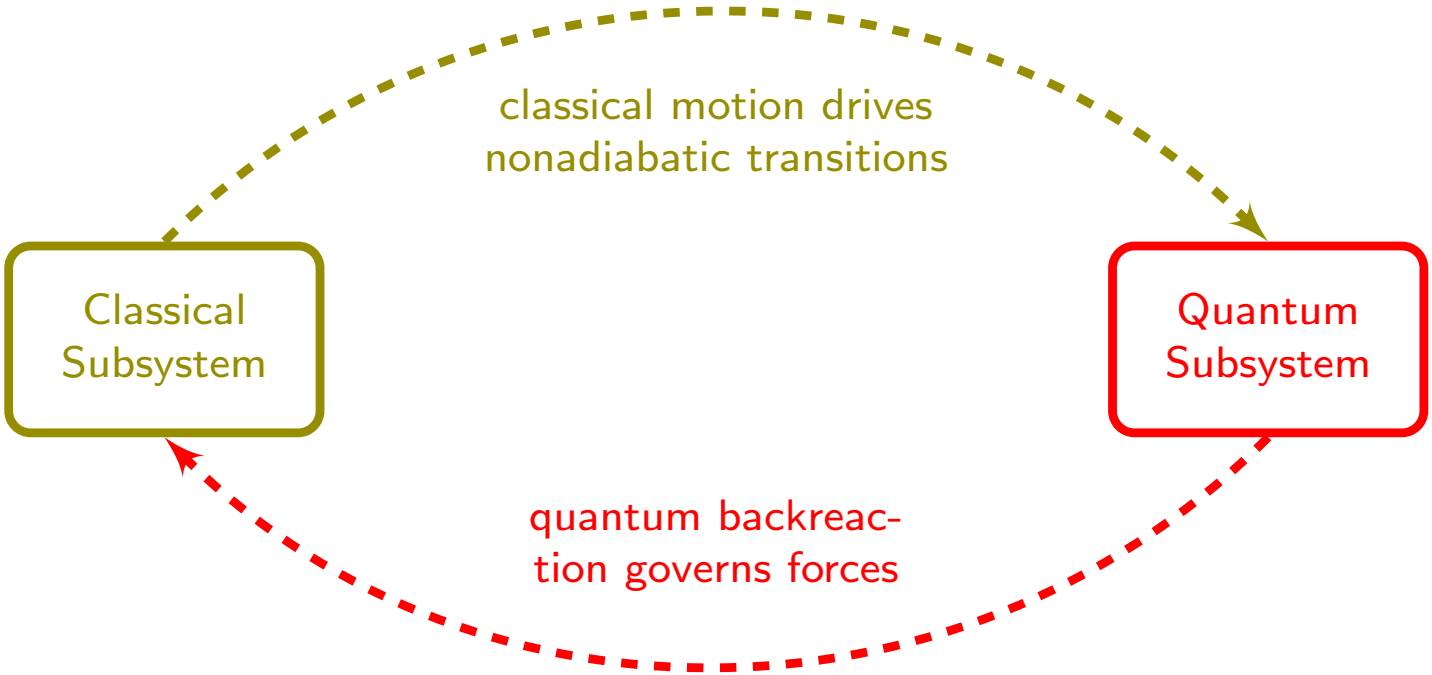
- MCTDH

Mixed Quantum-Classical

- Trajectory surface hopping (TSH)
- Ehrenfest dynamics
- ...

Mixed Quantum-Classical Dynamics

$$i\hbar\frac{\partial\Phi(\mathbf{r},\mathbf{R},t)}{\partial t} = \hat{\mathcal{H}}_{el}(\mathbf{r},\mathbf{R})\Phi(\mathbf{r},\mathbf{R},t); \qquad \Phi(\mathbf{r},\mathbf{R},t) = \sum_j C_j(t)\psi_j(\mathbf{r},\mathbf{R})$$



