University at Buffalo
*The State University of New York*

# *Excited States and Nonadiabatic Dynamics CyberTraining School/Workshop 2022*

**Alexey Akimov**

University at Buffalo, SUNY

July 5, 2022

# *Basic Concepts and Terminology of Nonadiabatic Dynamics*

## Wavefunction and selection of representation

University at Buffalo
The State University of New York

$|\Psi\rangle$ Abstract wavefunction

$\{|\boldsymbol{r}\rangle\}$ Position states (Hilbert space)

$\Psi(\boldsymbol{r}) = \langle \boldsymbol{r}|\Psi\rangle$ Wavefunction in a position representation – representation in the basis of position states

$\Psi(\boldsymbol{k}) = \langle \boldsymbol{k}|\Psi\rangle$ Likewise, the momentum space representation of a wavefunction

Indeed $\quad 1 = \int d\boldsymbol{r}'|\boldsymbol{r}'\rangle\langle\boldsymbol{r}'|$ Complete basis

$$|\Psi\rangle = \int d\boldsymbol{r}'|\boldsymbol{r}'\rangle\langle\boldsymbol{r}'|\Psi\rangle = \int d\boldsymbol{r}'\,|\boldsymbol{r}'\rangle\Psi(\boldsymbol{r}')$$

$\Psi(\boldsymbol{r}')$ is essentially an expansion coefficient in the basis of coordinate states $\{|\boldsymbol{r}\rangle\}$

Different Hilbert spaces:

$\Psi(\boldsymbol{r}) = \langle \boldsymbol{r}|\Psi\rangle$ only electrons

$\Psi(\boldsymbol{R}) = \langle \boldsymbol{R}|\Psi\rangle$ only nuclei

$\Psi(\boldsymbol{r},\boldsymbol{R}) = \langle \boldsymbol{r},\boldsymbol{R}|\Psi\rangle$ both electrons and nuclei

$\Psi_i(\boldsymbol{r}) = \langle \boldsymbol{r}|\Psi_i\rangle = \langle \boldsymbol{r}, i|\Psi\rangle$ electronic coordinates, i-th basis state

Density matrix operator

$$\hat{\rho} = |\Psi\rangle\langle\Psi|$$

# Shorthand notation. Adiabatic and diabatic representations

University at Buffalo
The State University of New York

$$|\boldsymbol{\psi}\rangle = \left(|\psi_1\rangle, |\psi_2\rangle, \dots, |\psi_N\rangle\right)$$

$$A = \langle\boldsymbol{\psi}|\hat{A}|\boldsymbol{\psi}\rangle \qquad A_{ij} = \langle\psi_i|\hat{A}|\psi_j\rangle$$

$$|\Psi(t)\rangle = |\boldsymbol{\psi}_{adi}(t)\rangle C_{adi}(t) = |\boldsymbol{\psi}_{dia}(t)\rangle C_{dia}(t)$$

Adiabatic (Hamiltonian is diagonal) Diabatic (NACs are exactly zero)

$$|\boldsymbol{\psi}_{adi}\rangle = |\boldsymbol{\psi}_{dia}\rangle U$$

$$H_{dia} = \langle\boldsymbol{\psi}_{dia}|\hat{H}_{el}|\boldsymbol{\psi}_{dia}\rangle \qquad H_{dia}U = SUH_{adi} \qquad H_{adi} = \langle\boldsymbol{\psi}_{adi}|\hat{H}_{el}|\boldsymbol{\psi}_{adi}\rangle$$

$$H_{adi} = U^{+}H_{dia}U = \widetilde{H}_{dia}$$

$$P_{adi} = \langle\boldsymbol{\psi}_{adi}|\hat{\rho}|\boldsymbol{\psi}_{adi}\rangle = \langle\boldsymbol{\psi}_{adi}|\boldsymbol{\psi}_{adi}\rangle C_{adi}C_{adi}^{+}\langle\boldsymbol{\psi}_{adi}|\boldsymbol{\psi}_{adi}\rangle = IC_{adi}C_{adi}^{+}I = C_{adi}C_{adi}^{+}$$

Basis

Coefficients in this basis (dynamical variables, not in the nHamiltonian)

$$|\boldsymbol{\psi}\rangle = (|\psi_1\rangle, |\psi_2\rangle, \dots, |\psi_N\rangle)$$

$$C = (c_1, c_2, \dots, c_N)^T$$

The diabatic basis is not necessarily orthonormal

$$\langle \boldsymbol{\psi}_{dia} | \boldsymbol{\psi}_{dia} \rangle = S$$

What it all means

$$|\boldsymbol{\psi}_{adi}\rangle C_{adi} = \sum_i \psi_i C_{adi,i}$$

Wavefunction is the same in all representations

$$|\Psi\rangle = |\boldsymbol{\psi}_{adi}\rangle C_{adi} = |\boldsymbol{\psi}_{dia}\rangle C_{dia}$$

Transformation

$$|\boldsymbol{\psi}_{adi}\rangle = |\boldsymbol{\psi}_{dia}\rangle U$$

One can then show:

$$C_{dia} = UC_{adi} \leftrightarrow C_{adi} = U^{-1}C_{dia} \leftrightarrow C_{adi} = U^{+}SC_{dia}$$

$$\langle \boldsymbol{\psi}_{adi} | \boldsymbol{\psi}_{adi} \rangle = U^{+} \langle \boldsymbol{\psi}_{dia} | \boldsymbol{\psi}_{dia} \rangle U = U^{+}SU = I$$

**nHamiltonian**

- level
- id
- nHamiltonian* parent
- vector<nHamiltonian*> children

- nnucl, nadi, ndia

- CMATRIX* ham_dia, nac_dia, hvib_dia
- CMATRIX* ham_adi, nac_adi, hvib_adi
- CMATRIX* ovlp_dia, time_overlap_dia
- CMATRIX* ovlp_adi, time_overlap_adi
- CMATRIX* basis_transform
- vector<CMATRIX*> dc1_adi, dc1_dia
- vector<CMATRIX*> d1ham_adi, d1ham_dia

- ampl_dia2adi
- ampl_adi2dia

University at Buffalo
The State University of New York

Hamiltonian matrix elements

$$H_{dia} = \langle \boldsymbol{\psi}_{dia} | \hat{H} | \boldsymbol{\psi}_{dia} \rangle$$

$$H_{adi} = \langle \boldsymbol{\psi}_{adi} | \hat{H} | \boldsymbol{\psi}_{adi} \rangle$$

Unitary (similarity) transformation

$$H_{adi} = U^{+} H_{dia} U = \widetilde{H}_{dia}$$

First-order derivative coupling

$$\boldsymbol{D}_{adi} \equiv \langle \boldsymbol{\psi}_{adi} | \boldsymbol{\nabla} \boldsymbol{\psi}_{adi} \rangle$$

$$\boldsymbol{D}_{dia} \equiv \langle \boldsymbol{\psi}_{dia} | \boldsymbol{\nabla} \boldsymbol{\psi}_{dia} \rangle$$

**nHamiltonian**

- CMATRIX* ham_dia, nac_dia, hvib_dia
- CMATRIX* ham_adi, nac_adi, hvib_adi
- CMATRIX* ovlp_dia, time_overlap_dia
- CMATRIX* ovlp_adi, time_overlap_adi
- CMATRIX* basis_transform
- vector<CMATRIX*> dc1_adi, dc1_dia
- vector<CMATRIX*> d1ham_adi, d1ham_dia
- compute_adiabatic()

How to compute NACs?

$$\begin{cases} U^{+} \nabla H_{dia} U - \left( \widetilde{D}_{dia}^{+} H_{adi} + H_{adi} \widetilde{D}_{dia} \right) = \nabla H_{adi} - \left( D_{adi}^{+} H_{adi} + H_{adi} D_{adi} \right) \\ \widetilde{\nabla H_{dia}} - \left( \widetilde{D}_{dia}^{+} \widetilde{H}_{dia} + \widetilde{H}_{dia} \widetilde{D}_{dia} \right) = \nabla H_{adi} - \left( D_{adi}^{+} H_{adi} + H_{adi} D_{adi} \right) \end{cases}$$

Then use special structure of the matrix

# Nonadiabatic couplings

University at Buffalo
The State University of New York

Properties of the NACs

$$\overline{\boldsymbol{D}}_{dia}^{+} + \overline{\boldsymbol{D}}_{dia} = \nabla S$$

$$\overline{\boldsymbol{D}}_{adi} + \overline{\boldsymbol{D}}_{adi}^{+} = \boldsymbol{\nabla} S_{adi} = 0 \rightarrow (D_{adi}^{\alpha})^{+} = -D_{adi}$$

This is a well-known property!

$$D_{adi}^{\alpha} = \widetilde{D}_{dia}^{\alpha} + U^{+} S \nabla_{\alpha} U$$

$D_{rep,ij}^{\alpha} \equiv \left\langle \psi_{rep,i} \middle| \nabla_{\alpha} \psi_{rep,j} \right\rangle$ is a scalar

$\boldsymbol{D}_{rep,ij} \equiv \left\langle \psi_{rep,i} \middle| \nabla \psi_{rep,j} \right\rangle$ understood as a column-vector

$\overline{\boldsymbol{D}}_{rep} \equiv \left\langle \boldsymbol{\psi}_{rep} \middle| \boldsymbol{\nabla} \boldsymbol{\psi}_{rep} \right\rangle$ understood as a vector of matrices $D_{rep}^{\alpha} = \left\langle \boldsymbol{\psi}_{rep} \middle| \nabla_{\alpha} \boldsymbol{\psi}_{rep} \right\rangle$

Important observations

the equation becomes an identity when $U = I$

$$\underbrace{\widetilde{\nabla_{\alpha} H_{dia}} - \left( (D_{adi}^{\alpha})^{+} \widetilde{H}_{dia} + \widetilde{H}_{dia} \widetilde{D}_{dia}^{\alpha} \right)}_{U^{+} \langle \boldsymbol{\psi}_{dia} | \nabla_{\alpha} H | \boldsymbol{\psi}_{dia} \rangle U} = \underbrace{\nabla_{\alpha} H_{adi} - \left( (D_{adi}^{\alpha})^{+} H_{adi} + H_{adi} D_{adi}^{\alpha} \right)}_{\langle \boldsymbol{\psi}_{adi} | \nabla_{\alpha} H | \boldsymbol{\psi}_{adi} \rangle}$$

## Quantum-classical Hamiltonian

University at Buffalo
The State University of New York

$$H_{qc} = T + E_{tot} = \sum_n \frac{p_n^2}{2m_n} + E_{tot}$$

For TSH (adiabatic) energies, just use the coefficients of special form $C = (0, 0, 1, \dots, 0)$

$$E_{tot} = \frac{\langle\Psi|H|\Psi\rangle}{\langle\Psi|\Psi\rangle} = \frac{C_{adi}^{+} H_{adi} C_{adi}}{C_{adi}^{+} C_{adi}} = \frac{C_{dia}^{+} H_{dia} C_{dia}}{C_{dia}^{+} S C_{dia}}$$

**nHamiltonian**

- Ehrenfest_energy_adi
- Ehrenfest_energy_dia

## Making a Quantum-classical transition. Way 1

$$i\hbar\frac{\partial|\Psi\rangle}{\partial t} = \hat{H}|\Psi\rangle$$

$$\hat{H}_{el}(\hat{\boldsymbol{R}}) = \sum_i |i\rangle H_{ii}(\hat{\boldsymbol{R}})\langle i| + \sum_{i,j;i\neq j} |i\rangle H_{ij}(\hat{\boldsymbol{R}})\langle j|$$

$$\hat{H}(\hat{\boldsymbol{R}},\hat{\boldsymbol{P}}) = \hat{T}(\hat{\boldsymbol{P}}) + \hat{H}_{el}(\hat{\boldsymbol{R}})$$

$$\hat{T}(\hat{\boldsymbol{P}}) = \sum_i \frac{1}{2}|i\rangle\hat{\boldsymbol{P}}^T M^{-1}\hat{\boldsymbol{P}}\langle i| = \sum_i -\frac{\hbar^2}{2}|i\rangle\nabla^T M^{-1}\nabla\langle i|$$

$$\hat{R}_i \rightarrow R_i$$

$$\hat{P}_i \rightarrow P_i$$

$$\hat{H}(\hat{\boldsymbol{R}},\hat{\boldsymbol{P}}) \rightarrow H^{qc}(\boldsymbol{R},\boldsymbol{P}) = \sum_i \frac{1}{2}|i\rangle[\boldsymbol{P}^T M^{-1}\boldsymbol{P} + H_{ii}(\boldsymbol{R})]\langle i| + \sum_{i,j;i\neq j}|i\rangle H_{ij}(\boldsymbol{R})\langle j|$$

$$i\hbar\sum_j S_{rep,ij}\frac{dC_{rep,j}}{dt} = \sum_j\left[H_{rep,ij} - i\hbar d_{rep,ij}\right]C_{rep,j}$$

