University at Buffalo
The State University of New York

Excited States and Nonadiabatic Dynamics CyberTraining School/Workshop 2023

Alexey Akimov

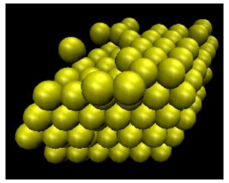
University at Buffalo, SUNY

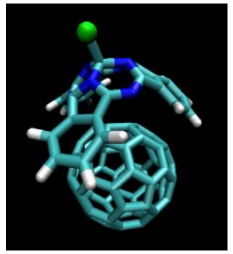
June 12, 2023

Libra overview

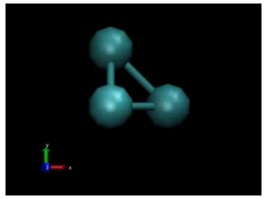
Libra History

Classical MD





Rigid body MD

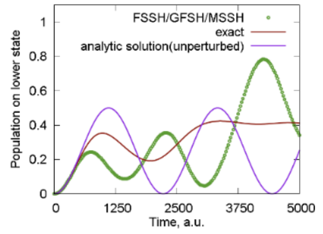


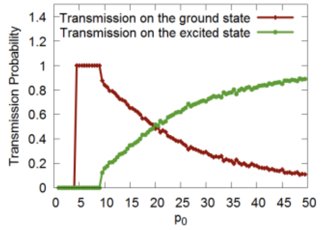
- Simplectic integrators for classical DOFs
- Thermostats
- Barostats
- Force fields
- ANN
- Chemical object representation

2007-2011 (LCCCS)

Akimov, Prezhdo, *JCTC*, **2013**, 9, 4959.
Akimov, Prezhdo, *JCTC*, **2014**, 10, 789

Ehrenfest & TSH



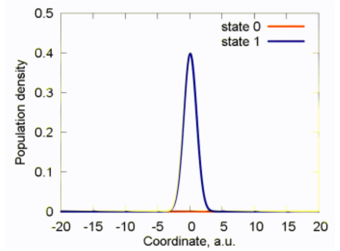


- Simplectic integrators for TDSE
- TSH, Ehrenfest, stand-alone scripts
- Decoherence methods
- Model problems
- Added HF and EHT to LCCCS
- Interface with VASP, then QE

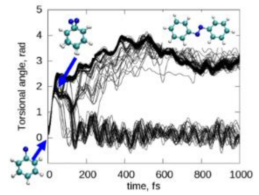
2011-2015 (Pyxaid)

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

DVR



Back-reaction



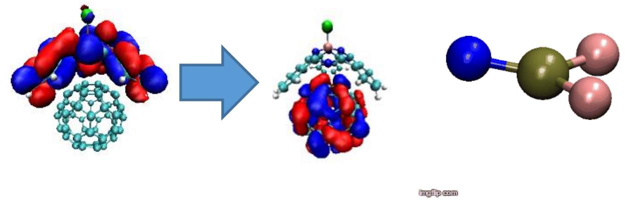
- Modularization and revision
- DVR methods
- Semiempirical Hamiltonians
- Molecular integrals
- Decoherence methods, TSH

2015/2016 (Libra)

Libra-X (with Drs. Ryoji Asahi, Kosuke Sato, Ekadashi Pradhan)

Sato, Pradhan, Asahi, Akimov *PCCP* **2018,** 20, 25275
Pradhan, Sato, Akimov *J. Phys.: Condens. Matter*, **2018**, 30, 484002

- Interfaces with GAMESS, QE
- Added back-reaction for QE
- More modularization



Pyxaid2 (with Prof. Wei Li)

Li, Zhou, Prezhdo, Akimov *ACS Energy Lett*, **2018**, 3, 2159

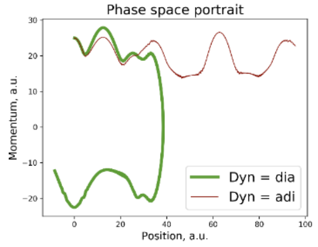
- SOC, multiple k-points, etc.

