University at Buffalo
The State University of New York

Nonadiabatic Dynamics with the Libra code

Department of Chemistry, University at Buffalo, The State University of New York, Buffalo, NY 14260-3000

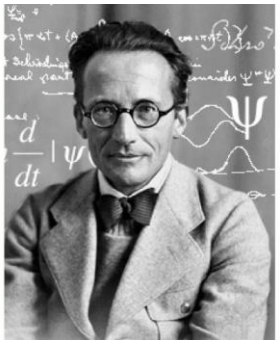
Alexey Akimov

"Excited States and Nonadiabatic Dynamics CyberTraining Workshop"
June 15, 2021

What is Nonadiabatic Dynamics?

University at Buffalo
The State University of New York

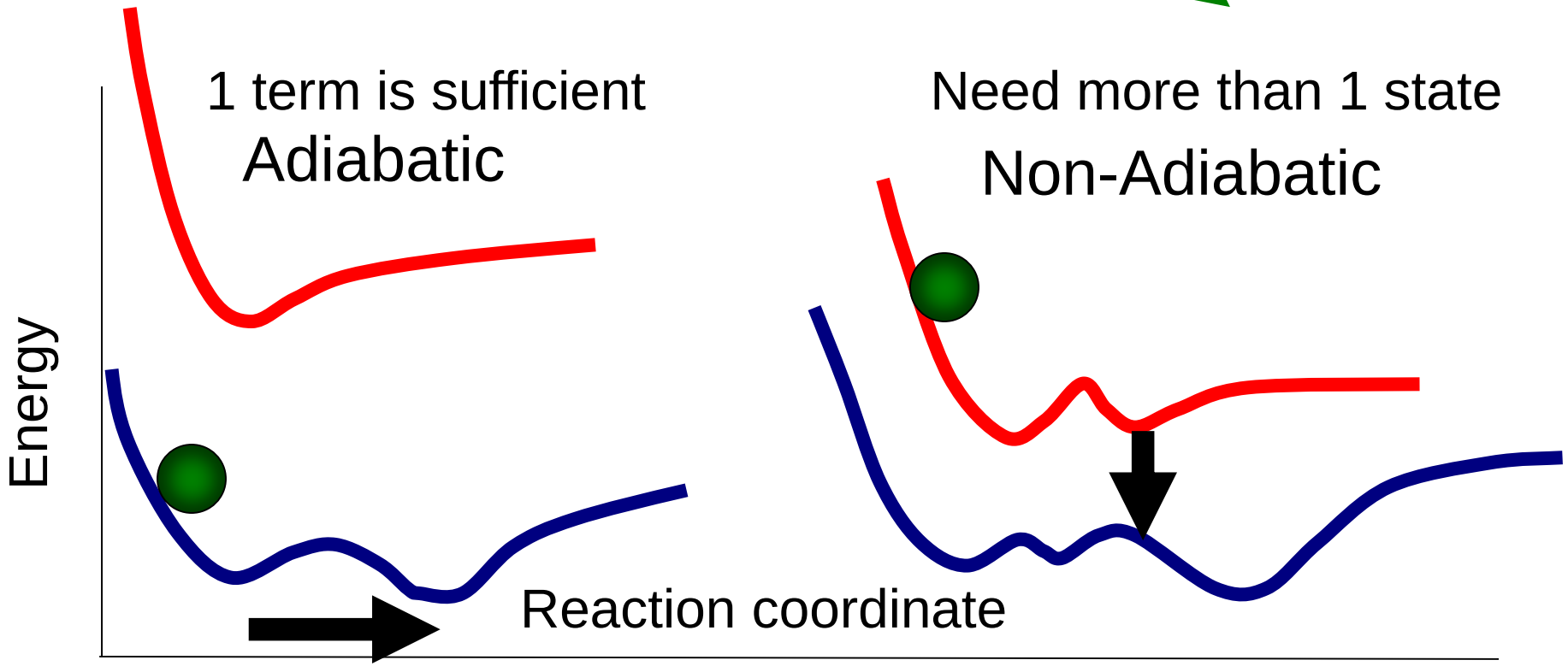
$$i\hbar \frac{\partial \Psi(\mathrm{r},\mathrm{R},\mathrm{t})}{\partial t} = H(\mathrm{r},\mathrm{R},\mathrm{t})\Psi(\mathrm{r},\mathrm{R},\mathrm{t})$$