Surface Hopping

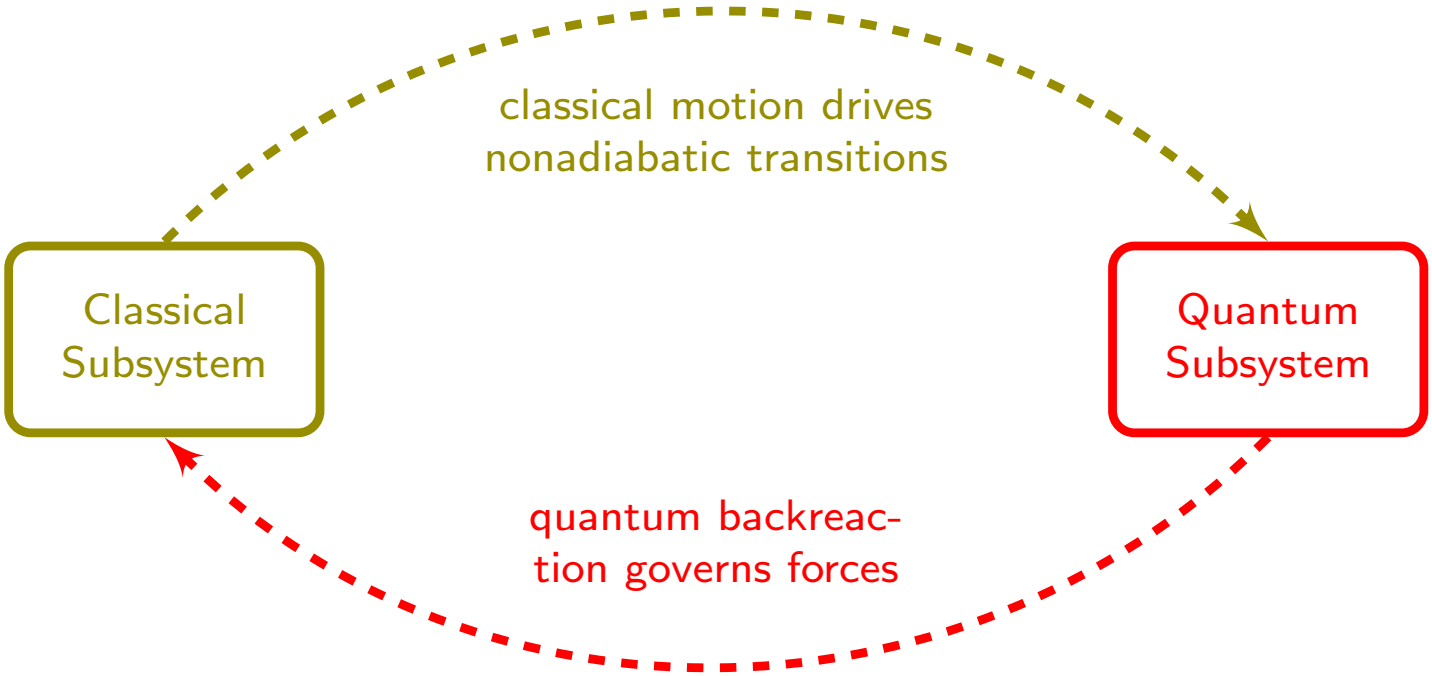
$$M_\alpha\ddot{\mathbf{R}}_\alpha = -\nabla_\alpha E_k^{el}(\mathbf{R})$$

Ehrenfest Dynamics

$$M_\alpha\ddot{\mathbf{R}}_\alpha = -\nabla_\alpha\langle\hat{\mathcal{H}}_{el}(\mathbf{r},\mathbf{R})\rangle$$

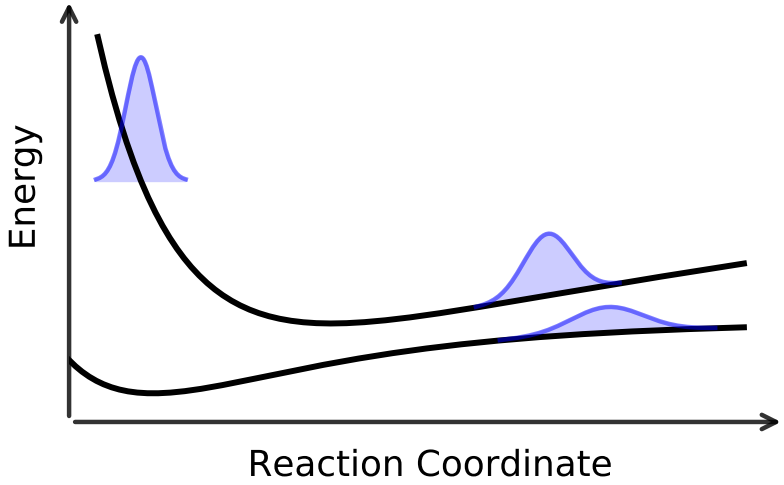
Mixed Quantum-Classical Dynamics

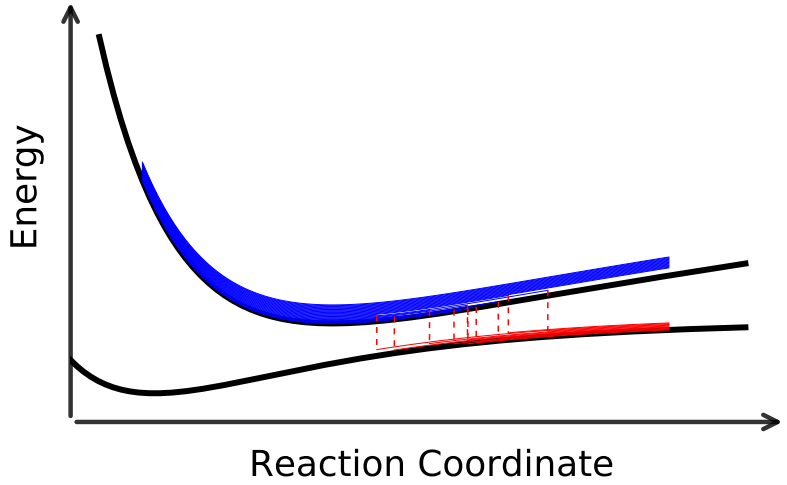
$$i\hbar\frac{\partial\Phi(\mathbf{r},\mathbf{R},t)}{\partial t} = \hat{\mathcal{H}}_{el}(\mathbf{r},\mathbf{R})\Phi(\mathbf{r},\mathbf{R},t); \qquad \Phi(\mathbf{r},\mathbf{R},t) = \sum_j C_j(t)\psi_j(\mathbf{r},\mathbf{R})$$



