University at Buffalo
The State University of New York

Excited States and Nonadiabatic Dynamics
CyberTraining School/Workshop 2023

Alexey Akimov

University at Buffalo, SUNY

June 13, 2023

University at Buffalo
The State University of New York

Structure of Libra package

Atomistic Workflows

libra_py
(All the Python modules)

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

University at Buffalo
The State University of New York

workflows
(self-explanatory)

dynamics
(self-explanatory)

compute_dynamics

This is the key function!

exact

- **compute**
- **plot**
- **save**

heom

Shakiba, M.; Smith, B.; Li, W.; Dutra, M.; Jain, A.; Sun, X.; Garashchuk, S.; Akimov, A. *Software Impacts* **2022**, *14*. https://doi.org/10.1016/j.simpa.2022.100445.

- **compute**
- **plot**
- **save**

qtag

Dutra, M.; Garashchuk, S.; Akimov, A. V. *Int. J. Quantum Chem.* ***2023***, e27078. https://doi.org/10.1002/qua.27078.

- **initialize**
- **compute**
- **plot**
- **save**

tsh

- **compute**
  - **generic_recipe** – initializes adiabatic/diabatic variables. Calls `run_dynamics`
  - **run_dynamics** – sets up the default parameters, initializes Hamiltonains, saves results, etc. Calls `compute_dynamics`
- **plot**
  - **plot_dynamics** – the one to be used the most
  - **plot_dyn** – to plot from the hdf5 files
  - **plot_dyn_old** – deprecated
  - **hdf2xyz** – convert data from hdf to xyz-formatted strings
- **save**
- **…**

libra_py
(All the Python modules)

workflows

nbra
(neglect of back-reaction approximation)

- **step2:** time-overlaps & NAC calculations
- **step2_many_body:** SP time-overlaps & NAC calculations, and CI info
- **step2_dftb:** time-overlaps & NAC calculations for DFTB+ specifically
- **step2_ergoscf:** time-overlaps & NAC calculations for ErgoSCF specifically
- **step2_analysis:** compute compositions of the MB states, spectra for QE, etc.

dynamics
(self-explanatory)

- **mapping**: functions for mapping single-particle properties (e.g. couplings) to the many-body ones. Most of it is to be replaced by `**mapping2**`
- **mapping2**: new, simplified and cleaned up version of the mapping module.
- **step3**: phase corrections, re-indexing, SD-basis properties, etc.
- **step3_many_body**: functions for computing NACs in the MB basis.
- **step4**: to be deprecated. use `**compute_dynamics**` instead