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

1 term is sufficient
Adiabatic

Need more than 1 state
Non-Adiabatic



TSH in the nutshell

University at Buffalo
The State University of New York

Initialization

Nuclear dynamics

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

Stationary adiabatic states

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

Non-adiabatic Couplings

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

Electronic Dynamics

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

NBRA

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

Many ways of doing the dynamics

University at Buffalo
The State University of New York

Hop proposal probabilities

FSSH

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

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

GFSH

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

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

MSSH

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

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

Hop acceptance probabilities

Boltzmann

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

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

Maxwell-Boltzmann

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

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

Trivial

$$P_{i\to f}^{A} = 1$$

Energy conservation

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

Can rescale along NACs?

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

Can rescale along other vectors?

(e.g. excited state forces difference)

Many ways of doing the dynamics

University at Buffalo
The State University of New York

Decoherence schemes

SDM

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

gradually change the amplitudes

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

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

DISH

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

collapse amplitudes at an event:

at a decoherence event: $t_i \geq t_{decoh}$

Wavefunction reduction

$$C_i' = C_i \sqrt{\frac{1}{1 - \left|C_f\right|^2}}, \forall i \neq f$$

$$C_f' = 0$$

Hops with quantum probabilities

$$P_{i \to f}^{P}(t, t + \Delta t) = \left|C_f(t)\right|^2$$

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

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

Decoherence times/rates

SDM/EDC

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

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

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.

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

Many ways of doing the dynamics

University at Buffalo
The State University of New York

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 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 Akimov, A. V.; Prezhdo, O. V. *J. Phys. Chem. Lett.* **2013**, *4*, 3857 Sifain, A. E.; Wang, L.; Tretiak, S.; Prezhdo, O. V. 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 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

Phase correction

University at Buffalo
The State University of New York

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:

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

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

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

Indeed:

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

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

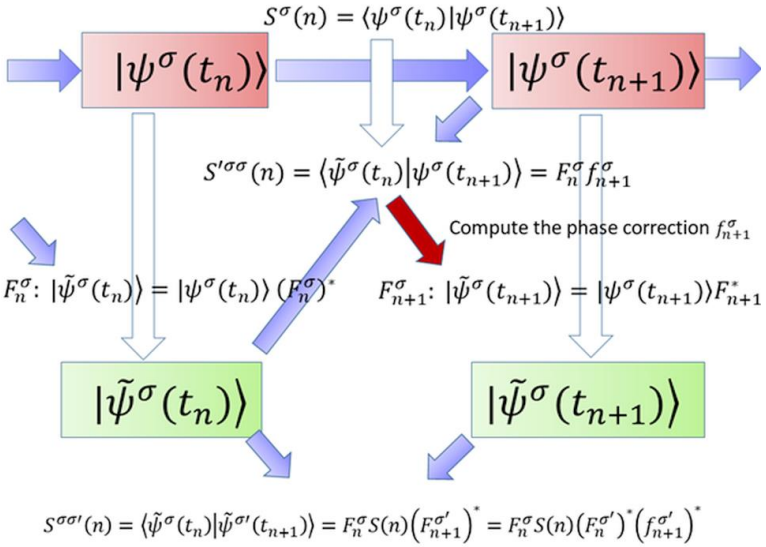
Then:

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

Phase-correct other properties:

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

Implementation in Libra:



Phase correction

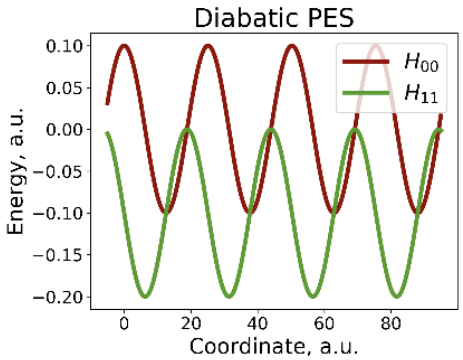
Akimov, A. V. *J. Phys. Chem. Lett.* **2018** 9, 6096-6102