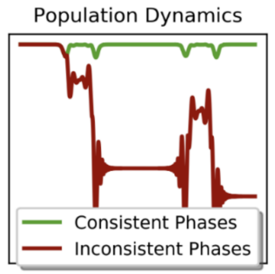
2018 (Pyxaid2, Libra-X)

Libra History

Phase correction for NACs

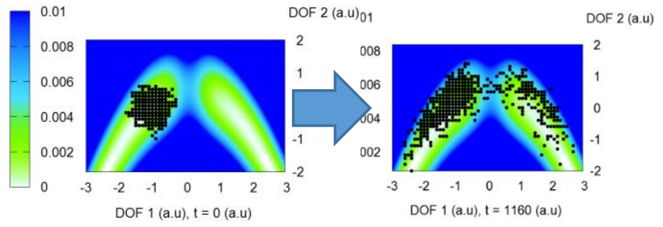
Akimov *JPCL* **2018** 9, 6096-6102





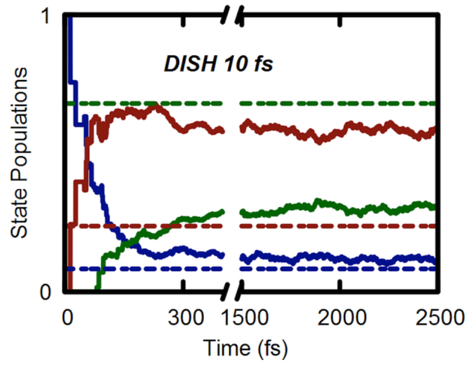
Entangled trajectories

Smith, Akimov *JCP* **2018**, 148, 144106



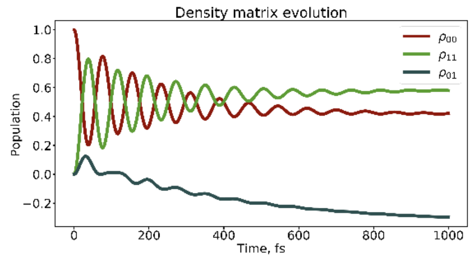
Bastida's Boltzmann-corrected Ehrenfest, mSDM

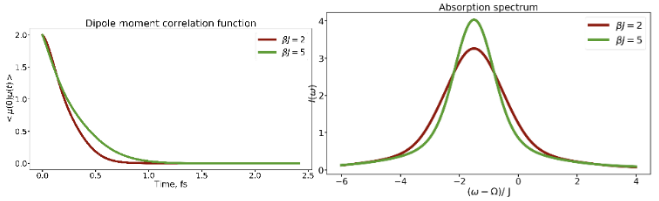
Smith; Akimov *JCP* **2019**, 151, 124107



HEOM

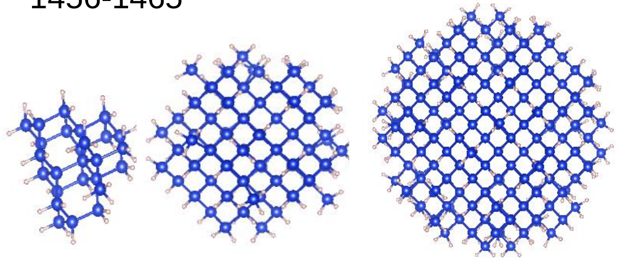
Temen, Jain, Akimov *Int. J. Quant. Chem.*, **2020**, 120, e26373