Projecting onto electronic basis:

$$i\hbar S\frac{dC_{rep}}{dt} = \left(H_{rep} - i\hbar d_{rep}\right)C_{rep}$$

University at Buffalo
The State University of New York

## Making a Quantum-classical transition. Way 1. Contd.

Quantum-classical energy

$$H^{MF}(\boldsymbol{R},\boldsymbol{P};\Psi) = \frac{\langle\Psi|H^{qc}(\boldsymbol{R},\boldsymbol{P})|\Psi\rangle}{\langle\Psi|\Psi\rangle} = \frac{1}{2}\boldsymbol{P}^T M^{-1}\boldsymbol{P} + \frac{C_{adi}^{+}H_{adi}C_{adi}}{C_{adi}^{+}C_{adi}} = \frac{1}{2}\boldsymbol{P}^T M^{-1}\boldsymbol{P} + \frac{C_{dia}^{+}H_{dia}C_{dia}}{C_{dia}^{+}SC_{dia}}$$

$$\frac{dH^{MF}}{dt} = 0$$

$$i\hbar S\frac{dC_{rep}}{dt} = \left(H_{rep} - i\hbar d_{rep}\right)C_{rep}$$

$$\dot{\boldsymbol{R}} = M^{-1}\boldsymbol{P}$$

$$\dot{\boldsymbol{P}} = \boldsymbol{f}_{rep}^{MF}\left(\boldsymbol{R},\Psi_{\text{rep}}\right)$$

$$f_n^{MF} \equiv \frac{1}{C_{adi}^{+}C_{adi}}C_{adi}^{+}F_{adi,n}^{HF}C_{adi} = \frac{1}{C_{dia}^{+}SC_{dia}}C_{dia}^{+}F_{dia,n}^{HF}C_{dia}$$

$$F_{adi,n}^{HF} = -\langle\boldsymbol{\psi}_{adi}|\nabla_n H|\boldsymbol{\psi}_{adi}\rangle = \left[-\nabla_n H_{adi} + D_{adi,n}^{+}H_{adi} + H_{adi}D_{adi,n}\right]$$

# Ehrenfest dynamics

University at Buffalo
The State University of New York

$$i\hbar \frac{\partial|\Psi\rangle}{\partial t} = H|\Psi\rangle$$

**int rep_tdse**
- 0: diabatic representation
- 1: adiabatic representation [ default ]

By projection

$$i\hbar \frac{dC_{adi}}{dt} = H_{adi}C_{adi} - i\hbar \sum_n D_{adi,n} \frac{p_n}{m_n} C_{adi}$$

$$i\hbar S \frac{dC_{dia}}{dt} = H_{dia}C_{dia} - i\hbar \sum_n D_{dia,n} \frac{p_n}{m_n} C_{dia}$$

By requiring the energy conservation

**int rep_force;**
- 0: diabatic
- 1: adiabatic [ default ]

$$\dot{q}_n = \frac{p_n}{m_n}$$

$$\dot{p}_n = f_{adi,n}^{MF} \equiv \frac{1}{C_{adi}^{+}C_{adi}} C_{adi}^{+}F_{adi,n}^{MF}C_{adi}$$

CMATRIX Ehrenfest_forces_adi
CMATRIX Ehrenfest_forces_adi_unit

$$F_{adi,n}^{MF} = -\langle\psi_{adi}|\nabla_n H|\psi_{adi}\rangle$$
$$= \left[-\nabla_n H_{adi} + D_{adi,n}^{+}H_{adi} + H_{adi}D_{adi,n}\right]$$

vector<CMATRIX> Ehrenfest_forces_tens_adi

$$\dot{q}_n = \frac{p_n}{m_n}$$

$$\dot{p}_n = f_{dia,n}^{MF} \equiv \frac{1}{C_{dia}^{+}SC_{dia}} C_{dia}^{+}F_{dia,n}^{MF}C_{dia}$$

CMATRIX Ehrenfest_forces_dia
CMATRIX Ehrenfest_forces_dia_unit

$$F_{dia,n}^{MF} = -\langle\psi_{dia}|\nabla_n H|\psi_{dia}\rangle$$
$$= \left[-\nabla_n H_{dia} + D_{dia,n}^{+}S^{-1}H_{dia} + H_{dia}S^{-1}D_{dia,n}\right]$$

vector<CMATRIX> Ehrenfest_forces_tens_adi

University at Buffalo
The State University of New York

## Making a Quantum-classical transition. Way 2.

$$|i\rangle\langle j| \to \hat{a}_i^+ \hat{a}_j$$

$$\hat{H}(\hat{\boldsymbol{R}}, \hat{\boldsymbol{P}}, \{\hat{a}_i^+\}, \{\hat{a}_i\}) = \sum_i \left[\frac{1}{2}\hat{\boldsymbol{P}}^T M^{-1}\hat{\boldsymbol{P}} + H_{ii}(\hat{\boldsymbol{R}})\right] \hat{a}_i^+ \hat{a}_i + \sum_{i,j;i\neq j} H_{ij}(\hat{\boldsymbol{R}})\hat{a}_i^+ \hat{a}_j$$

$$\hat{a}_i = \frac{1}{\sqrt{2}}(\hat{q}_i + i\hat{p}_i) \qquad \hat{a}_i^+ \hat{a}_i \to \frac{1}{2}(\hat{q}_i^2 + \hat{p}_i^2 - 1)$$

$$\hat{a}_i^+ = \frac{1}{\sqrt{2}}(\hat{q}_i - i\hat{p}_i) \qquad \hat{a}_i^+ \hat{a}_j + \hat{a}_j^+ \hat{a}_i \to (\hat{q}_i\hat{q}_j + \hat{p}_i\hat{p}_j)$$

$$\hat{H}(\hat{\boldsymbol{R}}, \hat{\boldsymbol{P}}, \hat{\boldsymbol{q}}, \hat{\boldsymbol{p}}) = \sum_i \frac{1}{2}(\hat{q}_i^2 + \hat{p}_i^2 - 1)\left[\frac{1}{2}\hat{\boldsymbol{P}}^T M^{-1}\hat{\boldsymbol{P}} + H_{ii}(\hat{\boldsymbol{R}})\right] + \frac{1}{2}\sum_{i,j;i\neq j} H_{ij}(\hat{\boldsymbol{R}})(\hat{q}_i\hat{q}_j + \hat{p}_i\hat{p}_j)$$

$\hat{R}_i \to R_i \qquad \hat{P}_i \to P_i \qquad \hat{q}_i \to q_i \qquad \hat{p}_i \to p_i \qquad \Big\Downarrow$ Classicalize

$$H^{MMTS}(\boldsymbol{R}, \boldsymbol{P}, \boldsymbol{q}, \boldsymbol{p}) = \frac{1}{2}\boldsymbol{P}^T M^{-1}\boldsymbol{P} + \sum_i \frac{1}{2}(q_i^2 + p_i^2 - 1)H_{ii}(\boldsymbol{R}) + \frac{1}{2}\sum_{i,j;i\neq j} H_{ij}(\boldsymbol{R})(q_i q_j + p_i p_j)$$

Meyer-Miller-Thoss-Stock Hamiltonian

# Making a Quantum-classical transition. Way 2. Contd.

University at Buffalo
The State University of New York

More generally:

$$H^{MMTS}(\boldsymbol{R},\boldsymbol{P},\boldsymbol{q},\boldsymbol{p}) = \frac{1}{2}\boldsymbol{P}^T M^{-1}\boldsymbol{P} + \sum_i \frac{1}{2}\left(q_i^2 + p_i^2 - \gamma\right)H_{ii}(\boldsymbol{R}) + \frac{1}{2}\sum_{i,j;i\neq j} H_{ij}(\boldsymbol{R})\left(q_i q_j + p_i p_j\right)$$

For N-level system: $\gamma = \frac{2}{N}\left(\sqrt{N+1} - 1\right)$

The equations of motion are "classical"

$$\dot{\boldsymbol{R}} = \frac{\partial H^{MMTS}(\boldsymbol{R},\boldsymbol{P},\boldsymbol{q},\boldsymbol{p})}{\partial \boldsymbol{P}} = M^{-1}\boldsymbol{P}$$

$$\dot{\boldsymbol{P}} = -\frac{\partial H^{MMTS}(\boldsymbol{R},\boldsymbol{P},\boldsymbol{q},\boldsymbol{p})}{\partial \boldsymbol{R}} = -\sum_i \frac{1}{2}\left(q_i^2 + p_i^2 - \gamma\right)\nabla H_{ii}(\boldsymbol{R}) - \frac{1}{2}\sum_{i,j;i\neq j} \nabla H_{ij}(\boldsymbol{R})\left(q_i q_j + p_i p_j\right)$$

$$\dot{\boldsymbol{q}} = \frac{\partial H^{MMTS}(\boldsymbol{R},\boldsymbol{P},\boldsymbol{q},\boldsymbol{p})}{\partial \boldsymbol{p}} = H\boldsymbol{p}$$

$$\dot{\boldsymbol{p}} = -\frac{\partial H^{MMTS}(\boldsymbol{R},\boldsymbol{P},\boldsymbol{q},\boldsymbol{p})}{\partial \boldsymbol{q}} = -H\boldsymbol{q}$$

These equations are equivalent to TD-SE, just consider

$$C_{rep,i} = \frac{1}{\sqrt{2}}\left(q_i + ip_i\right)$$

University at Buffalo
The State University of New York

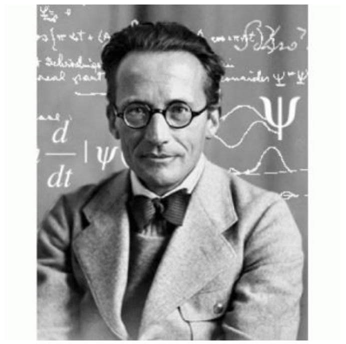
## *Overview of Dynamical Functions in Libra*

## What is Nonadiabatic Dynamics?

University at Buffalo
The State University of New York

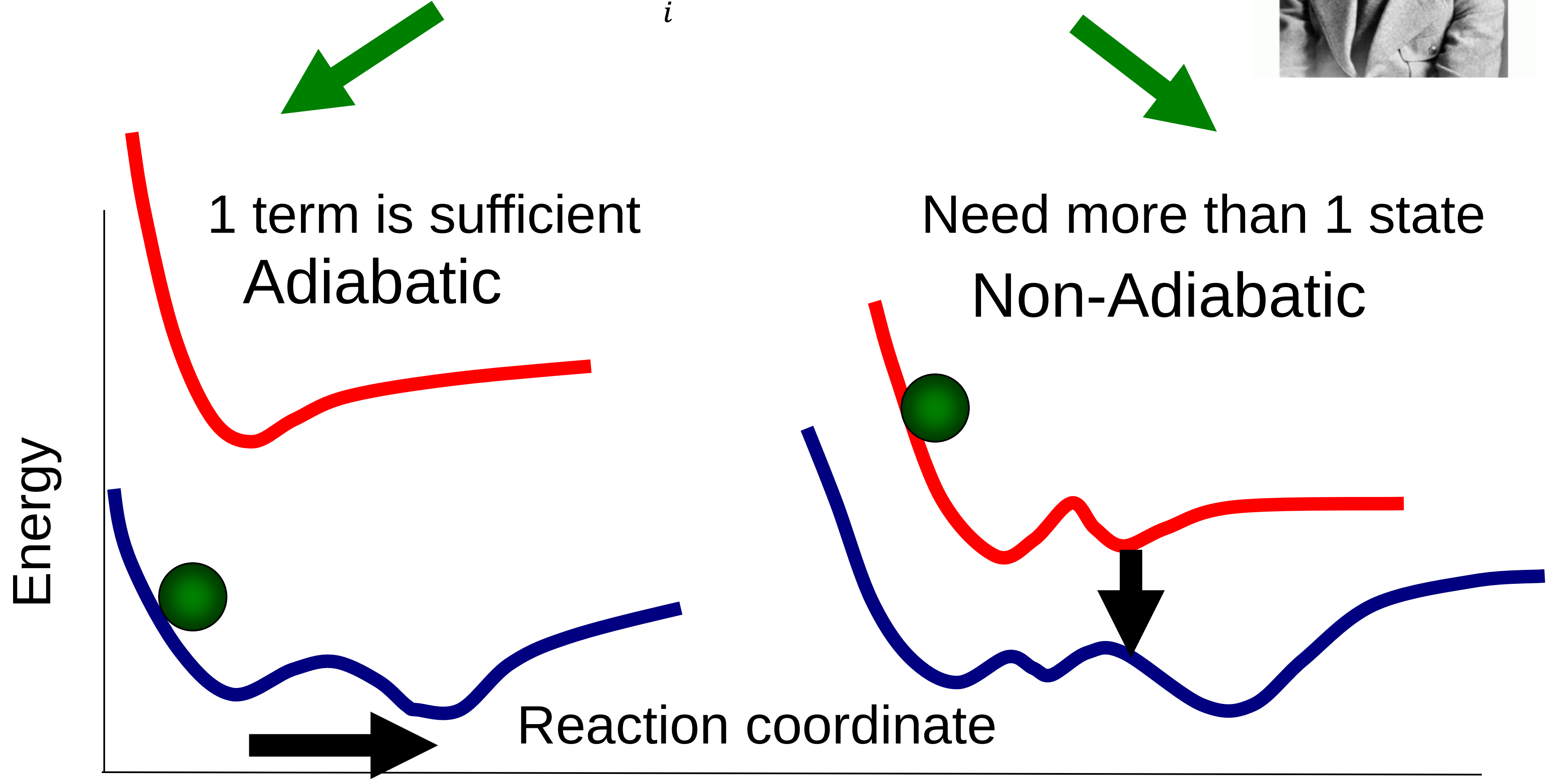
$$i\hbar \frac{\partial \Psi(\mathrm{r},\mathrm{R},\mathrm{t})}{\partial t} = H(\mathrm{r},\mathrm{R},\mathrm{t})\Psi(\mathrm{r},\mathrm{R},\mathrm{t})$$

$$\Psi(\mathrm{r},\mathrm{R},\mathrm{t}) = \sum_i \chi_i\big(t, R(t)\big)\Phi_{\mathrm{i}}(\mathrm{r};\mathrm{R(t)})$$