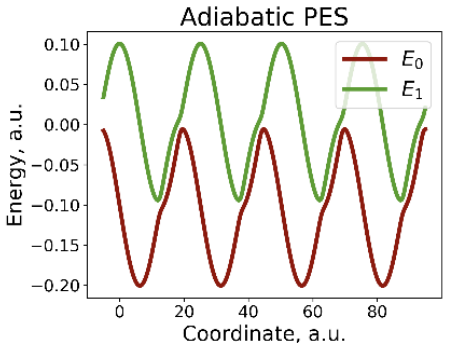
University at Buffalo
The State University of New York

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

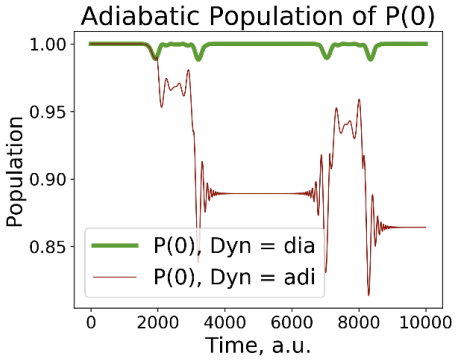
$$H_{ij} = V_{ij} = const.$$

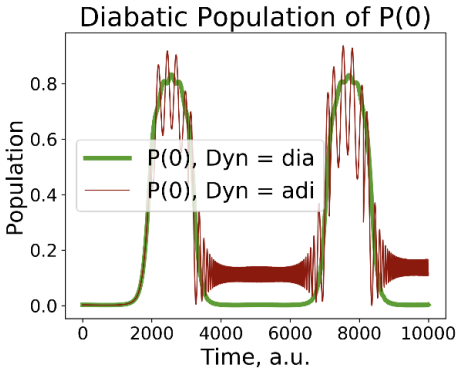
$$\Psi(t=0) = \psi_0$$

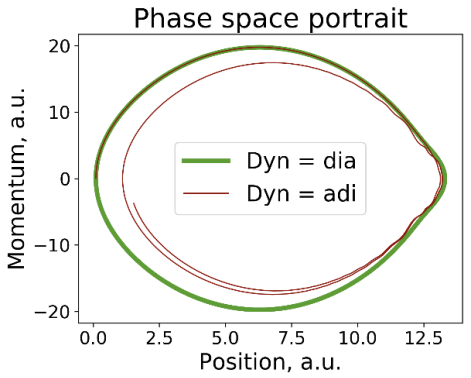




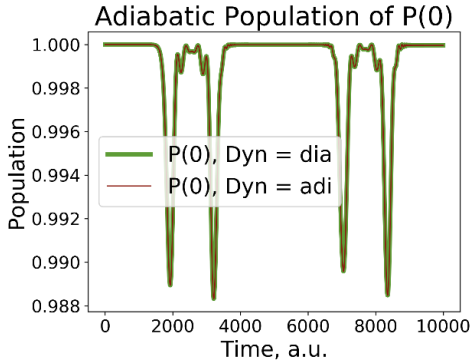
No

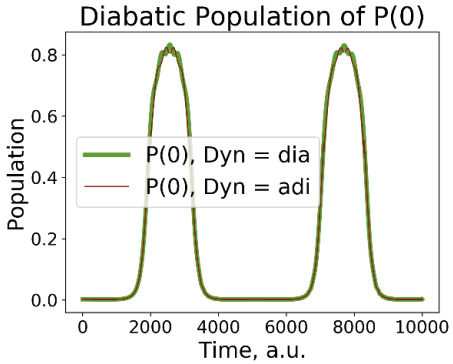


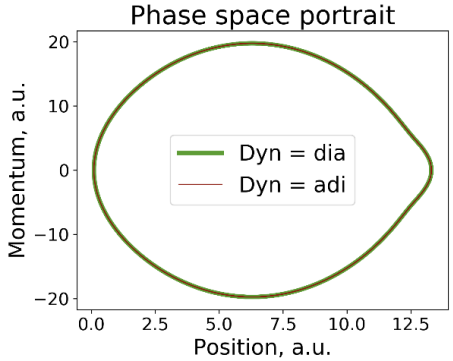




Yes







Comparing methods

University at Buffalo
The State University of New York

- use various representations
- define properties via external way

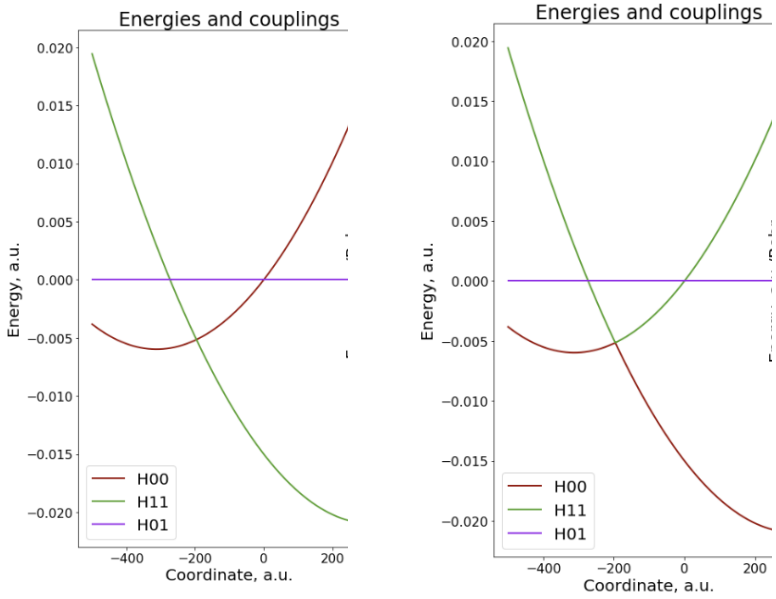
```
def compute_model(q, params, full_id):

    model = params["model"]
    res = None

    if model==1:
        res = my_test_dia_abstract(q, params, full_id)
    elif model==2:
        res = my_test_adi_abstract(q, params, full_id)
    elif model==3:
        res = my_test_adi_files(q, params, full_id)

    return res
```

Marcus-type (spin-Boson Hamiltonian)



Options

4.1. non-NBRA

- 4.1.1. adiabatic dynamics on the ground adiabatic state
- 4.1.2. adiabatic dynamics on the excited adiabatic state