Surface Hopping

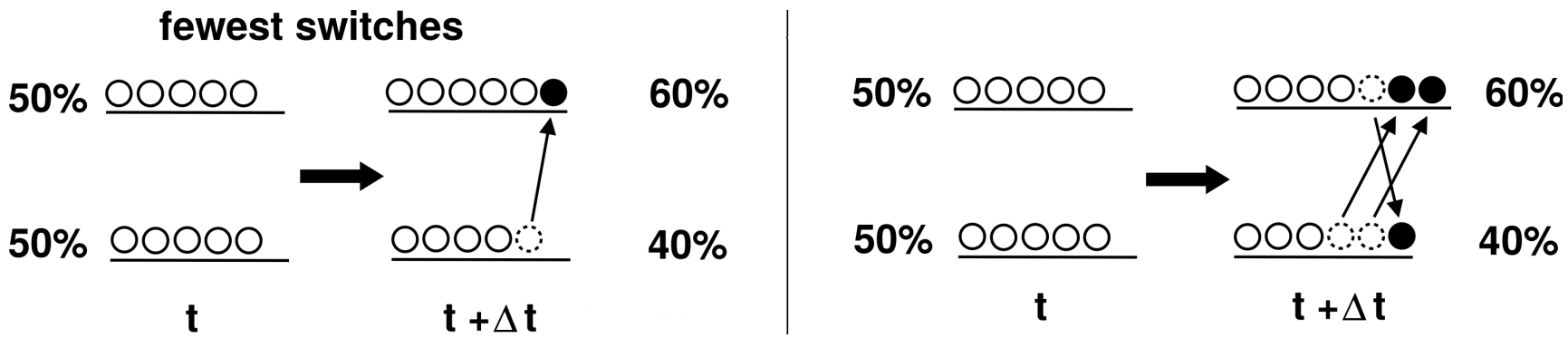
$$M_\alpha\ddot{\mathbf{R}}_\alpha = -\nabla_\alpha E_k^{el}(\mathbf{R})$$





Basic Idea

- Electronic wavefunction evolves according to time-dependent Schrödinger equation.
  - $\Phi(\mathbf{r},\mathbf{R},t)=\sum_j C_j(t)\psi_j(\mathbf{r},\mathbf{R}) \quad \Rightarrow \quad i\hbar\dot{C}_j(t)=\sum_k C_k(t)\left[H_{jk}-i\hbar\dot{\mathbf{R}}\cdot\mathbf{d}_{jk}\right]$
  - $H_{jk}=\langle\psi_j|\hat{\mathcal{H}}_{el}|\psi_k\rangle$
  - Non-adiabatic Couplings (NAC): $\mathbf{d}_{jk}=\langle\psi_j|\nabla_{\mathbf{R}}|\psi_k\rangle$
- An ensemble of independent nuclear trajectories is considered.
- Each trajectory propagates on one single potential energy surface at any given time.
  - $M_\alpha\ddot{\mathbf{R}}_\alpha=-\nabla_\alpha E_k^{el}(\mathbf{R})$
- Hops of trajectories between electronic states is possible.
  - Tully's Fewest-Switches algorithm et al.



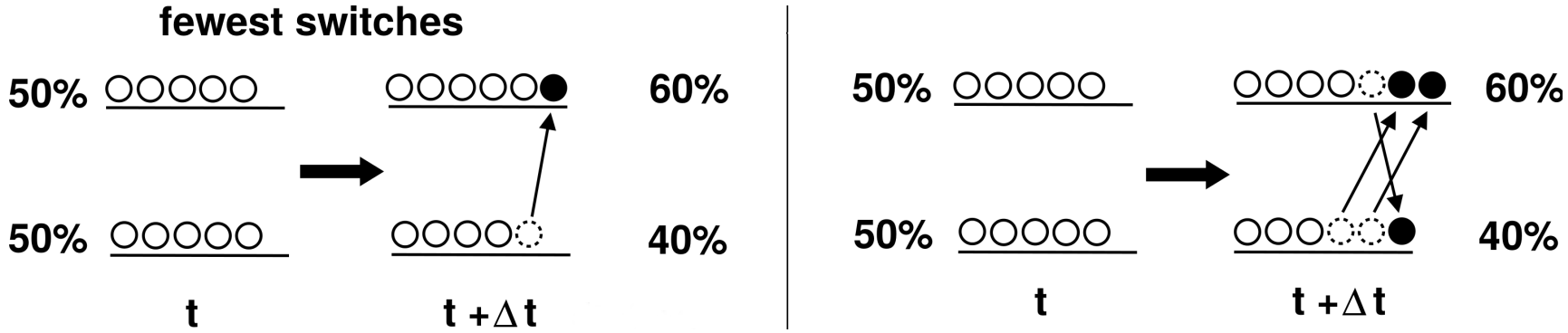
Assumptions

- Ensemble of independent trajectories have same coefficients $C_j(t)$.
- Internal consistency condition $N_j(t) \propto C_j^*(t) C_j(t) = \rho_{jj}(t)$.
- Hops from j to different $k \neq j$ are independent.
- Overall trajectory hops should be minimum.