1 term is sufficient
Adiabatic

Need more than 1 state
Non-Adiabatic



## TSH in the nutshell

**Initialization**

**Nuclear dynamics**

$$\boxed{\dot{p}_i = -\frac{\partial H}{\partial r_i} \quad \dot{r}_i = \frac{\partial H}{\partial p_i}}$$

**Stationary adiabatic states**

$$\hat{H}_{el}\psi_i = E_i\psi_i$$

**Non-adiabatic Couplings**

$$\boxed{d_{ij} = \frac{\langle \psi_i(t) | \psi_j(t+dt) \rangle - \langle \psi_i(t+dt) | \psi_j(t) \rangle}{2dt}}$$

**Electronic Dynamics**

$$\Psi(\mathrm{r},\mathrm{R},\mathrm{t}) = \sum_i c_i(t)\psi_\mathrm{i}(\mathrm{r};\mathrm{R(t)}) \qquad i\hbar\frac{\partial \mathrm{c_i(t)}}{\partial t} = \sum_j \left(E_i\delta_{ij} - i\hbar d_{ij}\right) c_j$$

**Decoherence 1**

$$c_i \rightarrow c_i \exp\left(-\frac{\Delta t}{\tau_{ij}}\right), \forall i \neq j$$

as in SDM

**Proposed Hops**
**Decoherence 2**

$$P_{i\rightarrow j} = \Delta t * Re\left(\frac{2\frac{p}{m}d_{ij}c_i^* c_j}{c_i^* c_i}\right)$$

or as in DISH

**Accept Hops**

based on energy conservation or

$$P_{i\rightarrow f}^{A} = \min\left(1, exp\left(-\frac{\Delta E}{k_B T}\right)\right)$$

**Change state**

change active electronic state, rescale velocity

NBRA

## Libra as a workhorse of our developments

University at Buffalo
The State University of New York

Akimov *JCC,* **2016**, 37, 1626

Implemented in **Libra: https://quantum-dynamics-hub.github.io/libra/index.html**
**https://github.com/Quantum-Dynamics-Hub/libra-code**

Examples& Tutorials: **https://github.com/compchem-cybertraining**

Some of the implemented methods:

| **Methods** | **Paper** |
|---|---|
| Surface hopping schemes | Tully, J. C. *J. Chem. Phys.* **1990**, *93*, 1061 (FSSH); Wang, L., et al. *JCTC* **2014**, 10, 3598 (GFSH); Akimov, A. V. et al. *J. Phys. Soc. Jpn.* **2015**, 84, 094002 (MSSH) |
| Decoherence schemes | Granucci, G.; Persico, M. *J. Chem. Phys.* **2007**, *126*, 134114 (SDM); Nelson, T. et al. *J. Chem. Phys.* **2013**, *138*, 224111. (ID-A, ID-S); Jaeger, H. M. et al. *J. Chem. Phys.* **2012**, *137*, 22A545 (DISH) |
| Dephasing times calculations | Smith, B.; Akimov, A. V. *J. Chem.Phys.* **2019**, 151, 124107<br>Akimov, A. V.; Prezhdo, O. V. *J. Phys. Chem. Lett.* **2013**, *4*, 3857<br>Sifain, A. E.; Wang, L.; Tretiak, S.; Prezhdo, O. V.<br>Granucci, G.; Persico, M. *J. Chem. Phys.* **2007**, *126*, 134114. |
| Neglect of back-reaction (NBRA) | Prezhdo, O. V.; Duncan, W. R.; Prezhdo, V. V. *Prog. Surf. Sci.* **2009**, *84*, 30 |
| Boltzmann-corrected Ehrenfest | Bastida, A. et al. *Chem. Phys. Lett.* **2006**, *417*, 53<br>Smith, B.; Akimov, A. V. *J. Chem.Phys.* **2019**, 151, 124107 |
| Phase corrections | Akimov, A. V *J. Phys. Chem. Lett.* **2018**, *9*, 6096 |
| State tracking | Fernandez-Alberti, S.; et al. *J. Chem. Phys.* **2012**, *137*, 014512 (mincost); Temen, S.; AVA. *JPCL* **2021**, 12, 10587-10597 (stochastic) |
| Interfaces with ES codes | DFTB+ (Smith, B.; AVA *JPCL*. **2020**, 11, 1456), QE (Pradhan et al. *JPCM,* **2018**, 30, 484002), CP2K (Smith, B. A. et al. JCTC, 2021, 17, 678), Gaussian, GAMESS (Sato et al. *PCCP,* **2018**, 20, 25275) |
| Exact dynamics | Kosloff, D. and Kosloff, R. J. Chem. Phys. **1983**, *52*, 35-53 (SOFT); Colbert, D. T. and Miller, W. H. **1992**, *96*, 1982-1991 (Colert-Miller DVR) |
| HEOM | Temen et al. *Int. J. Quant. Chem.,* **2020**, 120, e26373 |

**int rep_ham;**
- 0: diabatic representation [ default ]
- 1: adiabatic representation

The primary way the Hamiltonian is computed in Python function

**int force_method;**
- 0: don't compute forces at all - e.g. we do not really need them
- 1: state-specific  as in the TSH or adiabatic (including adiabatic excited states) [ default ]
- 2: Ehrenfest

**int enforce_state_following;**
Wheather we want to enforce nuclear dynamics to be on a given state, regardlenss of the TSH transitions
Options:
- 0: no [ default ]
- 1: yes

on-the-fly NBRA

Note: only matters is `force_method == 1`

**int enforced_state_index;**
If we enforce the nuclear dynamics to be on a given state, what is the index of that state [any integer >- 0, default = 0 ]
The default value of 0 enforces the nuclear dynamics to be on the ground state. This is a convenient way of doing NBRA calculations  with model systems without the need for pre-computing the trajectories

$$S_{t,ij} = \langle \psi_i(t) | \psi_j(t+\Delta t) \rangle \longleftarrow$$

**int time_overlap_method;**

How do get the time-overlaps in the dynamics.

Options:

- 0: based on the wavefunctions (the Hamiltonian shall have the basis_transform variables updated) [ default ]
- 1: based on external calculations (the Hamiltonian shall have the time_overlap_adi member updated) - use for NBRA

**int nac_update_method;**

How to update NACs and vibronic Hamiltonian before electronic TD-SE propagation.

$$d = \sum_n D_{dia,n} \frac{p_n}{m_n} \longleftarrow$$

Options:

- 0: don't update them (e.g. for simplest NAC)
- 1: update according to changed momentum and existing derivative couplings [ default ]
- 2: update according to time-overlaps (only time-derivative NACs)

**int nac_algo**

How to compute time-derivative NACs

Options:

- -1: don't, e.g. we use NACs from somewhere else [ default ]
- 0: use HST formula (if nac_update_method==2)
- 1: use NPI of Meek and Levine (if nac_update_method==2)

## Phase correction

University at Buffalo
The State University of New York

**int do_phase_correction;**
Options:
- 0: no phase correction
- 1: according to our phase correction algorithm [ default ]

Akimov, A. V. *J. Phys. Chem. Lett.* **2018** 9, 6096-6102

$$d_{ij} = \frac{\langle \psi_i(t) | \psi_j(t+dt) \rangle - \langle \psi_i(t+dt) | \psi_j(t) \rangle}{2dt}$$

Hammes-Schiffer, S.; Tully, J. C. *J. Chem. Phys.* **1994**, *101*, 4657–4667

But states are defined only up to a complex phase!

**Phase correction:**

$$f_i = \frac{\langle \psi_i(t) | \psi_i(t') \rangle}{\sqrt{|\langle \psi_i(t) | \psi_i(t') \rangle|}}$$

$$|\tilde{\psi}_i(t')\rangle = f_i^* |\psi_i(t')\rangle$$

$$C_{adi,i} \rightarrow \tilde{C}_{adi,i} = f_i C_{adi,i}$$

**phase_correction_tol**
Min value of time-overlap we care to phase-correct

Indeed:

$$|\psi_i(t)\rangle = e^{i\phi(t)} |\chi_i(t)\rangle$$

$$|\psi_i(t')\rangle = e^{i\phi(t')} |\chi_i(t')\rangle$$

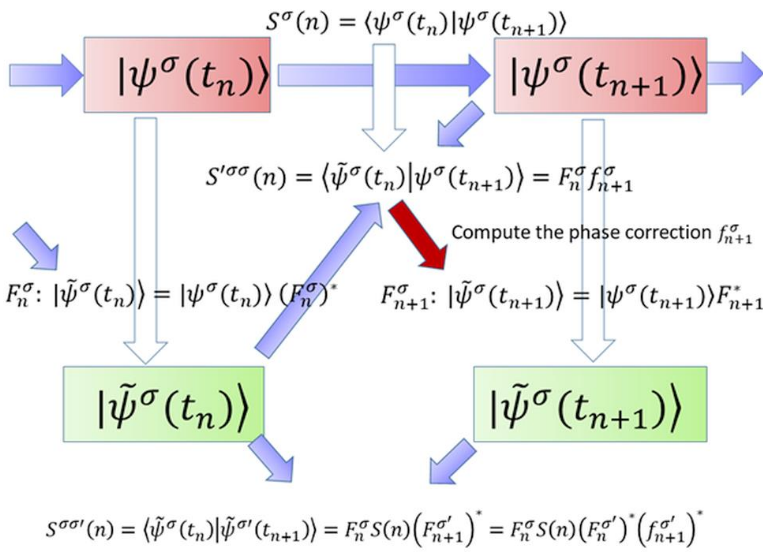
Then:

$$|\tilde{\psi}_i(t')\rangle = f_i^* |\psi_i(t')\rangle =$$
$$e^{-i[\phi(t')-\phi(t)]} e^{i\phi(t')} |\chi_i(t')\rangle = e^{i\phi(t)} |\chi_i(t')\rangle$$

Phase-correct other properties:

$$d_{ij}^{\alpha} = \langle \psi_i | \nabla_\alpha | \psi_j \rangle \rightarrow f_i \langle \psi_i | \nabla_\alpha | \psi_j \rangle f_j^*$$

Implementation in Libra:


$S^\sigma(n) = \langle \psi^\sigma(t_n) | \psi^\sigma(t_{n+1}) \rangle$
$|\psi^\sigma(t_n)\rangle$
$|\psi^\sigma(t_{n+1})\rangle$
$S'^{\sigma\sigma}(n) = \langle \tilde{\psi}^\sigma(t_n) | \psi^\sigma(t_{n+1}) \rangle = F_n^\sigma f_{n+1}^\sigma$
Compute the phase correction $f_{n+1}^\sigma$
$F_n^\sigma$: $|\tilde{\psi}^\sigma(t_n)\rangle = |\psi^\sigma(t_n)\rangle (F_n^\sigma)^*$
$F_{n+1}^\sigma$: $|\tilde{\psi}^\sigma(t_{n+1})\rangle = |\psi^\sigma(t_{n+1})\rangle F_{n+1}^*$
$|\tilde{\psi}^\sigma(t_n)\rangle$
$|\tilde{\psi}^\sigma(t_{n+1})\rangle$
$S^{\sigma\sigma'}(n) = \langle \tilde{\psi}^\sigma(t_n) | \tilde{\psi}^{\sigma'}(t_{n+1}) \rangle = F_n^\sigma S(n) \left(F_{n+1}^{\sigma'}\right)^* = F_n^\sigma S(n) \left(F_n^{\sigma'}\right)^* \left(f_{n+1}^{\sigma'}\right)^*$