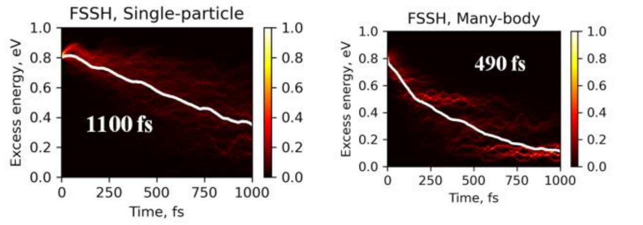
Belyaev-Lebedev LZ method

Smith, B.; Akimov, A. V *JPCL* **2020**, 11, 1456-1465



Many-body NA-MD

Smith, B.; Shakiba, M.; AVA *JCTC* **2021**, 17, 678
Smith, B.; Shakiba, M.; AVA *JPCL* **2021**, 12, 2444

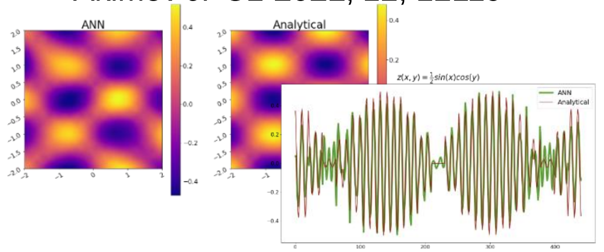


Revised DISH, new workflows

Akimov *JCP* **2021**, 155, 134106

Machine Learning revised. TD-ML approach

Akimov *JPCL* **2021**, 12, 12119



2018 **2019** **2020** **2021**

Libra Philosophy/Vision

University at Buffalo
The State University of New York

- **modular**

  Maximize and simplify the re-use, OOP

- **versatile**

  linear algebra, integrals,
  quantum and classical mechanics/dynamics,
  nonadiabatic methods, surface hopping,
  IO utilities, model preparation and analysis

- **"methodology discovery"** (prototyping)

  - Use with model problems and atomistic simulations
  - Python – for convenience, C++ - for efficiency

- **practical**

  Fully-functional tool that can be applied to real (atomistic) systems to study materials

- **user-friendly & documented**

  The code is convenient to users and they have plenty resources – examples and documentation

- **community tool**

  - A platform to adopt the past and latest developments
  - The developers can understand and contribute to the code

Libra Motivation

University at Buffalo
The State University of New York

- Many codes (Newton-X, SHARC, NEXMD, FIERBALL, JADE, MOLPRO, PYXAID, PYUNIXMD, …)
  - Black-box. Difficult to re-use to formulate other methods, etc.
  - Limited functionality (high focus, e.g. atomistic of special kind)
- Many methods (FSSH, DISH, A-FSSH, QTAG, QTSH, etc.)
  - Not always available
  - Not always user-friendly (e.g. my experience with PYXAID prototype)
  - Not always portable/modular, lack of best coding standards, no version control, etc.
  - Limited consistency of different codes
  - Possible redundancies even in the same code

Libra Motivation

University at Buffalo
The State University of New York

- Adopt the best practices
    - Modularity (e.g. PySCF, Psi4NumPy, PyQuante, HORTON)
    - Language standards (Python, C++ vs. Fortran? Hybrid programming)
    - Testing & Documentations (pytest, unittest, Doxygen/Sphinx)
    - User/developer training (Workshops, Summer/Winter schools)
- Focus on the community
    - Every group has expertise in their field – rely on that
    - Community contributions – PR on GitHub
    - Use version control and collaborative workflows via GitHub, Issues
    - Frequent communication and close collaboration e.g. via Slack

Community Tool: Code Contributions/Integration

Amber Jain – Hierarchical Equations of Motion (HEOM)
https://github.com/amber-jain-group-iitb/heom_amber

src/dyn/heom

Xiang Sun – (Non)-equilibrium Fermi Golden Rule (FGR)
https://github.com/tsiangsun/FGR

src/fgr

Nandini Ananth – Initial value representation (IVR)
https://github.com/AnanthGroup/SC-IVR-Code-Package

src/ivr

Sophya Garaschchuk – quantum trajectory guided Gaussians (QTAG)

src/libra_py/dynamics/qtag

Craig Martens – quantum trajectory surface hopping (QTSH)