Fewest-Switches: Hopping Probability

Transition from current state j to state $k \neq j$ is allowed only if population of state j is decreasing.

$$P_{jk}(t, \Delta t) = \max\left(- \frac{2 \int_t^{t+\Delta t} \mathrm{d}t \left[\hbar^{-1} \mathrm{Im}(\rho_{jk} H_{jk}) - \mathrm{Re}(\rho_{jk} \mathbf{d}_{jk} \cdot \dot{\mathbf{R}}) \right]}{\rho_{jj}}, 0 \right)$$

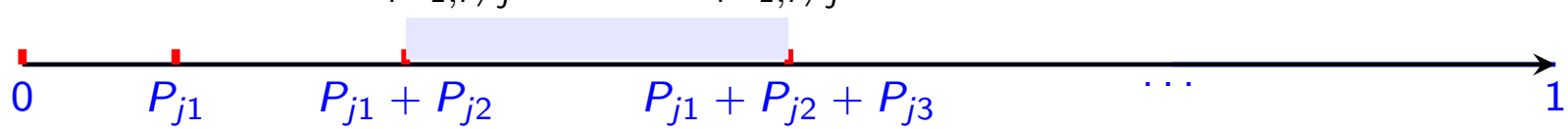


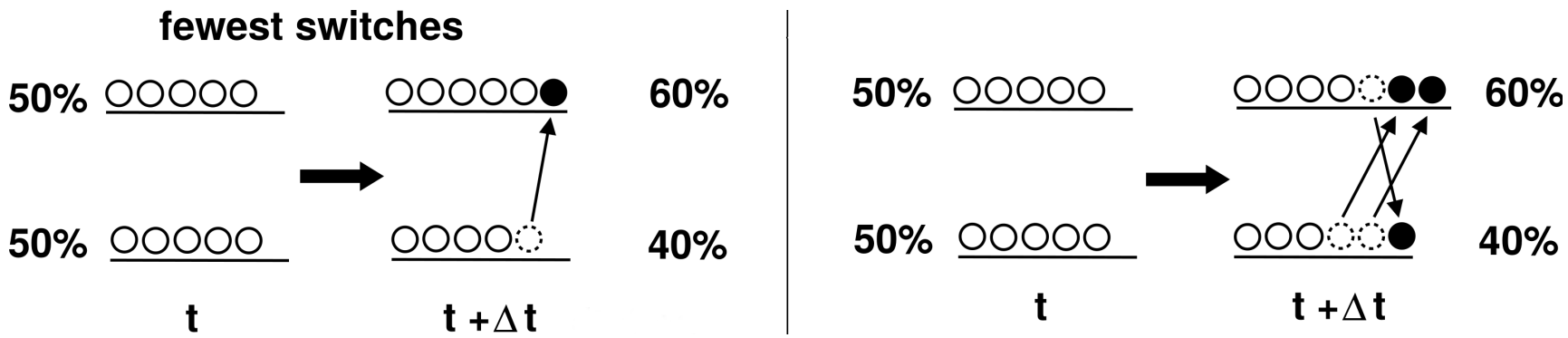
Assumptions

- Ensemble of independent trajectories have same coefficients $C_j(t)$.
- Internal consistency condition $N_j(t) \propto C_j^*(t) C_j(t) = \rho_{jj}(t)$.
- Hops from j to different $k \neq j$ are independent.
- Overall trajectory hops should be minimum.

Fewest-Switches: Which state to hop

$$\sum_{l=1, l\neq j}^{k-1} P_{jl} < \xi < \sum_{l=1, l\neq j}^{k} P_{jl} \quad \text{then} \quad j \to k$$





Assumptions

- Ensemble of independent trajectories have same coefficients $C_j(t)$.
- Internal consistency condition $N_j(t) \propto C_j^*(t) C_j(t) = \rho_{jj}(t)$.
- Hops from j to different $k \neq j$ are independent.
- Overall trajectory hops should be minimum.

Fewest-Switches: After trajectory hops

- Energy should be conserved after the hop e.g. by rescaling the nuclear velocity in the direction of non-adiabatic coupling.
  - $\mathrm{E}_j + \mathrm{K}_j = \mathrm{E}_k + \mathrm{K}_k \quad \Rightarrow \quad \mathbf{R}_\alpha^k = \mathbf{R}_\alpha^j - M_\alpha^{-1} \lambda \mathbf{d}_{jk}$
- Hops rejected when energy conservation not satisfied.

FSSH combined with TDDFT

single particle representation

$$\psi_p(\mathbf{r},t) = \sum_j c_j(t)\phi_j(\mathbf{r};\mathbf{R})$$

time-dependent Kohn-Sham equation

$$i\hbar\frac{\partial\psi_p(\mathbf{r},t)}{\partial t} = \mathcal{H}(\mathbf{r};\mathbf{R})\psi_p(\mathbf{r},t)$$