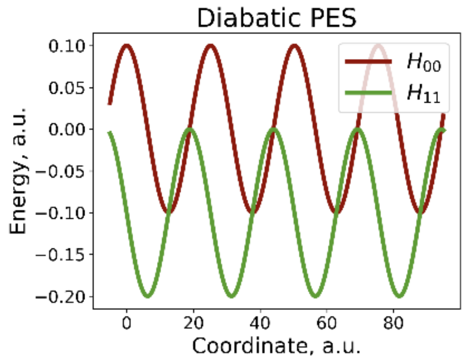
# Phase correction

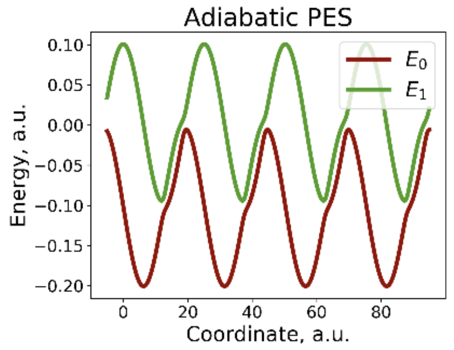
Akimov, A. V. *J. Phys. Chem. Lett.* **2018** 9, 6096-6102

$$H_{ii} = A_i \cos(\omega_i x + \delta_i) + B_i,$$

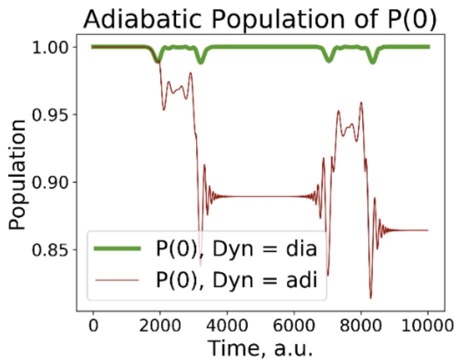
$$H_{ij} = V_{ij} = const.$$

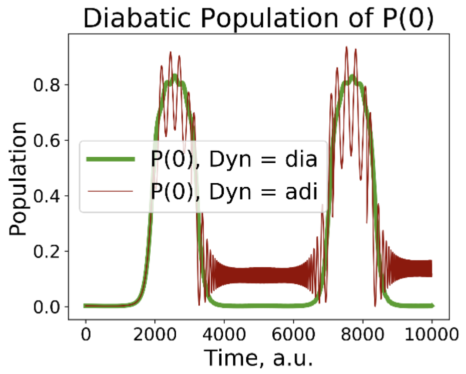
$$\Psi(t=0) = \psi_0$$

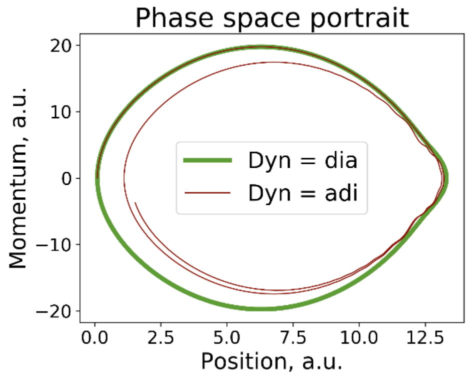




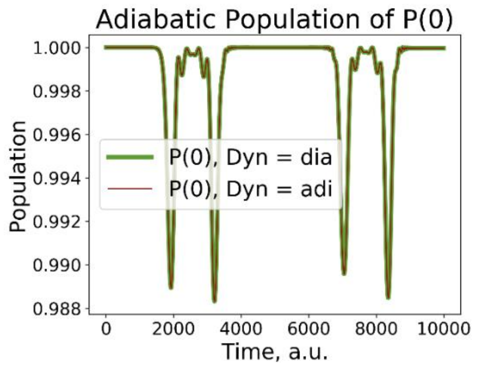
**No**

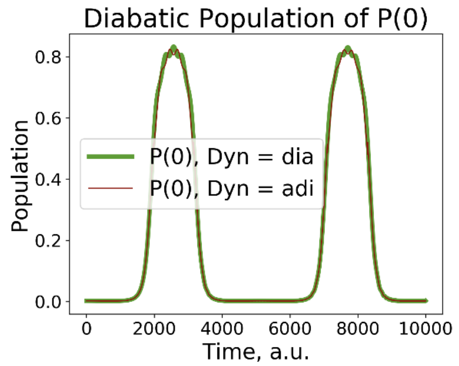


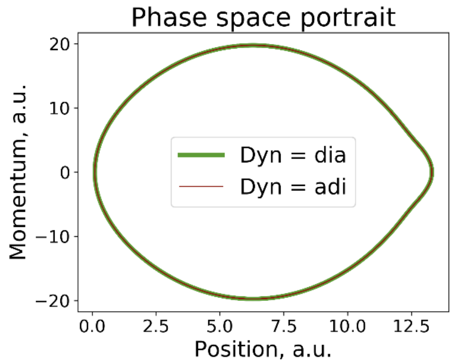




**Yes**

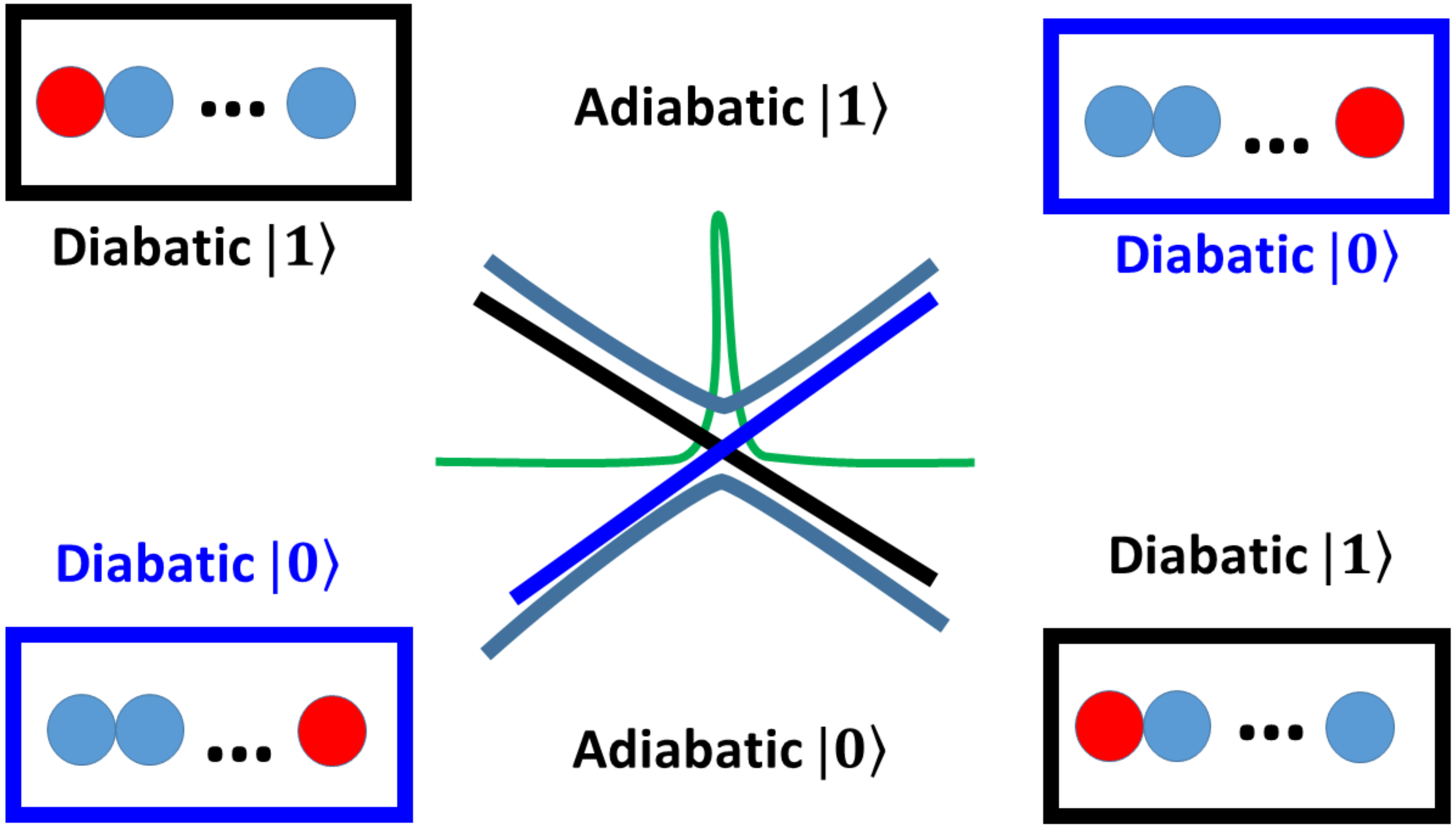




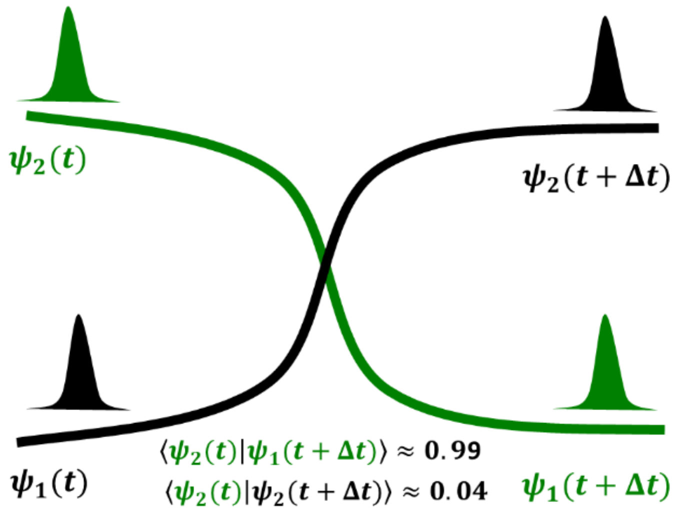


## State Tracking in NA-MD

Arises because of finite $\Delta t$ or due to inconsistency of energy and NAC (due to approximations)



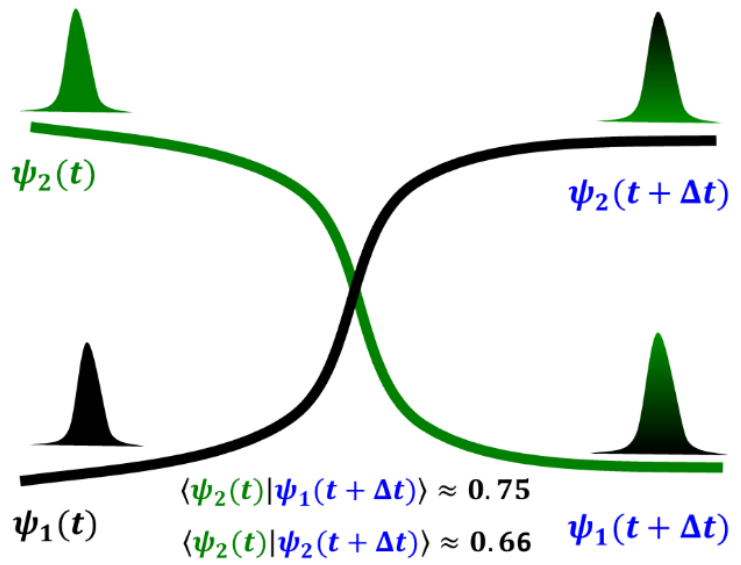
# State Tracking in NA-MD


$\psi_2(t)$
$\psi_2(t+\Delta t)$
$\psi_1(t)$
$\psi_1(t+\Delta t)$
$\langle\psi_2(t)|\psi_1(t+\Delta t)\rangle \approx 0.99$
$\langle\psi_2(t)|\psi_2(t+\Delta t)\rangle \approx 0.04$

State reassignment is easy:
[0, 1] → [1, 0]

- Fernandez-Alberti, S.; Roitberg, A. E.; Nelson, T.; Tretiak, S. *J. Chem. Phys.* **2012**, *137*, 014512.
- Wang, L.; Prezhdo, O. V. *J. Phys. Chem. Lett.* **2014**, *5*, 713
- Ryabinkin, I. G.; Nagesh, J.; Izmaylov, A. F. *J. Phys. Chem. Lett.* **2015**, *6*, 4200
- Qiu, J.; Bai, X.; Wang, L. *J. Phys. Chem. Lett.* **2018**, *9*, 4319

**Physics of the state swapping: adiabatic transition**


$\psi_2(t)$
$\psi_2(t+\Delta t)$
$\psi_1(t)$
$\psi_1(t+\Delta t)$
$\langle\psi_2(t)|\psi_1(t+\Delta t)\rangle \approx 0.75$
$\langle\psi_2(t)|\psi_2(t+\Delta t)\rangle \approx 0.66$

State reassignment is not clear:
[0, 1] → [??, ??]