in progress

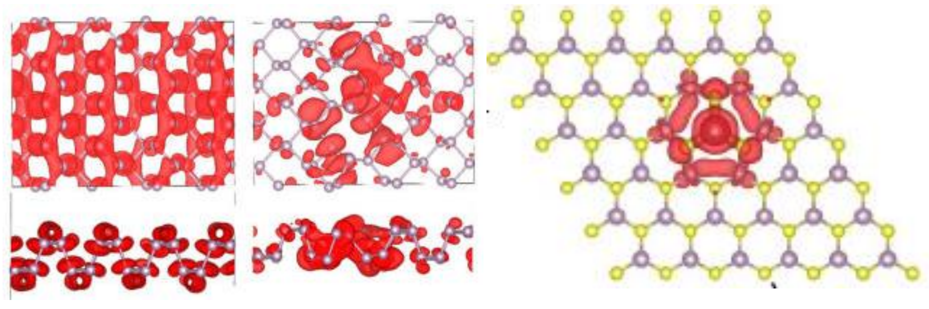
… and more

Practical: Libra in Materials Research

University at Buffalo
The State University of New York

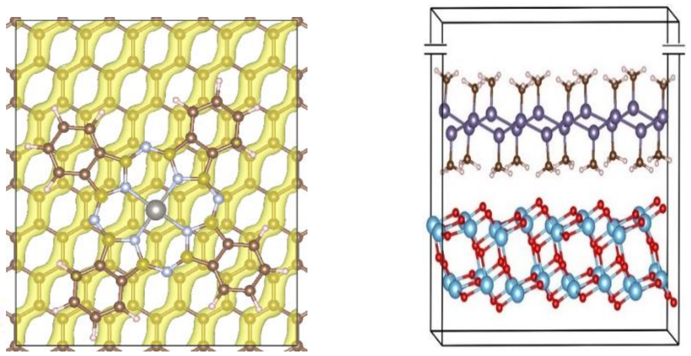
2D systems

Long el al. *JPCL* **2016**, *7*, 653.

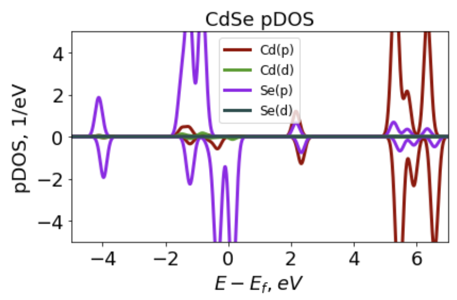


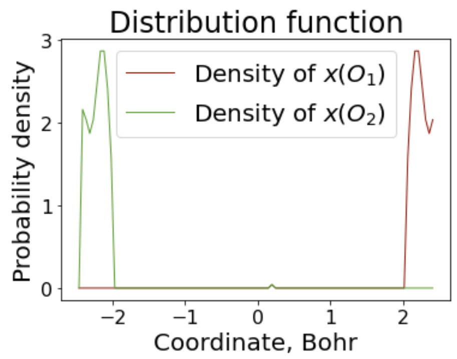
2D heterojunctions

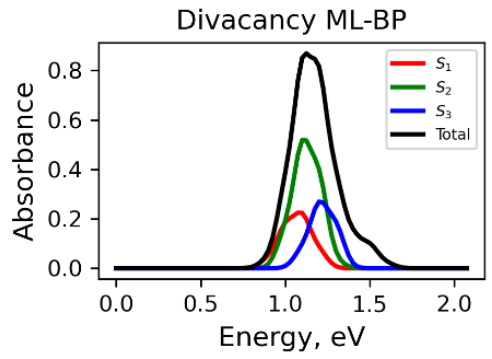
Nijamudheen, A.; AVA *JPCC*, 2017, *121*, 6520