$$i\hbar\dot{c}_j(t) = \sum_k c_k(t)(\varepsilon_k\delta_{jk} - i\hbar d_{jk})$$

$$d_{jk} = \langle\phi_j|\frac{\partial}{\partial t}|\phi_k\rangle$$

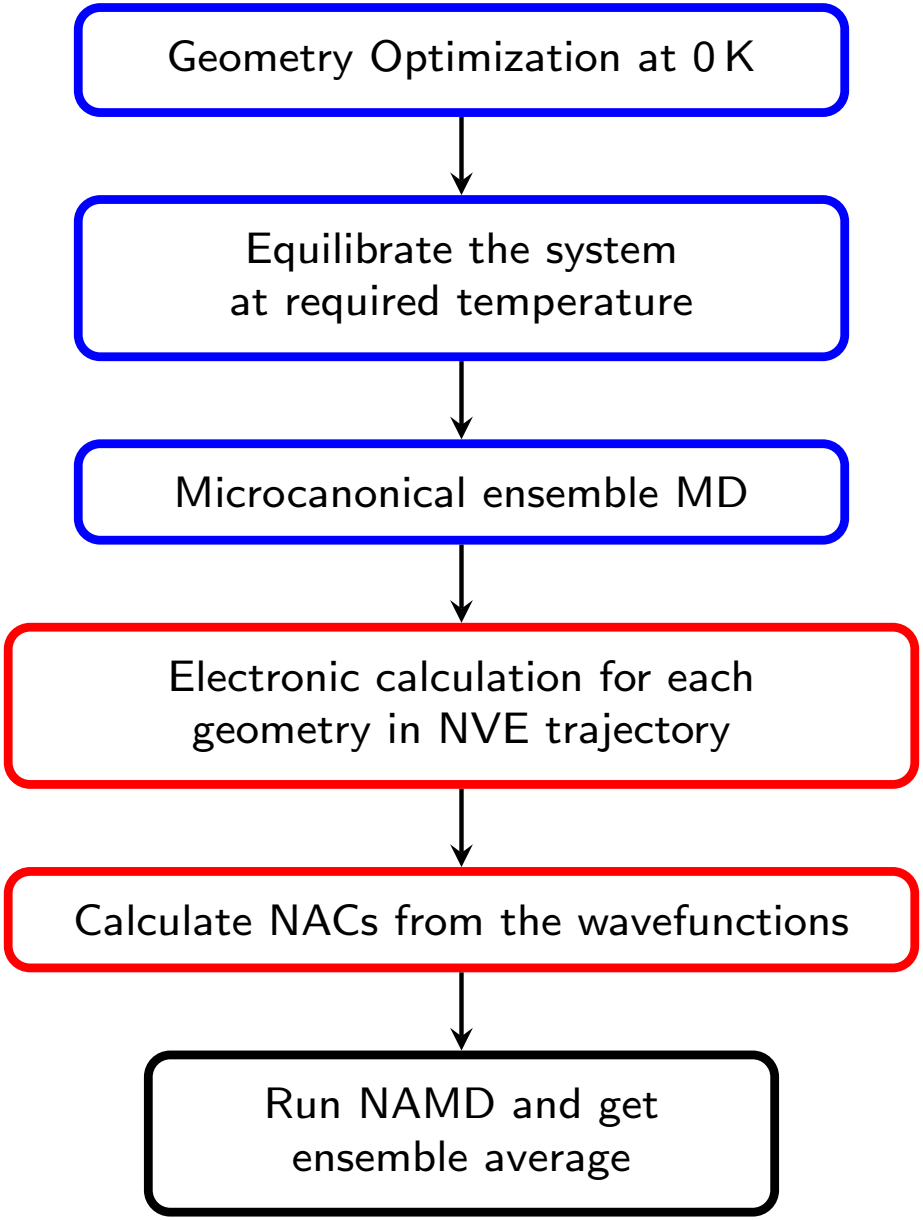
FSSH hopping probability

$$P_{jk}(t,\Delta t) = \max\left[\frac{2\int_t^{t+\Delta t}\mathrm{d}t\left[\mathrm{Re}(\rho_{jk}d_{jk})\right]}{\rho_{jj}}, 0\right]$$

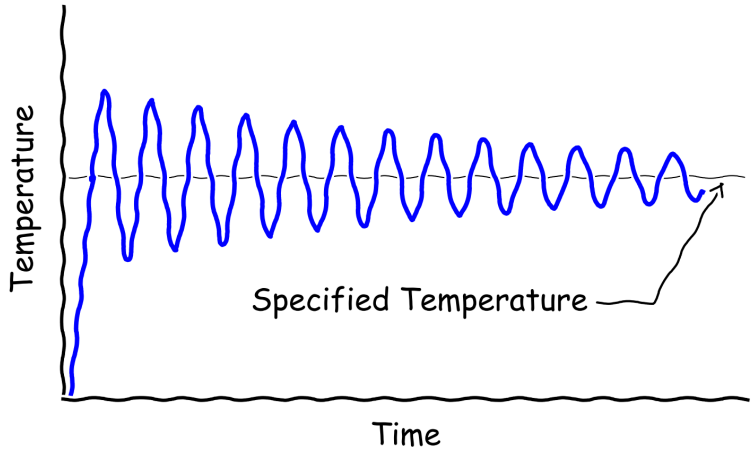
Classical-Path Aprroximation

$$G_{jk} = P_{jk}B_{jk}; \quad B_{jk} = \begin{cases}\exp\left\{-\frac{\varepsilon_j-\varepsilon_k}{k_BT}\right\}, & \text{if } \varepsilon_j > \varepsilon_k \\ 1, & \text{if } \varepsilon_j \leqslant \varepsilon_k\end{cases}$$