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

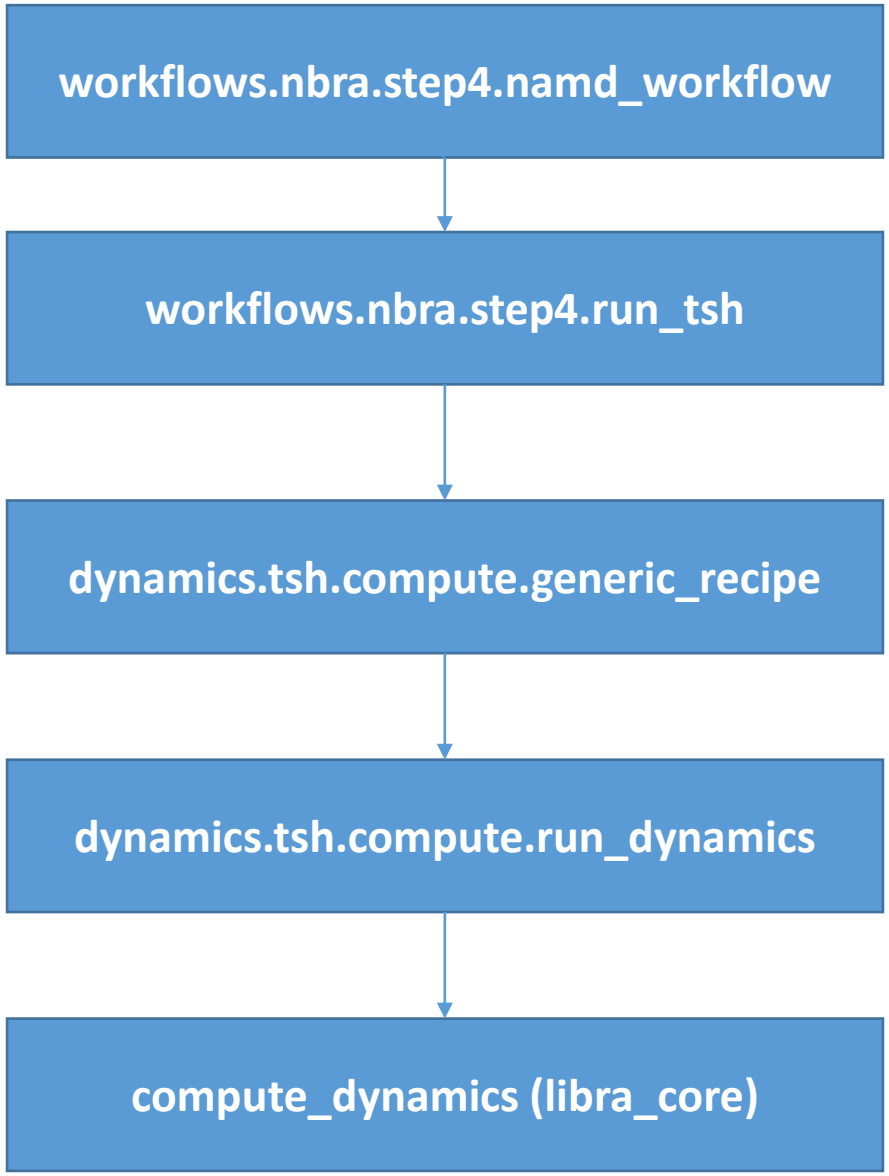
compute_dynamics
This is the key function!

- **ann:** complete module for training ANN on the NBRA data and for forecasting it for longer times
- **decoherence_times:** to compute dephasing times and influence spectra from the NBRA data
- **qsh:** module for quasi-stochastic Hamiltonian construction
- **lz:** module for Belyaev-Lebedev-Landau-Zener calculations
- **etc**

University at Buffalo
The State University of New York

Brief Overview of the Deprecated step4

University at Buffalo
The State University of New York



Parallelization, multiple methods/initial conditions

Initialization of nuclear variables

Initialization of electronic variables, transformation to the desired representation, nHamiltonian object construction and initialization

Initialization of default parameters, writing files, computing some observables, thermostat variables, iteration over nuclear timesteps

TSH/Ehrenfest and decoherence algorithms, trajectory coupling, Hamiltonian properties updates (calling external Python functions)