Inspirator:
Prof. Ivan Infante

Story Temen

# Stochastic State Tracking

University at Buffalo
The State University of New York

Temen, S.; AVA *JPCL* **2021**, 12, 850-860

The idea:

$$S_{ij}\,(t,t+\Delta t) = \langle \psi_j(t+\Delta t) | \psi_i(t) \rangle$$ measures the two states' similarity

$$\left.\begin{aligned} |a_{j,i}|^2 &= |S_{ij}(t,t+\Delta t)|^2 \\ P_{i\to j} &= \frac{|S_{ij}(t,t+\Delta t)|^2}{\sum_k |S_{ik}(t,t+\Delta t)|^2} \end{aligned}\right\}$$ probability of the two "adiabatic" state transfer (state switch)

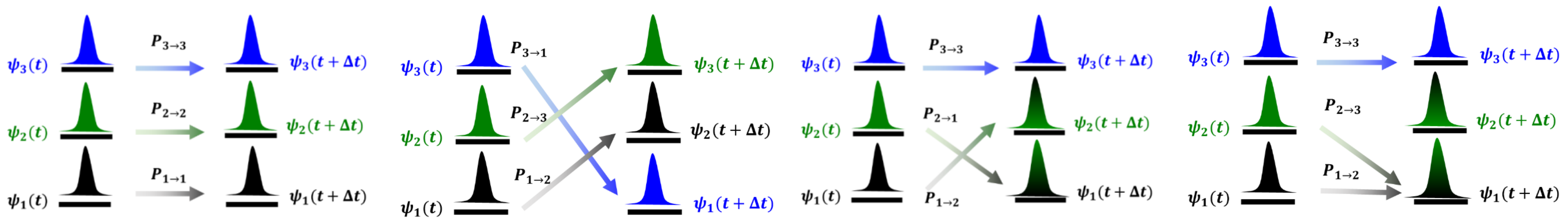
But:

- we also need to account for the all states' transition probabilities
- there are n! permutations = probabilities to compute

So:

- stochastic "many-state" hops
- reject unfeasible states

an adiabatic counterpart to the FSSH?



$$P = P_{1\to1}P_{2\to2}P_{3\to3}$$

no swapping

$$P = P_{1\to2}P_{2\to3}P_{3\to1}$$

$$P = P_{1\to2}P_{2\to1}P_{3\to3}$$

swaps can include inactive states

Excluded

bijective reordering

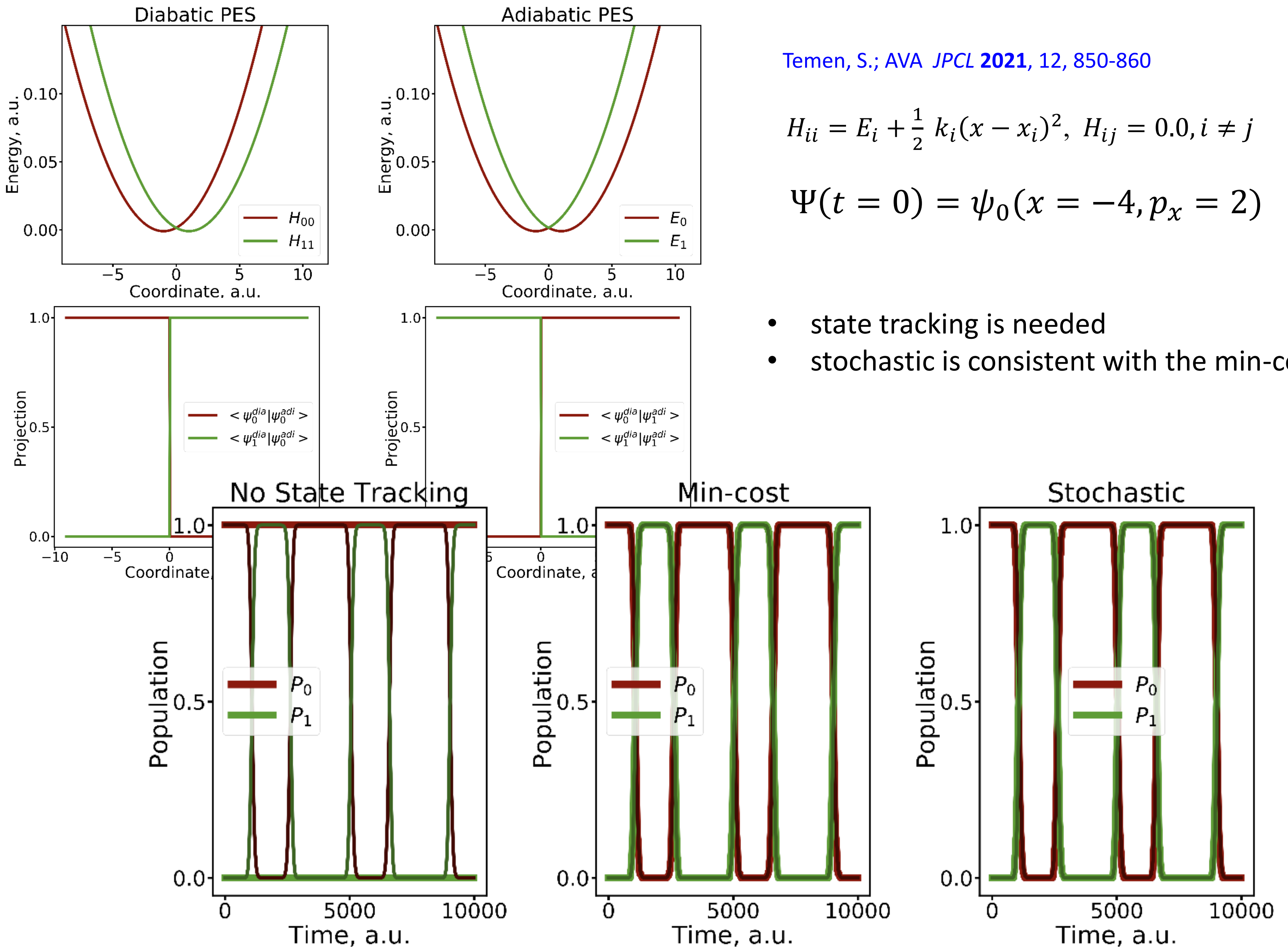
# Simple 2-state model

Temen, S.; AVA *JPCL* **2021**, 12, 850-860

$$H_{ii} = E_i + \frac{1}{2} k_i (x - x_i)^2, \;\; H_{ij} = 0.0, i \neq j$$

$$\Psi(t=0) = \psi_0(x = -4, p_x = 2)$$

- state tracking is needed
- stochastic is consistent with the min-cost



**int state_tracking_algo;**

State tracking algorithm:

- 0: no state tracking
- 1: method of Kosuke Sato (may fail by getting trapped into an infinite loop)
- 2: Munkres-Kuhn (Hungarian) algorithm [ default ]
- 3: experimental stochastic algorithm, the original version with elimination (known problems)
- 32: experimental stochastic algorithms with all permutations (too expensive)
- 33: the improved stochastic algorithm with good scaling and performance, on par with the mincost

**double MK_alpha;**

Munkres-Kuhn alpha (selects the range of orbitals included in reordering) [default: 0.0]

$$C_{ij} = -\left|S_{t,ij}(t, t+\Delta t)\right|^2 \exp(-\alpha^2 \Delta E_{ij}^2)$$

**int convergence;**

A switch for stochastic reordering algorithm to choose what happens when an acceptable permutation isn't generated in the set number of attempts:

- 0: returns the identity permutation (does not require convergence)
- 1: exits and prints an error (requires convergence)

**int max_number_attempts;**

The maximum number of hops that an be attempted before either choosing the identity or exiting in stochastic reordering algorithm 3.

**double min_probability_reordering;**

The probability threshold for stochastic state reordering algorithm. If a probability for a multi-state stransition is below this value, it will be disregarded and set to 0. The rest of the probabilities will be renormalized

# Hop proposal probability

University at Buffalo
*The State University of New York*

**int tsh_method;**

Options:

- [-1]: adiabatic dynamics, no hops [ default ]
- 0: FSSH
- 1: GFSH
- 2: MSSH
- 3: DISH

$$P^P_{i\to f} = \max\left(0, \frac{\Delta t}{\hbar P_{ii}} Im\left[P_{i,f} H^{vib}_{f,i} - H^{vib}_{i,f} P_{f,i}\right]\right)$$

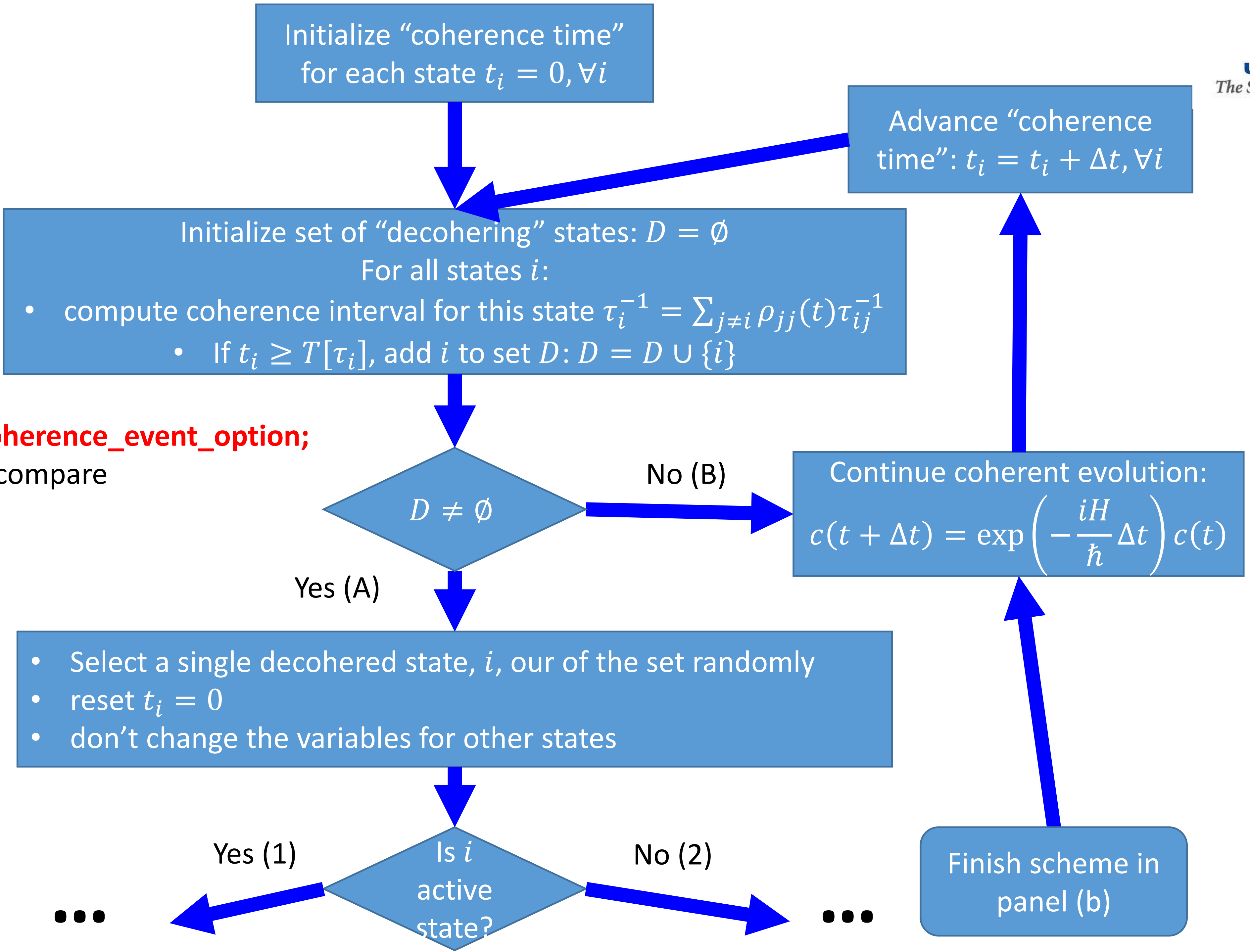
Tully, J. C. *J. Chem. Phys.* **1990**, *93*, 1061

$$P^P_{i\to f} = \max\left(0, \frac{\Delta P_{ff}}{P_{ff}} \frac{\Delta P_{jj}}{\sum_{k\in A} \Delta P_{kk}}\right). i \in A, j \in B$$