Work Flow


Geometry Optimization at 0 K
Equilibrate the system at required temperature
Microcanonical ensemble MD
Electronic calculation for each geometry in NVE trajectory
Calculate NACs from the wavefunctions
Run NAMD and get ensemble average

1. Geometry optimization: to find an energy minimum configuration.
   - Supercell should be large enough because BZ is only sampled at Γ point.
2. Molecular dynamics equilibration run — canonical ensemble,



   usually by velocity rescaling method, e.g. set `SMASS` $= -1$ in VASP.[1]
   - considered equilibrated when temperature fluctuation within $\pm 10\%$ of the specified temperature.
   - check the configuration before proceeding to next step.
3. Molecular dynamics production run — microcanonical ensemble.
   - The equilibrated configuration i.e. `CONTCAR` of the last step is used as input of this run.
   - The duration of MD is determined by the specific problem.

[1] https://cms.mpi.univie.ac.at/wiki/index.php/SMASS

Example INCARs for molecular dynamics using VASP

Example INCAR for equilibration run

```
General
  SYSTEM   = Your Job Name
  PREC     = Med
  ISPIN    = 1
  ISTART   = 0
  ICHARG   = 1
Ionic Relaxation
  ISIF     = 2
  EDIFFG   = -1E-2
Electronic relaxation
  NPAR     = 4
  ISMEAR   = 0
  SIGMA    = 0.1
  ALGO     = Fast
  NELMIN   = 4
  NELM     = 120
  EDIFF    = 1E-5
Molecular Dynamics
  ISYM     = 0      # turn off symmetry
  IBRION   = 0
  NSW      = 500    # No. of ionic steps
  POTIM    = 1      # time step 1.0 fs
  SMASS    = -1     # velocity rescaling
  NBLOCK   = 4      # velocity rescaled
                    # every NBLOCK step
  TEBEG    = 300    # start temperature
  TEEND    = 300    # end temperature
Writing Flag
  NWRITE   = 1      # make OUTCAR small
  LWAVE    = .FALSE.
  LCHARG   = .FALSE.
```

Example INCAR for production run

```
General
  SYSTEM   = Your Job Name
  PREC     = Med
  ISPIN    = 1
  ISTART   = 0
  ICHARG   = 1
Ionic Relaxation
  ISIF     = 2
  EDIFFG   = -1E-2
Electronic relaxation
  NPAR     = 4
  ISMEAR   = 0
  SIGMA    = 0.1
  ALGO     = Fast
  NELMIN   = 4
  NELM     = 120
  EDIFF    = 1E-6
Molecular Dynamics
  ISYM     = 0      # turn off symmetry
  IBRION   = 0
  NSW      = 5000   # NSW*POTIM fs
  POTIM    = 1      # time step 1.0 fs
  SMASS    = -3     # Microcanonical
  NBLOCK   = 1      # XDATCAR contains
                    # positions of each step
Writing Flag
  NWRITE   = 1      # make OUTCAR small
  LWAVE    = .FALSE. # WAVECAR not needed
  LCHARG   = .FALSE. # CHG not needed
```

1. Extract the atomic positions from `OUTCAR` or `XDATCAR` (`NBLOCK` should be 1). This can be easily done by a python script using ASE[2], e.g.

```
#!/usr/bin/env python
import os
from ase.io import read, write

# This may take a little while, since OUTCAR may be large for long MD run.
CONFIGS = read('/path/to/OUTCAR', format='vasp-out', index=':')

#########################################################################
# or you can read positions from XDATCAR, which is faster since it only contains
# the coordinates.  You may have to process the XDATCAR beforehand when VASP
# forget to write some lines. Executing the following line on shell prompt.
#########################################################################
# sed -i 's/^\s*$/Direct configuration=/' XDATCAR
#########################################################################
# CONFIGS = read('/path/to/XDATCAR', format='vasp-xdatcar', index=':')

NSW    = len(CONFIGS)                 # The number of ionic steps
NSCF   = 2000                         # Choose last NSCF steps for SCF calculations
NDIGIT = len("{:d}".format(NSCF))     #
PREFIX = './run/'                     # run directories
DFORM  = "/%%0%dd" % NDIGIT           # run dirctories format
for ii in range(NSCF):                # write POSCARs
    p = CONFIGS[ii - NSCF]
    r = (PREFIX + DFORM) % (ii + 1)
    if not os.path.isdir(r): os.makedirs(r)
    write('{:s}/POSCAR'.format(r), p, vasp5=True, direct=True)
```

SCF calculations II

2. With each extracted configuration, perfrom an SCF calculation to obtain the `WAVECAR` files for subsequent NAMD calculations. Example parallel job script for this run:

```
START=1          # start configuration
END=2000         # end configuration
for i in `seq ${START} ${END}`
do
    (( j = i - 8 ))
    ii=`printf "%04d" $i`
    jj=`printf "%04d" $j`
    if [[ -d "run/${ii}" ]]; then
        cd run/${ii}
        if [[ -f RUNNING || -f ENDED ]]; then  # Job still running or has ended, skip it
            cd ../..
            continue
        fi
        touch RUNNING
        echo "#### RUNNING in DIR: RUN/${ii}"
        sleep 0.4
        [[ -s ../${jj}/CHGCAR ]] && cp ../${jj}/CHGCAR .
        $OPEN_MPI $IB_FLAG -np $NCPUS -machinefile ${HOSTFILE} $VASP_EXEC   # VASP RUN
        if grep 'Total CPU' OUTCAR >& /dev/null; then
            touch ENDED
        else
            rm ENDED 2> /dev/null
        fi
        rm RUNNING CHG vasprun.xml
        cd ../..
    fi
done
```