- 4.1.3. non-adiabatic dynamics on the both states via FSSH
- 4.1.4. non-adiabatic dynamics on the both states via FSSH + SDM (user-defined decoherenc
- 4.1.5. non-adiabatic dynamics on the both states via FSSH + SDM (EDC decoherence rates,
- 4.1.6. non-adiabatic dynamics on the both states via FSSH + ID
  - 4.1.6.1. non-adiabatic dynamics on the both states via FSSH + ID-S
  - 4.1.6.2. non-adiabatic dynamics on the both states via FSSH + ID-A
  - 4.1.6.3. non-adiabatic dynamics on the both states via FSSH + ID-C

4.2. NBRA-FSSH

- 4.2.1. FSSH along the adiabatic dynamics on the ground state
- 4.2.2. FSSH along the adiabatic dynamics on the excited state
- 4.2.3. FSSH along the non-adiabatic dynamics trajectories obtained with FSSH
- 4.2.4. FSSH along the non-adiabatic dynamics trajectories obtained with FSSH + mSDM

4.3. NBRA-mSDM

- 4.3.1. mSDM along the adiabatic dynamics on the ground state
- 4.3.2. mSDM along the adiabatic dynamics on the excited state
- 4.3.3. mSDM along the non-adiabatic dynamics trajectories obtained with FSSH
- 4.3.4. mSDM along the non-adiabatic dynamics trajectories obtained with FSSH + mSDM