Auxiliary Analysis Tools

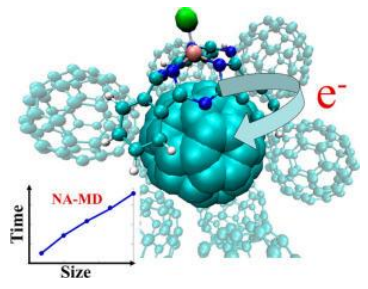






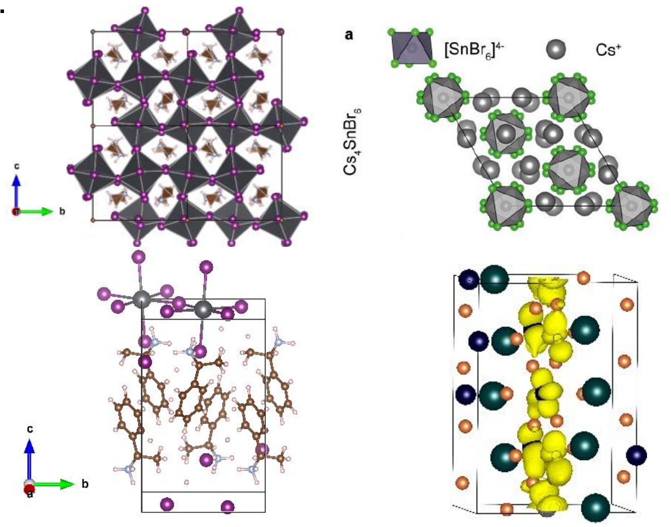
Organic heterojunctions

Sato et al. *PCCP,* **2018**, 20, 25275.



Perovskites

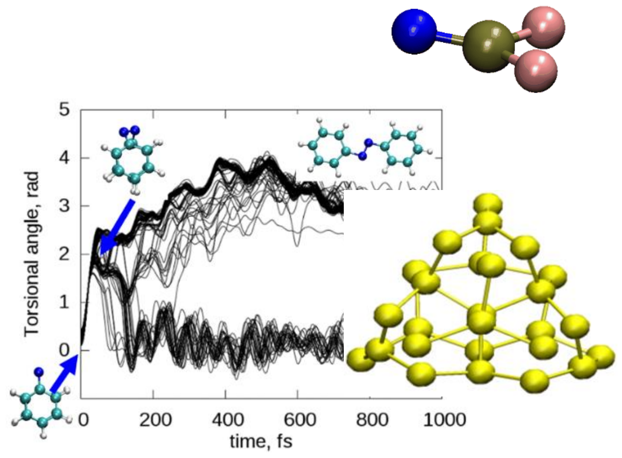
Nijamudheen, A.; AVA *JPCL* **2018**, *9*, 248