[2] https://wiki.fysik.dtu.dk/ase/

NAMD calculations

1. Prepare input files for NAMD calculations, for example,

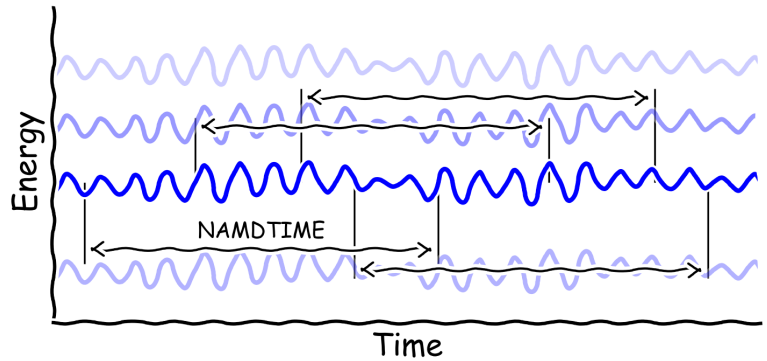
`inp`

```
&NAMDPARA
  BMIN      = 325     ! bottom band index
  BMAX      = 340     ! top band index
  NBANDS    = 388     ! number of bands

  NSW       = 2000
! number of ionic steps
  POTIM     = 1       ! MD time step
  TEMP      = 100     ! temperature

  NSAMPLE   = 100     ! number of samples
  NAMDTIME  = 1000    ! time for NAMD run
  NELM      = 1000    ! electron time step
  NTRAJ     = 5000    ! SH trajectories
  LHOLE     = .FALSE.! hole/electron SH

  RUNDIR    = "/path/to/run_dir/"
!  LCPEXT    = .TRUE.
/ ! NOTE THE SLASH HERE!
```

`INICON`

```
 87   329
519   330
 13   330
533   329
541   329
542   329
558   329
 59   329
 62   329
 65   329
...
```

- Column 1: initial time.
- Column 2: initial band index.
- No. of rows: `NSAMPLE` in `inp`.



2. The program will first calculate non-adiabatic couplings (NACs) and store the NACs in the file `COUPCAR`. If there is already the `COUPCAR`, then the program will read `COUPCAR` and perform NAMD calculatons.

The key quantity in the NAMD is the non-adiabatic couplings (NACs) and can be calculated by finite difference method.

$$\begin{aligned}\mathbf{d}_{jk} &= \langle\phi_j|\frac{\partial}{\partial t}|\phi_k\rangle = \langle\phi_j|\nabla_{\mathrm{R}}|\phi_k\rangle\cdot\dot{\mathbf{R}} \\ &= \frac{\langle\phi_j(t)|\phi_k(t+\Delta t)\rangle - \langle\phi_j(t+\Delta t)|\phi_k(t)\rangle}{2\Delta t}\end{aligned} \tag{1}$$

- $\phi_j(t)$ and $\phi_j(t+\Delta t)$ are Kohn-Sham (KS) pseudo-wavefunctions (WFC) from WAVECAR at different time steps. Note that $\phi_j(t)$ are not orthonormalized for PAW PS.
- For Gamma-only VASP, $\phi_j(t)$ are real functions. In addition, there is an arbitrary ± 1 phase factor for each KS-WFC, which may affect the evaluaton of Eq. (1).
- The inner product in the Eq. (1) can be obtained simply by summing over the products of plain-wave coefficients[3]

$$\langle\phi_j|\phi_k\rangle = \sum_G c_j^*(G)\cdot c_k(G) \tag{2}$$

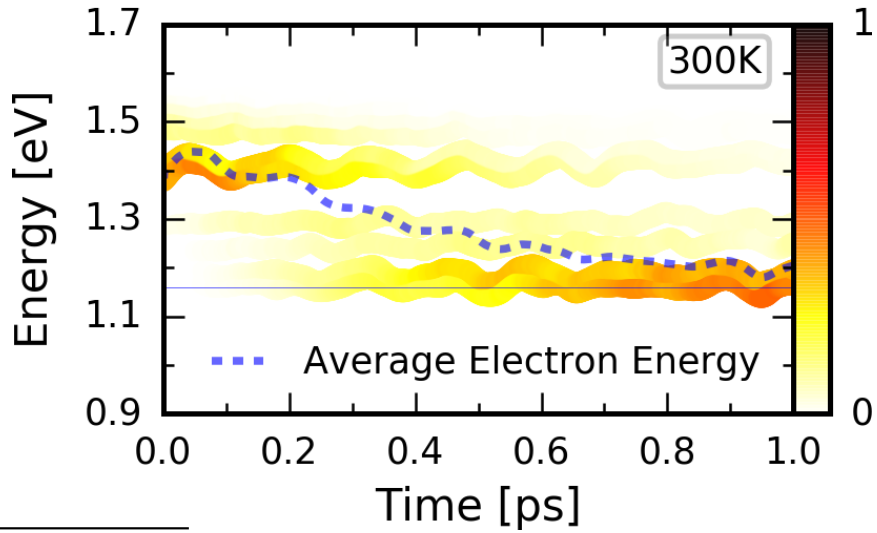
[3] For Gamma-only WFCs, the plain-wave coefficients other $G = 0$ are multiplied by a factor of $\sqrt{2}$.