How to define your model

University at Buffalo
The State University of New York

```
class tmp:
    pass

def my_test_dia_abstract(q, params, full_id):
    """

    2-state spin
         | 0.5*m
    H =  |
         |

    M = sqrt(E_r

    Args:
        q ( MATR
        params (

            * **
            * **
            * **
            * **
            * **

    Returns:
        PyObject

            * ob
            * ob
            * ob

            * ob
zero ]
    """
                    critical_params = []
                    default_params = { "omega":3.5e-4, "E_r":2.39e-2, "mass":1.0, "V":1.49e-5, "epsilon":1.5e-2  }
                    comn.check_input(params, default_params, critical_params)

                    omega = params["omega"]
                    E_r = params["E_r"]
                    mass = params["mass"]
                    V = params["V"]
                    epsilon = params["epsilon"]

                    n = 2

                    Hdia = CMATRIX(n,n)
                    Sdia = CMATRIX(n,n)
                    d1ham_dia = CMATRIXList();  d1ham_dia.append(
                    dc1_dia = CMATRIXList();  dc1_dia.append( CMAT

                    Id = Cpp2Py(full_id)
                    indx = Id[-1]

                    x = q.col(indx).get(0)

                    Sdia.identity()

                    M = omega * math.sqrt(0.5*E_r * mass)

                    Hdia.set(0,0,  (0.5*mass*omega*omega*x*x + M *
                    Hdia.set(1,1,  (0.5*mass*omega*omega*x*x - M *
                    Hdia.set(0,1,  V  * (1.0+0.0j) )
                    Hdia.set(1,0,  V  * (1.0+0.0j) )

                    d1ham_dia[0].set(0, 0, (omega*omega*x + M)*(1.
                    d1ham_dia[0].set(1, 1, (omega*omega*x - M)*(1.
                    d1ham_dia[0].set(0, 1, (0.0+0.0j) )
                    d1ham_dia[0].set(1, 0, (0.0+0.0j) )


                    obj = tmp()
                    obj.ham_dia = Hdia
                    obj.ovlp_dia = Sdia
                    obj.d1ham_dia = d1ham_dia
                    obj.dc1_dia = dc1_dia

                    return obj
```

```
import numpy as np
import psi4
import os
import sys


def run(grad_method, charge, spin_multiplicity, coords, options):

    psi4.core.set_output_file('tmp.dat',False)

    mol = psi4.geometry(F"""
    {charge} {spin_multiplicity}
    Li 0.0 0.0 0.0
    H  0.0 0.0 1.0

    units bohr
    """)

    mol.set_geometry(psi4.core.Matrix.from_array(coords))
    #mol.set_options( {'reference': 'rhf', 'basis':'6-31Gss', 'guess':'core'} )
    psi4.set_options(options)

    # Compute the Counterpoise-Corrected interaction energy
    # bsse_type='cp'
    E, wfc = psi4.energy(grad_method, molecule=mol, return_wfn=True)

    # Compute force at the converged density
    force = -np.asarray(psi4.gradient(grad_method,ref_wfn=wfc))

    print(E, force)


    return E,force
```

Simplicity

University at Buffalo
The State University of New York

```
# Random numbers generator object
rnd = Random()

ntraj = 25

# Initialize nucle
init_nucl = {"init
q, p, iM = dynamic
```

```
DR = MATRIX(2,2)
AG = MATRIX(2,2)

dyn_params = { "rep_tdse":1, "rep_ham":0, "rep_sh":1, "rep_lz":0, "tsh_method":-1,
               "force_method":1, "nac_update_method":1, "rep_force":1,
               "hop_acceptance_algo":20, "momenta_rescaling_algo":201,
               "use_boltz_factor":0, "Temperature":300.0, "do_reverse":1, "vel_rescale_opt":-1,
               "do_phase_correction":1, "tol":1e-3,
               "state_tracking_algo":2, "MK_alpha":0.0, "MK_verbosity":0,
               "entanglement_opt":0, "ETHD3_alpha":0.0, "ETHD3_beta":0.0,
               "decoherence_algo":-1, "decoherence_rates":DR,
               "decoherence_times_type":0, "decoherence_C_param":1.0,
               "decoherence_eps_param":0.1, "dephasing_informed":0,
               "ave_gaps":AG, "instantaneous_decoherence_variant":1, "collapse_option":0,
               "ensemble":0, "thermostat_params":{},
```