Wang, L.; Trivedi, D.; Prezhdo, O. V. *JCTC* **2014**, 10, 3598

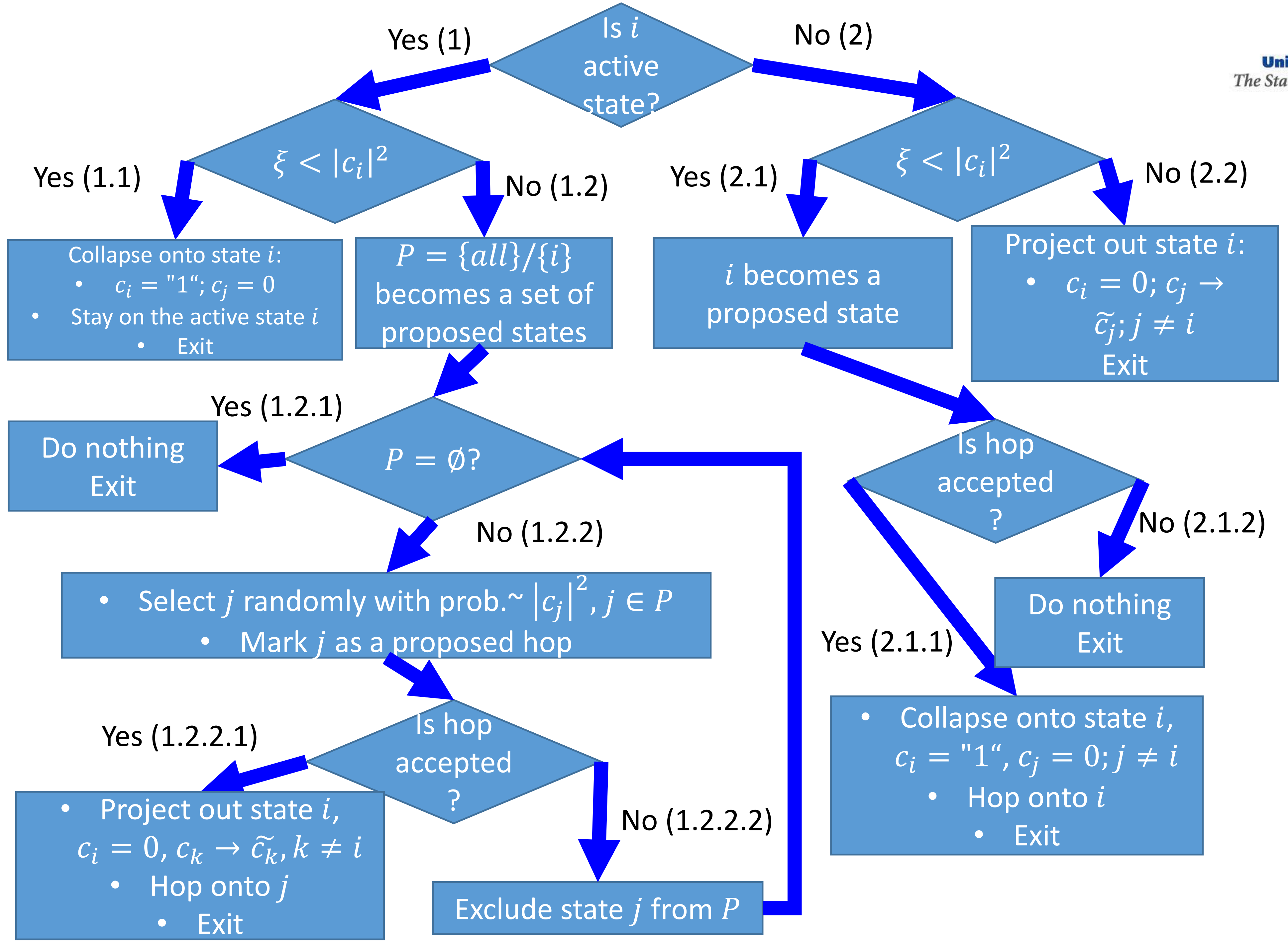
$$P^P_{i\to f}(t, t+\Delta t) = P_{ff}(t+\Delta t)$$

Akimov, A. V.; Trivedi, D.; Wang, L.; Prezhdo, O. V. *J. Phys. Soc. Jpn.* **2015**, 84, 094002


Initialize "coherence time" for each state $t_i = 0, \forall i$
Advance "coherence time": $t_i = t_i + \Delta t, \forall i$
Initialize set of "decohering" states: $D = \emptyset$
For all states $i$:
• compute coherence interval for this state $\tau_i^{-1} = \sum_{j \neq i} \rho_{jj}(t)\tau_{ij}^{-1}$
• If $t_i \geq T[\tau_i]$, add $i$ to set $D$: $D = D \cup \{i\}$
$D \neq \emptyset$
No (B)
Continue coherent evolution: $c(t+\Delta t) = \exp\left(-\frac{iH}{\hbar}\Delta t\right)c(t)$
Yes (A)
• Select a single decohered state, $i$, our of the set randomly
• reset $t_i = 0$
• don't change the variables for other states
Is $i$ active state?
Yes (1)
No (2)
…
…
Finish scheme in panel (b)

**int dish_decoherence_event_option;**

- 0 – direct compare
- 1 -Poisson


Is $i$ active state?
Yes (1)
No (2)
$\xi < |c_i|^2$
Yes (1.1)
No (1.2)
$\xi < |c_i|^2$
Yes (2.1)
No (2.2)
Collapse onto state $i$:
• $c_i$ = "1"; $c_j = 0$
• Stay on the active state $i$
• Exit
$P = \{all\}/\{i\}$ becomes a set of proposed states
$i$ becomes a proposed state
Project out state $i$:
• $c_i = 0$; $c_j \rightarrow \tilde{c_j}$; $j \neq i$
Exit
Yes (1.2.1)
Do nothing
Exit
$P = \emptyset$?
Is hop accepted ?
No (2.1.2)
No (1.2.2)
• Select $j$ randomly with prob.~ $|c_j|^2$, $j \in P$
• Mark $j$ as a proposed hop
Do nothing
Exit
Yes (2.1.1)
• Collapse onto state $i$, $c_i$ = "1", $c_j = 0$; $j \neq i$
• Hop onto $i$
• Exit
Yes (1.2.2.1)
Is hop accepted ?
• Project out state $i$, $c_i = 0$, $c_k \rightarrow \tilde{c_k}$, $k \neq i$
• Hop onto $j$
• Exit
No (1.2.2.2)
Exclude state $j$ from $P$

University at Buffalo
The State University of New York

# Hop acceptance probabilities

**int hop_acceptance_algo;**

Options:

- 0: accept all proposed hops [ default ]

$$P^A_{i\to f} = 1$$

- 10: based on adiabatic energy - accept only those hops that can obey the energy conservation with
  adiabatic potential energies
- 11: based on diabatic energy - same as 10, but we use diabatic potential energies

- 20: based on derivative coupling vectors - accept only those hops that can obey the energy conservation
  by rescaling nuclear velocities along the directions of derivative couplings for the quantum nuclear DOFs
- 21: based on difference of state-specific forces - same as 20, but the rescaling is done along the vector
  parallel to the difference of adiabatic forces on initial and target states

$$P^A_{i\to f} = \Theta\left(E_{kin} + E_f - E_f\right)$$

Tully, J. C. *J. Chem. Phys.* **1990**, *93*, 1061

- 31: accept hops with the probability taken from the quantum Boltzmann distribution
- 32: accept hops with the probability taken from the classical Maxwell-Boltzmann distribution
- 33: accept hops with the probability taken from the updated quantum Boltzmann distribution (experimental)

Prezhdo, O. V.; Duncan, W. R.; Prezhdo, V. V. *Prog. Surf. Sci.* **2009**, *84*, 30

$$P^A_{i\to f} = \min\left(1, exp\left(-\frac{\Delta E}{k_B T}\right)\right)$$

$$P^A_{i\to f} = 1 - \left[\mathrm{erf}\left(\left(\frac{\Delta E}{k_B T}\right)^{\frac{1}{2}}\right) - \sqrt{\frac{4}{\pi}}\left(\frac{\Delta E}{k_B T}\right)^{\frac{1}{2}} \exp\left(-\frac{\Delta E}{k_B T}\right)\right]$$

Smith, B.; Akimov, A. V. *J. Chem.Phys.* **2019**, 151, 124107

## Momentum rescaling

int momenta_rescaling_algo;

Options:

- 0: don't rescale [ default ]

- 100: based on adiabatic energy, don't reverse on frustrated hops
- 101: based on adiabatic energy, reverse on frustrated hops
- 110: based on diabatic energy, don't reverse on frustrated hops
- 111: based on diabatic energy, reverse on frustrated hops

- 200: along derivative coupling vectors, don't reverse on frustrated hops
- 201: along derivative coupling vectors, reverse on frustrated hops
- 210: along difference of state-specific forces, don't reverse on frustrated hops
- 211: along difference of state-specific forces, reverse on frustrated hops

# Decoherence

**double decoherence_algo;**
Options:

- [-1]: no decoherence [ default ]
- 0: SDM and alike
- 1: instantaneous decoherence options (ID-S, ID-A, ID-C)
- 2: AFSSH

## SDM

Granucci, G.; Persico, M. *J. Chem. Phys.* **2007**, *126*, 134114.

gradually change the amplitudes

$$C_i' = C_i \exp\left(-\frac{\Delta t}{\tau_{if}}\right), \forall i \neq f$$

$$C_f' = C_f \sqrt{\frac{1 - \sum_{i \neq f} |C_i'|^2}{|C_f|^2}}$$

## ID-A

Nelson, T.; Fernandez-Alberti, S.; Roitberg, A. E.; Tretiak, S. *J. Chem. Phys.* **2013**, *138*, 224111.

- on a successful hop (ID-S)
- on an attempted hop (ID-A)
- at every timestep (ID-C)

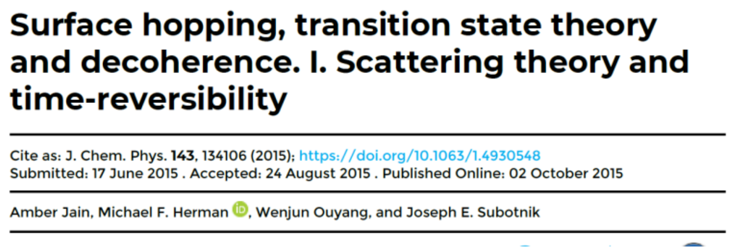
Wavefunction reduction

$$C_f = 1, C_i = 0, \forall i \neq f$$

**int collapse_option;**
How to collapse wavefunction amplitudes in the decoherence schemes:
- 0: by rescaling the magnitude of the amplitude vector elements, but preserving "phase" [ default ]
- 1: by resetting the amplitudes to 1.0+0.0j. This option changes phase

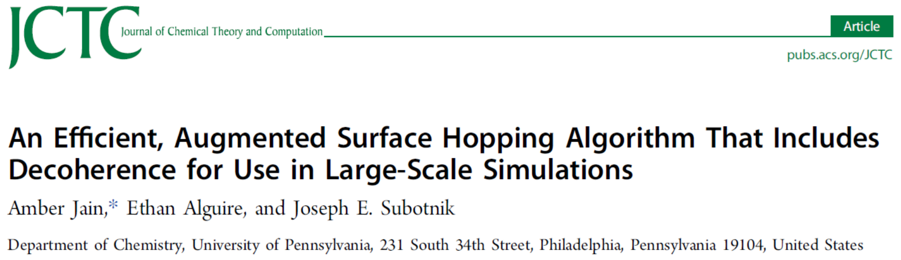
**int instantaneous_decoherence_variant;**
- 0: ID-S
- 1: ID-A [default]
- 2: ID-C - consistent ID - an experimental algorithm

# Decoherence: A-FSSH


**Surface hopping, transition state theory and decoherence. I. Scattering theory and time-reversibility**

Cite as: J. Chem. Phys. 143, 134106 (2015); https://doi.org/10.1063/1.4930548
Submitted: 17 June 2015 . Accepted: 24 August 2015 . Published Online: 02 October 2015

Amber Jain, Michael F. Herman, Wenjun Ouyang, and Joseph E. Subotnik


JCTC Journal of Chemical Theory and Computation — Article — pubs.acs.org/JCTC

**An Efficient, Augmented Surface Hopping Algorithm That Includes Decoherence for Use in Large-Scale Simulations**

Amber Jain,* Ethan Alguire, and Joseph E. Subotnik

Department of Chemistry, University of Pennsylvania, 231 South 34th Street, Philadelphia, Pennsylvania 19104, United States