Output files & postprocessing

- There are two types of output files: `SHPROP.*` and `PSICT.*`, where the suffix corresponds to the initial time in `INICON`, i.e. column 1.
  - `PSICT.*`: time-dependent propagation coefficients $c_i(t)$.
  - `SHPROP.*`: population of the selected states. The final results are obtained by averaging over these files.

```
##############################################################
#!/usr/bin/env python
import numpy as np
from glob import glob

inFiles = glob('/path/to/SHPROP.*')
outData = np.array([np.loadtxt(inf) for inf in inFiles])
np.savetxt('shprop_aver.dat', np.average(outData, axis=0))
##############################################################
```

  - The following plot shows the time evolution of the state populations.[4]



[4] script can be found here: https://github.com/QijingZheng/UsefulPythonSnippet/blob/master/poen_fssh.py

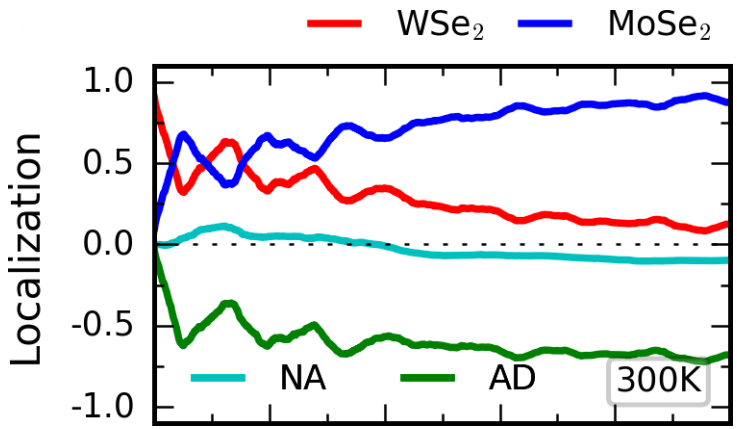
Define **charge localization** in a region V as

$$\begin{aligned}\int_V \rho_{\mathrm{PE}}(\mathbf{r},t)\,\mathrm{d}\mathbf{r} &= \int_V |\psi_{\mathrm{PE}}(\mathbf{r},t)|^2\,\mathrm{d}\mathbf{r} \\ &= \sum_{ij} c_i^*(t)c_j(t)\int_V \phi_i^*(\mathbf{r},\mathbf{R})\phi_j^*(\mathbf{r},\mathbf{R})\,\mathrm{d}\mathbf{r}\end{aligned}$$

The contribution to the change of the above term can be separated into two terms

$$\frac{\mathrm{d}\int_V \rho_{\mathrm{PE}}(\mathbf{r},t)\,\mathrm{d}\mathbf{r}}{\mathrm{d}t} = \sum_{ij}\left\{\frac{\mathrm{d}(c_i^*c_j)}{\mathrm{d}t}\int_V \phi_i^*\phi_j\,\mathrm{d}\mathbf{r} + c_i^*c_j\frac{\mathrm{d}\int_V \phi_i^*\phi_j\,\mathrm{d}\mathbf{r}}{\mathrm{d}t}\right\}$$

Example plot showing the NA/AD contribution to ET.[5]



[5] script can be found here: https://github.com/QijingZheng/UsefulPythonSnippet/blob/master/spatial_localization_fssh.py

Compilation of Hefei-NAMD code

- The compilation of the code is very simple, no special libraries needed.

```
 ~/tmp/namd_compilation   34 files, 524Kb
 zqj@node100   0   17:21:33 Sat Jun 09 $ make clean
rm -f *.mod *.a namd
rm -f prec.o lattice.o wave.o fileio.o couplings.o hamil.o TimeProp.o SurfHop.o main.o namd

 ~/tmp/namd_compilation   15 files, 100Kb
 zqj@node100   0   17:21:36 Sat Jun 09 $ make
gfortran -g -O2 -c prec.f90
gfortran -g -O2 -c lattice.f90
gfortran -g -O2 -c wave.f90
gfortran -g -O2 -c fileio.f90
gfortran -g -O2 -c couplings.f90
gfortran -g -O2 -c hamil.f90
gfortran -g -O2 -c TimeProp.f90
gfortran -g -O2 -c SurfHop.f90
gfortran -g -O2 -c main.f90
gfortran -g -O2 -o namd prec.o lattice.o wave.o fileio.o couplings.o hamil.o TimeProp.o SurfHop.o main.o

 ~/tmp/namd_compilation   34 files, 524Kb
 zqj@node100   0   17:21:39 Sat Jun 09 $ ll namd
-rwxr-xr-x 1 zqj penguin 96682 Jun  9 17:21 namd
```

- Yunhai Li[6] from Southeast University also write an MPI parallelized version of the code.

[6] liyunhai1016@hotmail.com

Thank you!