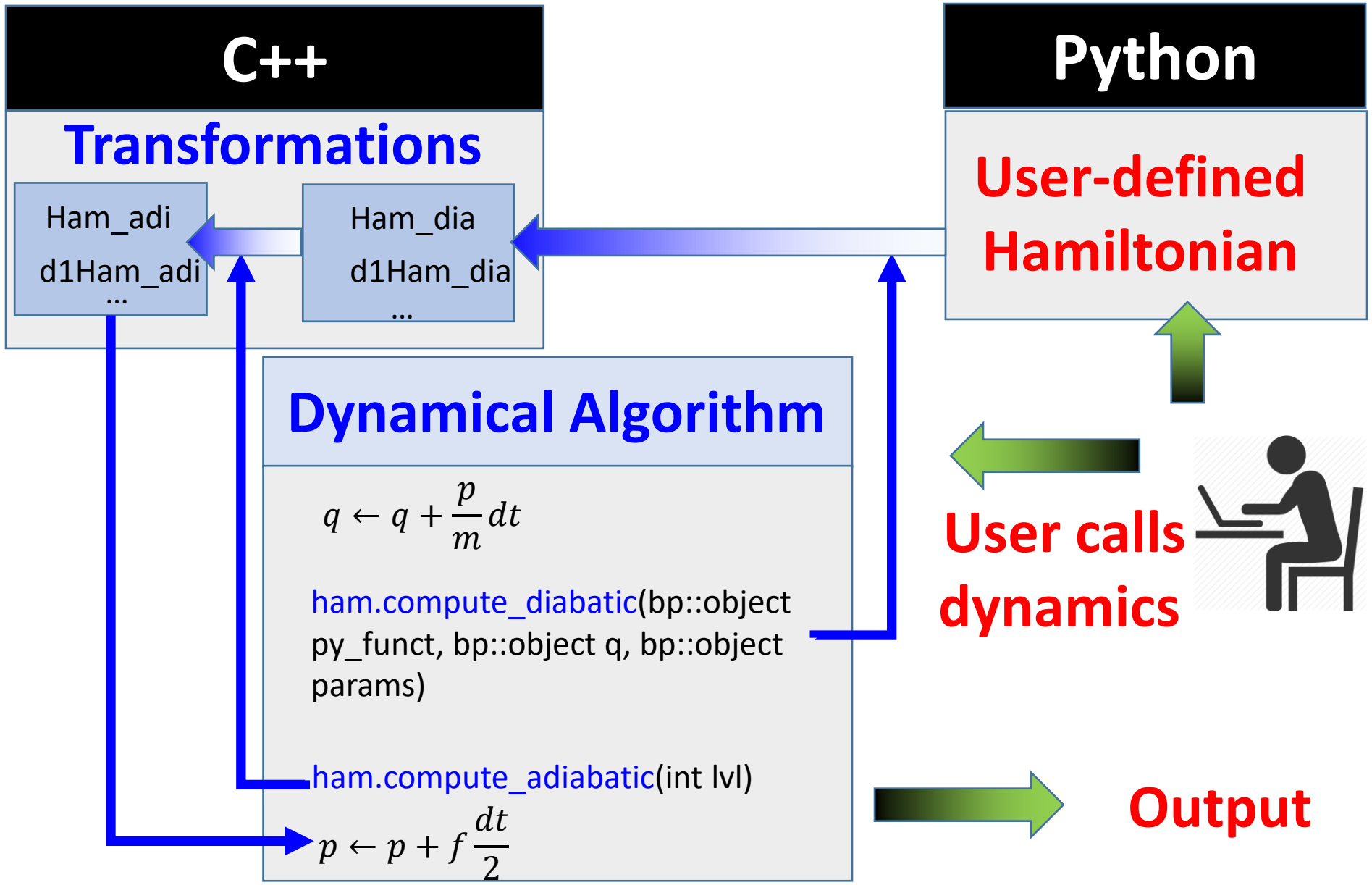
University at Buffalo
The State University of New York

More on the nHamiltonian class.

Making Interfaces

How `compute_dynamics` works

University at Buffalo
The State University of New York



Different ways of computing matrix elements. Example of H_{dia}^{vib}

University at Buffalo
The State University of New York

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**(…)						

Different ways of computing matrix elements.
Example of H_{adi}^{vib}

University at Buffalo
The State University of New York

Function	Q	P	S	H_{dia}	∇H_{dia}	D_{dia}	U	H_{adi}	∇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 class as a hierarchical data type to handle multiple trajectories