Case 1.1.1. Ehrenfest, adiabatic tdse, diabatic Ham

```
# Dynamics (Simulation) parameters
dyn_params.update({ "force_method":2, "rep_tdse":1, "rep_ham":0, "prefix":"T1-1-1-1", "tsh_method":-1, "re
p_sh":1 })
dyn_params.update({"state_tracking_algo":2, "do_phase_correction":1, "tol":1e-3, "nsteps":500 })

print(dyn_params)
res = dynamics.run_dynamics(q, p, iM, Cdia, Cadi, projectors, states, dyn_params, compute_model, {"model":
1}, rnd)

plot_dyn(res)
```

Variety of options

University at Buffalo
The State University of New York

Initialization, sampling

```
# Random numbers generator object
rnd = Random()

ntraj = 25

# Initialize nuclear variables
init_nucl = {"init_type":3, "force_constant":[0.001], "ntraj":ntraj}
q, p, iM = dynamics.init_nuclear_dyn_var([-300.0], [20.0], [10.0], init_nucl, rnd)
```

Model, methods, common options

```
DR = MATRIX(2,2)
AG = MATRIX(2,2)

dyn_params = { "rep_tdse":1, "rep_ham":0, "rep_sh":1, "rep_lz":0, "tsh_method":-1,
               "force_method":1, "nac_update_method":1, "rep_force":1,
               "hop_acceptance_algo":20, "momenta_rescaling_algo":201,
               "use_boltz_factor":0, "Temperature":300.0, "do_reverse":1, "vel_rescale_opt":-1,
               "do_phase_correction":1, "tol":1e-3,
               "state_tracking_algo":2, "MK_alpha":0.0, "MK_verbosity":0,
               "entanglement_opt":0, "ETHD3_alpha":0.0, "ETHD3_beta":0.0,
               "decoherence_algo":-1, "decoherence_rates":DR,
               "decoherence_times_type":0, "decoherence_C_param":1.0,
               "decoherence_eps_param":0.1, "dephasing_informed":0,
               "ave_gaps":AG, "instantaneous_decoherence_variant":1, "collapse_option":0,
               "ensemble":0, "thermostat_params":{},

               "dt":0.1*units.fs2au, "nsteps":500,
               "output_level":2, "file_output_level":0
             }
```

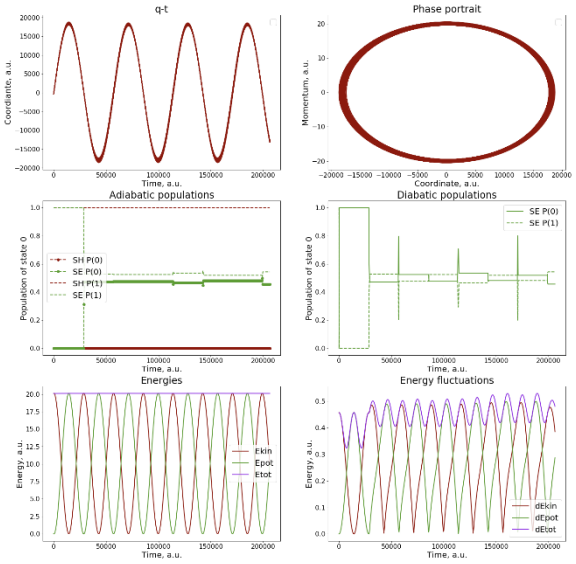
Ehrenfest

Case 1.1.1. Ehrenfest, adiabatic tdse, diabatic Ham

```
# Dynamics (Simulation) parameters
dyn_params.update({ "force_method":2, "rep_tdse":1, "rep_ham":0, "prefix":"T1-1-1-1", "tsh_method":-1, "rep_sh":1 })
dyn_params.update({"state_tracking_algo":2, "do_phase_correction":1, "tol":1e-3, "nsteps":500 })

print(dyn_params)
res = dynamics.run_dynamics(q, p, iM, Cdia, Cadi, projectors, states, dyn_params, compute_model, {"model":1}, rnd)

plot_dyn(res)
```