Propagate extra set of variables

$$\delta\vec{\mathbf{R}} = \mathrm{Tr}_N[(\vec{R} - \vec{R}_{SH})\boldsymbol{\rho}]$$

$$\delta\vec{\mathbf{P}} = \mathrm{Tr}_N[(\vec{P} - \vec{P}_{SH})\boldsymbol{\rho}]$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\delta\vec{R}_{jk} = \left[-\frac{i}{\hbar}\mathbf{V} - \mathbf{T},\, \delta\vec{\mathbf{R}}\right]_{jk} + \frac{\delta\vec{P}_{jk}}{\vec{m}}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\delta\vec{P}_{jk} = \left[-\frac{i}{\hbar}\mathbf{V} - \mathbf{T},\, \delta\vec{\mathbf{P}}\right]_{jk} + \frac{1}{2}(\delta\vec{\mathbf{F}}\boldsymbol{\sigma} + \boldsymbol{\sigma}\delta\vec{\mathbf{F}})_{jk}$$

These variables define the rates for wavefunction collapse

$$\frac{1}{\tau_d^{n\lambda}} = -\frac{\frac{\mathrm{d}}{\mathrm{d}t}|\sigma_{n\lambda}|}{|\sigma_{n\lambda}|} \simeq \frac{\delta\vec{F}_{nn}\cdot(\delta\vec{R}_{nn} - \delta\vec{R}_{\lambda\lambda})}{2\hbar} - \frac{2|\vec{F}_{\lambda n}\cdot(\delta\vec{R}_{nn} - \delta\vec{R}_{\lambda\lambda})|}{\hbar}$$

5. Compute the probability to collapse the amplitudes for the state $n \neq i$ (where $i$ is the active surface) as

$$\gamma_n^{collapse} = dt\left(\frac{(F_{nn} - F_{ii})\delta x_{nn}}{2\hbar} - \frac{2|F_{in}\delta x_{nn}|}{\hbar}\right). \qquad \text{(A10)}$$

Also compute the probability to reset the moments as

$$\gamma_n^{reset} = -dt\left(\frac{(F_{nn} - F_{ii})\delta x_{nn}}{2\hbar}\right). \qquad \text{(A11)}$$

```
void compute_dynamics(MATRIX& q, MATRIX& p, MATRIX& invM, CMATRIX& C, vector<CMATRIX>& projectors, vector<int>& act_states,
                      nHamiltonian& ham, bp::object py_funct, bp::dict model_params, bp::dict dyn_params, Random& rnd);

void compute_dynamics(MATRIX& q, MATRIX& p, MATRIX& invM, CMATRIX& C, vector<CMATRIX>& projectors, vector<int>& act_states,
                      nHamiltonian& ham, bp::object py_funct, bp::dict& model_params, bp::dict& dyn_params, Random& rnd,
                      vector<Thermostat>& therm);

void compute_dynamics(MATRIX& q, MATRIX& p, MATRIX& invM, CMATRIX& C, vector<CMATRIX>& projectors, vector<int>& act_states,
                      nHamiltonian& ham, bp::object py_funct, bp::dict& model_params, bp::dict& dyn_params, Random& rnd,
                      vector<Thermostat>& therm, dyn_variables& dyn_var);
```

## Dephasing times

University at Buffalo
The State University of New York

**int decoherence_times_type;**
- 0 : read from input
- 1 : EDC

### SDM/EDC

$$\tau_{ij}^{EDC} = \frac{\hbar}{|E_i - E_j|}\left(1 + \frac{C}{E_{kin}}\right)$$

Granucci, G.; Persico, M. *J. Chem. Phys.* **2007**, *126*, 134114.

**double decoherence_C_param;**

**double decoherence_eps_param;**

### mSDM

$$\tau_{ij}^{-1} = \sqrt{\frac{5\langle \delta E_{ij}^2 \rangle}{12\hbar^2}}$$

Akimov, A. V.; Prezhdo, O. V. *J. Phys. Chem. Lett.* **2013**, *4*, 3857
Smith, B.; Akimov, A. V. *J. Chem.Phys.* **2019**, 151, 124107

### DISH

Decoherence interval

$$\tau_i^{-1} = \sum_{j \neq i} P_{jj} \tau_{ij}^{-1}$$

Jaeger, H. M.; Fischer, S.; Prezhdo, O. V. *J. Chem. Phys.* **2012**, *137*, 22A545

### Phase-informed Decoherence times

Sifain, A. E.; Wang, L.; Tretiak, S.; Prezhdo, O. V.

**int dephasing_informed;**
- 0: don't apply [ default ]
- 1: use it

$$\tau_{ij}^{-1,PI} = \tau_{ij}^{-1} \frac{|E_i - E_j|}{\langle |E_i - E_j| \rangle}$$

# Many ways of doing the dynamics

## Corrections

BBCE

thermal Ehrenfest

$$H_{ij}^{Bastida,adi} = |c_j| f_{ij} V_{ij} + |c_i| f_{ji} V_{ji}$$

$$f_{ij} = \left( \frac{2}{1 + \exp\left( - \frac{E_{ij}}{k_B T} \right)} \right)^{1/2}$$

Bastida, A. et al. *Chem. Phys. Lett.* **2006**, *417*, 53

- State tracking (e.g. mincost)
- Phase corrections (Akimov, A. V *J. Phys. Chem. Lett.* **2018**, *9*, 6096)

## Forces & Nuclear Dynamics

- Adiabatic (NBRA, various states)
- TSH
- Ehrenfest
- Quantized nuclei (Bohmian trajectories?)
- Bath: Langevin, Nose-Hoover, etc.

- Frustrated hops:
  - Reverse momenta
  - Keep momenta

- Accepted hops:
  - Don’t rescale momenta
  - Rescale along NACs
  - Rescale along force difference

**libra_py**
(All the Python modules)

**workflows**
(self-explanatory)

**nbra**
(currently, not only NBRA!)

**step2:** time-overlaps & NAC calculations

**step2_many_body:** SP time-overlaps & NAC calculations, and CI info

**step3**: phase corrections, reindexing, SD-basis properties, etc.

**step3_many_body**: functions for computing NACs in the MB basis.

**step4**: dynamics!

- **namd_workflow** – parallelization, multiple methods, and iconds
- **run_tsh** - initializes nuclear coordinates and runs NAMD
- **run** - the older method

**dynamics**
(self-explanatory)

**tsh**
(self-explanatory)

**compute**

- **generic_recipe** – initializes adiabatic/diabatic variables
- **run_dynamics** – sets up the default parameters, initializes Hamiltonains, saves results, etc.

**libra_core**
(All the C++ stuff, exposed to Python)

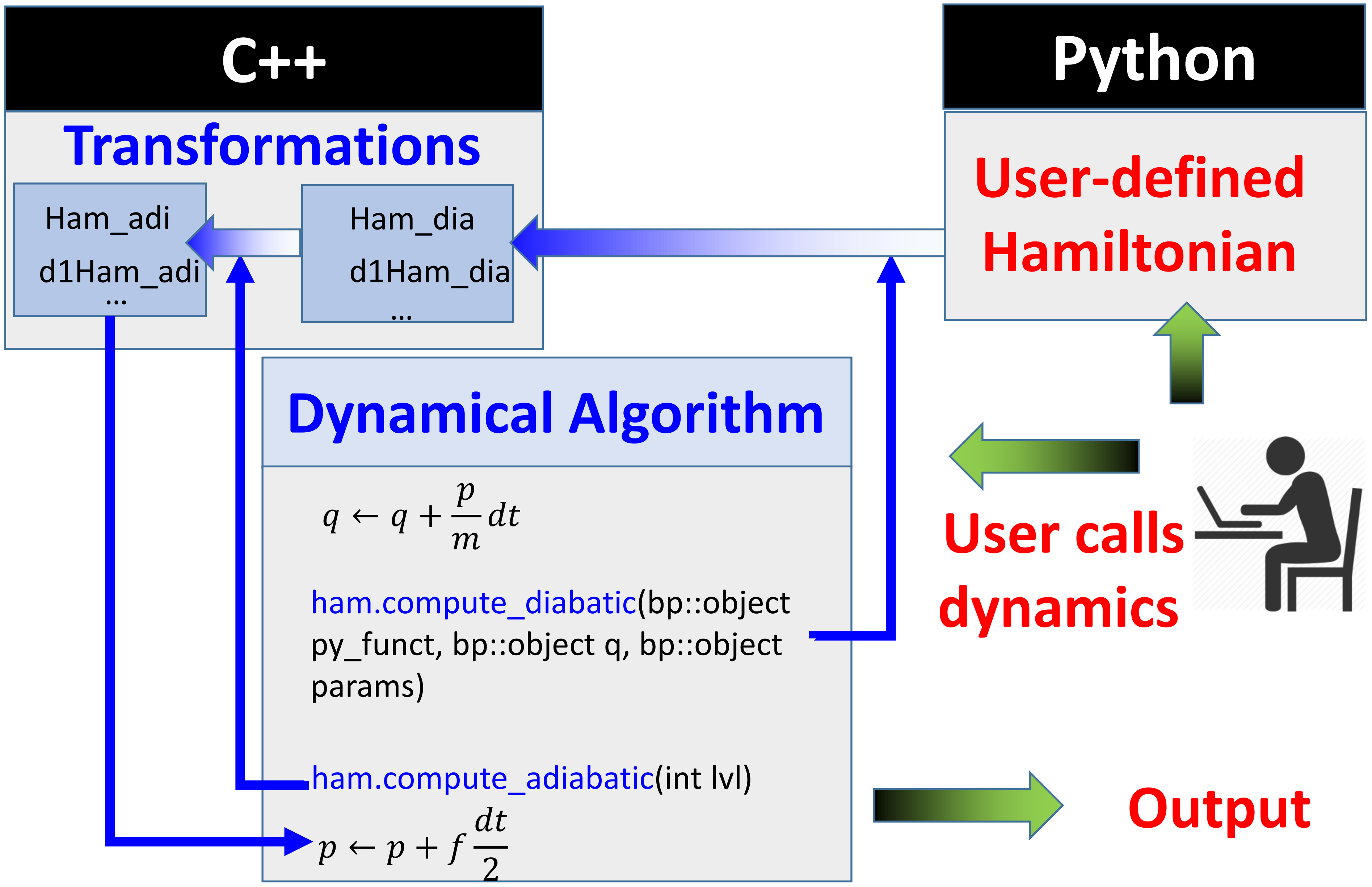
**compute_dynamics**

**This is the key function!**

| | |
|---|---|
| **workflows.nbra.step4.namd_workflow** | Parallelization, multiple methods/initial conditions |
| ↓ | |
| **workflows.nbra.step4.run_tsh** | Initialization of nuclear variables |
| ↓ | |
| **dynamics.tsh.compute.generic_recipe** | Initialization of electronic variables, transformation to the desired representation, nHamiltonian object construction and initialization |
| ↓ | |
| **dynamics.tsh.compute.run_dynamics** | Initialization of default parameters, writing files, computing some observables, thermostat variables, iteration over nuclear timesteps |
| ↓ | |
| **compute_dynamics (libra_core)** | TSH/Ehrenfest and decoherence algorithms, trajectory coupling, Hamiltonian properties updates (calling external Python functions) |

## *nHamiltonian class*

# How it works with dynamics



$$q \leftarrow q + \frac{p}{m} dt$$

```
ham.compute_diabatic(bp::object py_funct, bp::object q, bp::object params)
```

```
ham.compute_adiabatic(int lvl)
```

$$p \leftarrow p + f \frac{dt}{2}$$

$$i\hbar\frac{dC_{adi}}{dt} = H_{adi}C_{adi} - i\hbar\sum_n D_{adi,n}\frac{p_n}{m_n}C_{adi} = H_{vib}C_{adi}$$

$$H_{vib,adi} = H_{adi} - i\hbar d$$

- compute_hvib_adi, compute_hvib_dia

NAC (scalars)

$$d = \sum_n D_{adi,n}\frac{p_n}{m_n}$$

- compute_nac_adi, compute_nac_dia

# Computing $H_{dia}^{vib}$

Blue = Required Input
Green = Output
Green with D = Can be set up directly via Python function call

| Function | $Q$ | $P$ | $H_{dia}$ | $D_{dia}$ | $d_{dia}$ | $H_{dia}^{vib}$ |
|---|---|---|---|---|---|---|
| nHamiltonian::**compute_diabatic**(bp ::object py_funct …) | 🟦 | | 🟩 D | 🟩 D | 🟩 D | 🟩 D |
| nHamiltonian::**compute_nac_dia**(…) | | 🟦 | | 🟦 | 🟩 | |
| nHamiltonian::**compute_hvib_dia**(… ) | | | 🟦 | | 🟦 | 🟩 |