University at Buffalo
The State University of New York

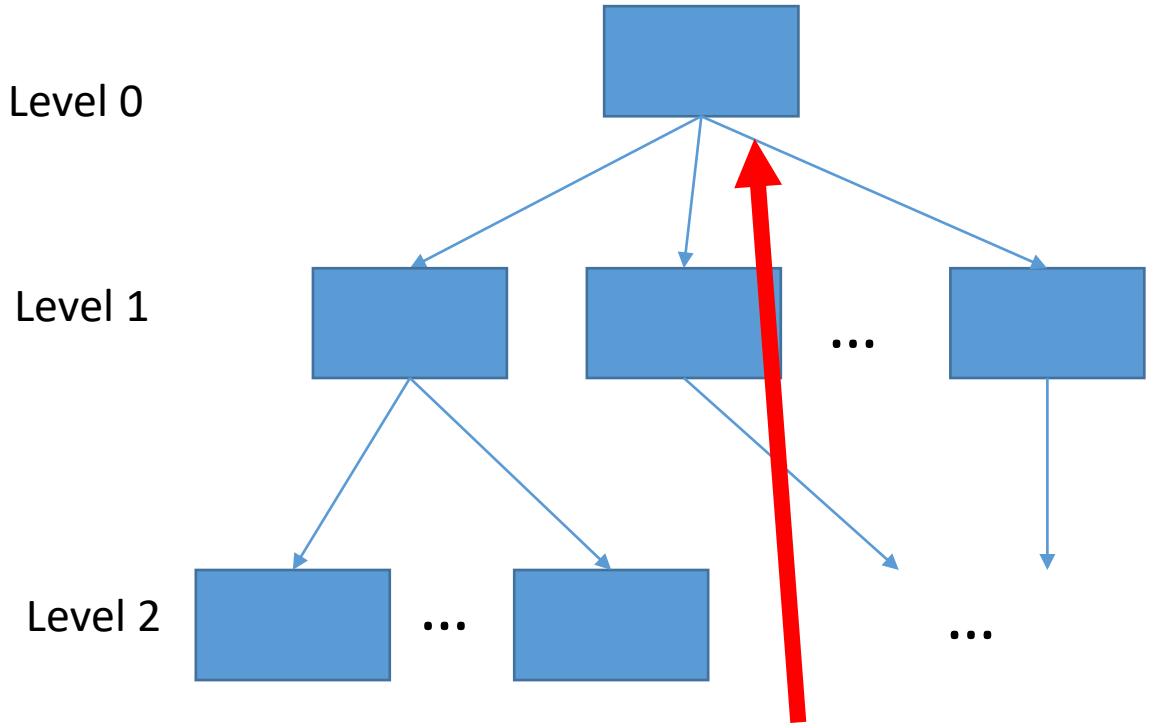
nHamiltonian

- 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



int entanglement_opt

A selector of a method to couple the trajectories in this ensemble.
- 0: no coupling [default]
- 1: ETHD
- 2: ETHD3 (experimental)
- 22: another flavor of ETHD3 (experimental)

Packing variables for multiple trajectories

University at Buffalo
The State University of New York

Individual trajectory

Swarm of uncoupled trajectories

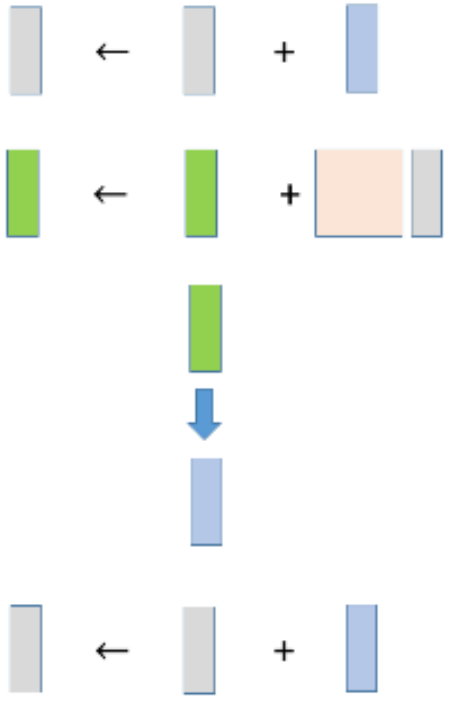
Swarm of coupled trajectories

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

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

Forces update

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



Keep the 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

University at Buffalo
The State University of New York

```
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