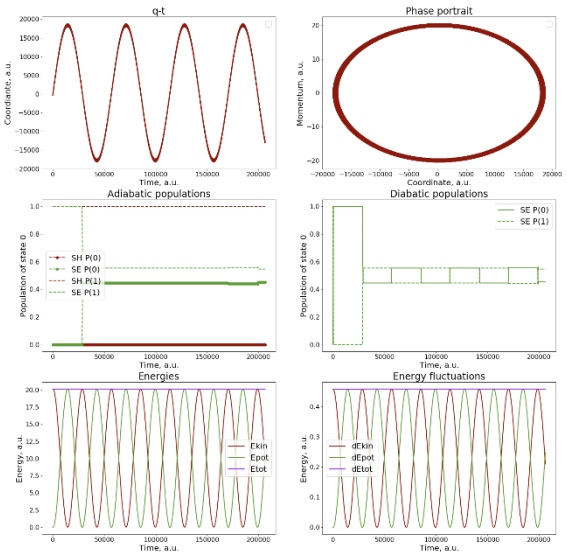
Adiabatic MD

Case 2.1.1.1. Adiabatic dynamics (no hops), TDSE is in adiabatic rep, Ham in diabatic

In this case, the SH populations won't change, but the TD-SE popultations may change - but regard this simply as the TD-SE evolution along a classical (adiabatic) trajectories. The evolved amplitudes do not affect the nuclear dynamics.

```
# Dynamics (Simulation) parameters
dyn_params.update({ "force_method":1, "rep_tdse":1, "rep_ham":0, "prefix":"T1-2-1-1-1", "tsh_method":-1, "rep_sh":1 })
dyn_params.update({"state_tracking_algo":2, "do_phase_correction":1 })

print(dyn_params)
res = dynamics.run_dynamics(q, p, iM, Cdia, Cadi, projectors, states, dyn_params, compute_model, {"model":1}, rnd)
plot_dyn(res)
```

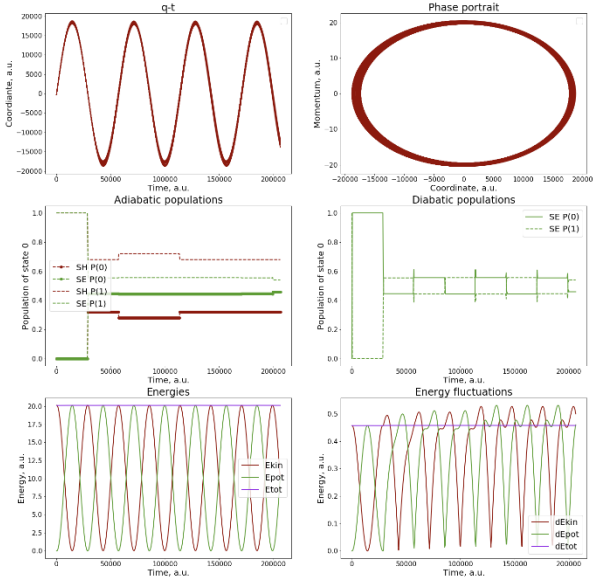


FSSH dynamics

Case 2.1.1.2.1. FSSH, TDSE is in adiabatic rep, Ham in diabatic

```
# Dynamics (Simulation) parameters
dyn_params.update({ "force_method":1, "rep_tdse":1, "rep_ham":0, "prefix":"T1-2-1-1-2-1", "tsh_method":0, "rep_sh":1 })
dyn_params.update({"hop_acceptance_algo":20, "momenta_rescaling_algo":201 })

print(dyn_params)
res = dynamics.run_dynamics(q, p, iM, Cdia, Cadi, projectors, states, dyn_params, compute_model, {"model":1}, rnd)
plot_dyn(res)
```



Acknowledgements

Funds:

UB startup

Computing resources:

OAC-NSF

Thank you! Questions?