# Computing $H_{adi}^{vib}$

University at Buffalo
The State University of New York

| Function | $Q$ | $P$ | $S$ | $H_{dia}$ | $\nabla H_{dia}$ | $D_{dia}$ | $U$ | $H_{adi}$ | $\nabla H_{adi}$ | $D_{adi}$ | $d_{adi}$ | $H_{adi}^{vib}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| nHamiltonian::**compute_diabatic**(bp::object py_funct …) | 🟦 |  | 🟩 D | 🟩 D | 🟩 D | 🟩 D |  |  |  |  |  |  |
| nHamiltonian::**compute_adiabatic**(…) |  |  | 🟦 | 🟦 | 🟦 | 🟦 | 🟩 | 🟩 | 🟩 | 🟩 |  |  |
| nHamiltonian::**compute_adiabatic**(bp::object py_funct …) | 🟦 |  |  |  |  |  | 🟩 D | 🟩 D | 🟩 D | 🟩 D | 🟩 D | 🟩 D |
| nHamiltonian::**compute_nac_adi**(…) |  | 🟦 |  |  |  |  |  |  |  | 🟦 | 🟩 |  |
| nHamiltonian::**compute_hvib_adi**(…) |  |  |  |  |  |  |  | 🟦 |  |  | 🟦 | 🟩 |

# nHamiltonian is hierarchical

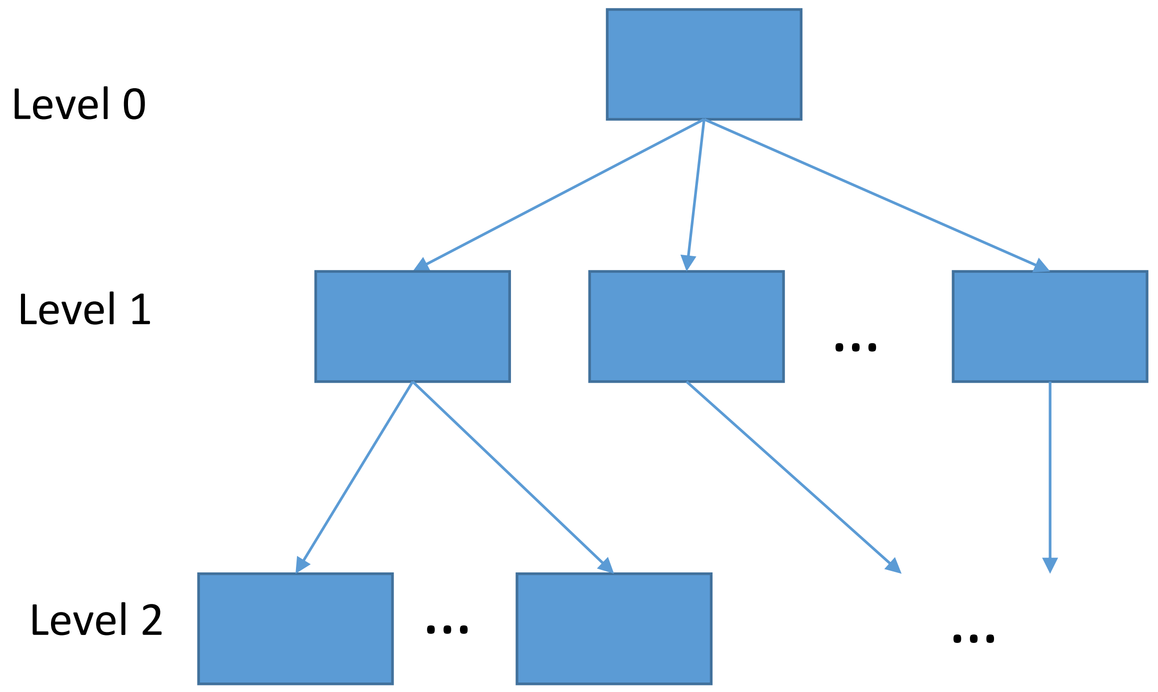
**nHamiltonian**

- level
- id
- nHamiltonian* parent
- vector<nHamiltonian*> children

- nnucl, nadi, ndia

- CMATRIX* ham_dia, nac_dia, hvib_dia
- CMATRIX* ham_adi, nac_adi, hvib_adi
- CMATRIX* ovlp_dia, time_overlap_dia
- CMATRIX* ovlp_adi, time_overlap_adi
- CMATRIX* basis_transform
- vector<CMATRIX*> dc1_adi, dc1_dia
- vector<CMATRIX*> d1ham_adi, d1ham_dia

- ampl_dia2adi
- ampl_adi2dia



University at Buffalo
The State University of New York

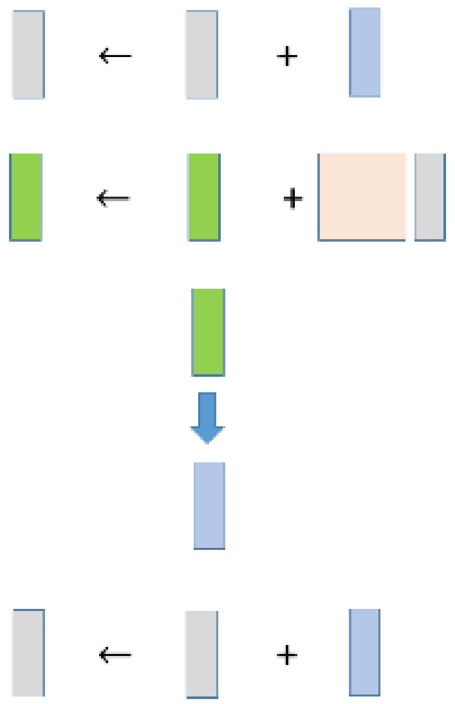
## Individual trajectory

$$p \leftarrow p + f\frac{dt}{2}$$

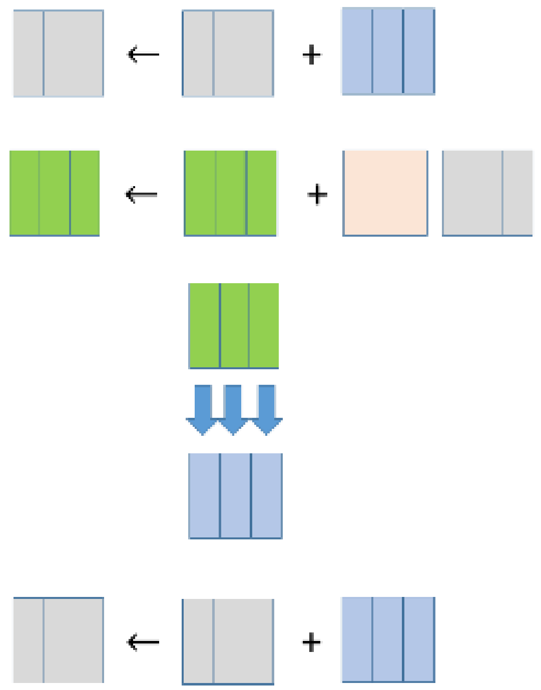
$$q \leftarrow q + M^{-1}pdt$$

Forces update

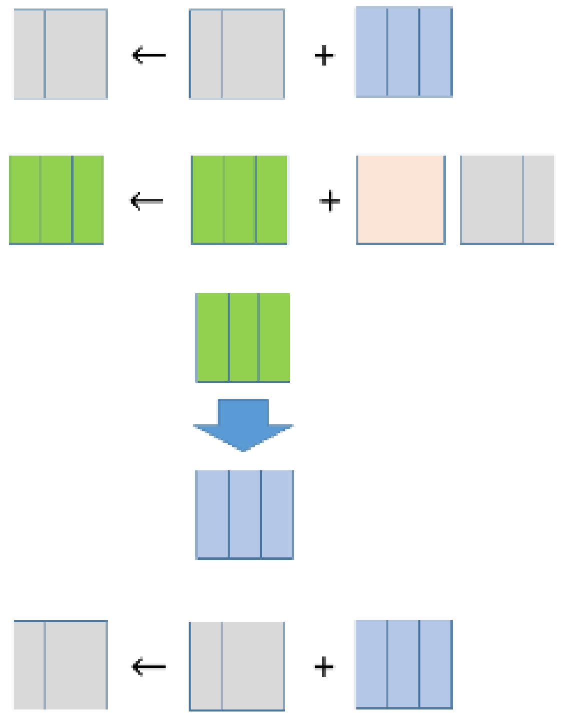
$$p \leftarrow p + f\frac{dt}{2}$$



## Swarm of uncoupled trajectories



## Swarm of coupled trajectories



## Keep Dynamical Workflow Fixed

University at Buffalo
*The State University of New York*

**User defines how to run the dynamical simulation**

```
for i in range(500):
      propagate_el(Cdia, Cadi, Hvib, Sdia, 0.5*dt, rep)
      p = p + 0.5*f*dt
      q = q + dt*p/m
      compute_model(model, Hdia, Sdia, d1ham_dia, dc1_dia, q, params)
      ham.compute_adiabatic(1);
      f = compute_frc(ham, Cdia, Cadi, rep)
      p = p + 0.5*f*dt
      Hvib = compute_Hvib(Hdia, Hadi, dc1_dia, dc1_adi, p, m, rep)
      propagate_el(Cdia, Cadi, Hvib, Sdia, 0.5*dt, rep)
Etot = compute_etot(ham, p, Cdia, Cadi, m, rep)
```

**User defines what function to use to compute entries in the Hamiltonian object (diabatic/adiabatic Ham, overlap matrix, derivatives, etc.) - NEXT**

# Example: Model Calculations

University at Buffalo
The State University of New York

```
def model2(q, params):

    obj = tmp()
    obj.ham_dia = CMATRIX(2,2);   obj.ovlp_dia = CMATRIX(2,2);
    obj.d1ham_dia = CMATRIXList();  obj.d1ham_dia.append( CMATRIX(2,2))
    obj.dc1_dia = CMATRIXList();  obj.dc1_dia.append( CMATRIX(2,2))

    x = q.get(0)
    x0,k,D,V = params["x0"], params["k"], params["D"], params["V"]

    obj.ovlp_dia.set(0,0, 1.0+0.0j);  obj.ovlp_dia.set(0,1, 0.0+0.0j);
    obj.ovlp_dia.set(1,0, 0.0+0.0j);  obj.ovlp_dia.set(1,1, 1.0+0.0j);

    obj.ham_dia.set(0,0, k*x*x*(1.0+0.0j) );   obj.ham_dia.set(0,1, V*(1.0+0.0j));
    obj.ham_dia.set(1,0, V*(1.0+0.0j));           obj.ham_dia.set(1,1, (k*(x-x0)**2 + D)*(1.0

    for i in [0]:
        obj.d1ham_dia[i].set(0,0, 2.0*k*x*(1.0+0.0j) );   obj.d1ham_dia[i].set(0,1, 0.0+0.0j);
        obj.d1ham_dia[i].set(1,0, 0.0+0.0j);                     obj.d1ham_dia[i].set(1,1,2.0*k*(x-x0)*(1.0+0.0j));

        obj.dc1_dia[i].set(0,0, 0.0+0.0j);   obj.dc1_dia[i].set(0,1,-0.1+0.0j);
        obj.dc1_dia[i].set(1,0, 0.1+0.0j);   obj.dc1_dia[i].set(1,1, 0.0+0.0j);

    return obj
```

**Initialize Python objects**

**Set matrix elements according to your model**

# Example: Atomistic Calculations

```
def model_atomistic(q, params, indx):

   natoms = params["natoms"];  ndof = q.num_of_rows;  ndia = params[ "ndia" ]
   params[ "output_filename" ] = "detailed.out“

   obj = tmp()
   obj.ham_dia = CMATRIX(1,1);
   obj.ovlp_dia = CMATRIX(1,1);           obj.ovlp_dia.set(0,0, 1.0+0.0j)
   obj.d1ham_dia = CMATRIXList();
   for i in xrange(ndof):
      obj.d1ham_dia.append( CMATRIX(1,1) )

   os.system("mkdir wd/job_"+str(indx))
   os.system("cp dftb_in.hsd wd/job_"+str(indx)) #+"/dftb_in.hsd")
   os.chdir("wd/job_"+str(indx))

   create_input.update_coordinates(q, params)
   os.system("srun %s < dftb_in.hsd > out" % (exe_name) )  # DFTB calculations are run here!
   dftb_forces = parse_output.get_forces(params)
   os.chdir("../../")

   for i in xrange(ndof):
      obj.d1ham_dia[i].set(0,0, dftb_forces[i]*(-1.0+0.0j) )
      obj.dc1_dia[i].set(0, 0, 0.0+0.0j)

return obj
```

**Initialize Python objects**

**Prepare and Run external program**

**Set matrix elements according to your model**