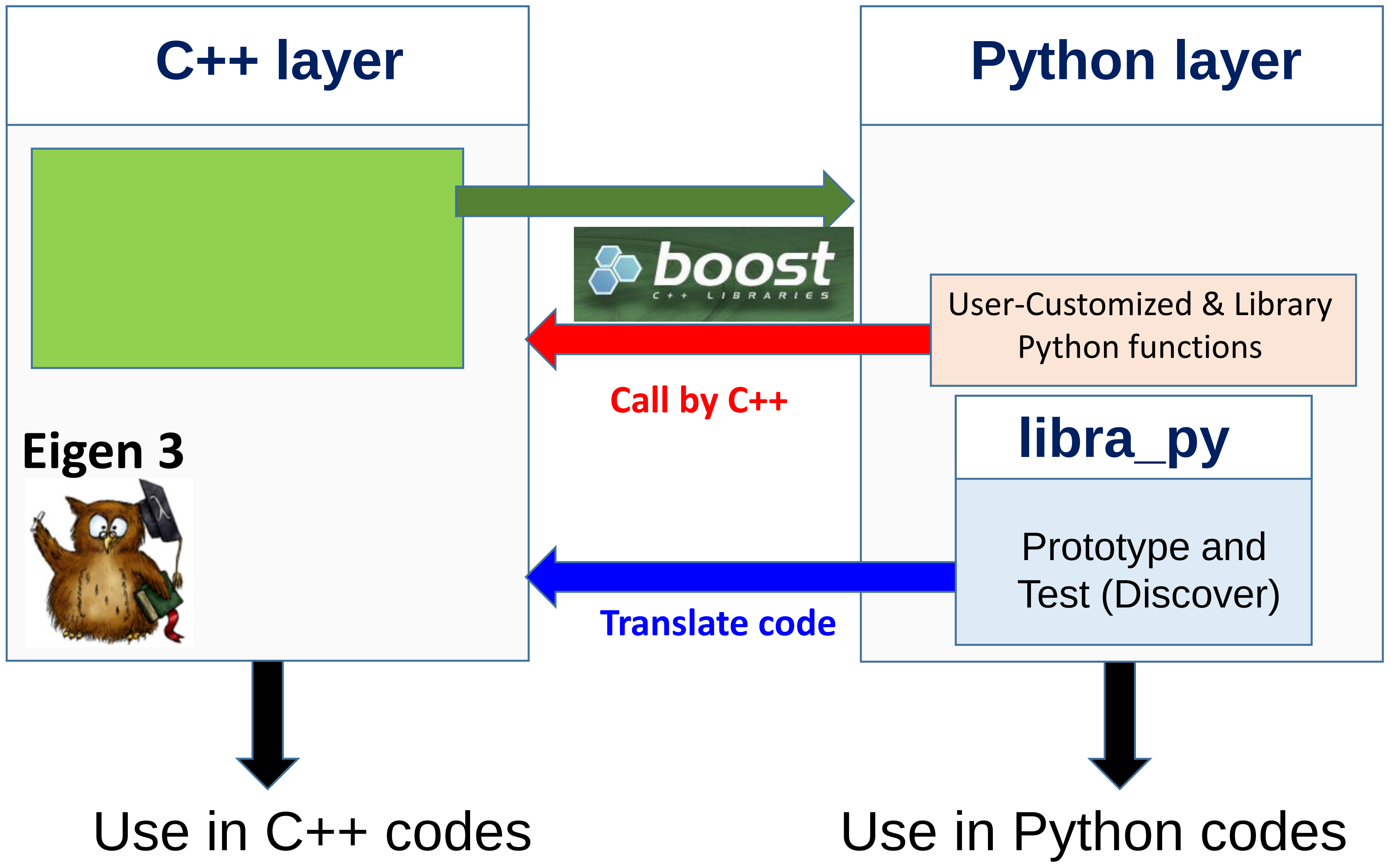
Quantum Dots & Molecules

Lin, Y.; AVA *JPCA.* **2016**, 120, 9028

Pradhan et al. *JPCM,* **2018**, 30, 484002



C++/Python Interoperability



Modularity: API Diversity

- The goal is to suite the needs of the **users of various levels**
- Find a balance between **simplicity** and **flexibility**

Developer/Efficiency

```
double gaussian_overlap( AO* AOa, AO* AOb,int is_normalize, int is_derivs,  VECTOR& dIdA, VECTOR& dIdB,
vector<double*>& auxd,int n_aux);

double gaussian_overlap( AO* AOa, AO* AOb,int is_normalize, int is_derivs, VECTOR& dIdA, VECTOR& dIdB );

double gaussian_overlap(AO* AOa, AO* AOb,int is_normalize);

double gaussian_overlap(AO* AOa, AO* AOb);
```

User/Convenience

Example

Computing kinetic energy between Gaussians

```
g1 = PrimitiveG()
g2 = PrimitiveG()
g1.init(n1,m1,k1, a1, VECTOR(x1, y1, z1))
g2.init(n2,m2,k2, a2, VECTOR(x2, y2, z1))

kin = kinetic_integral(g1,g2)
```

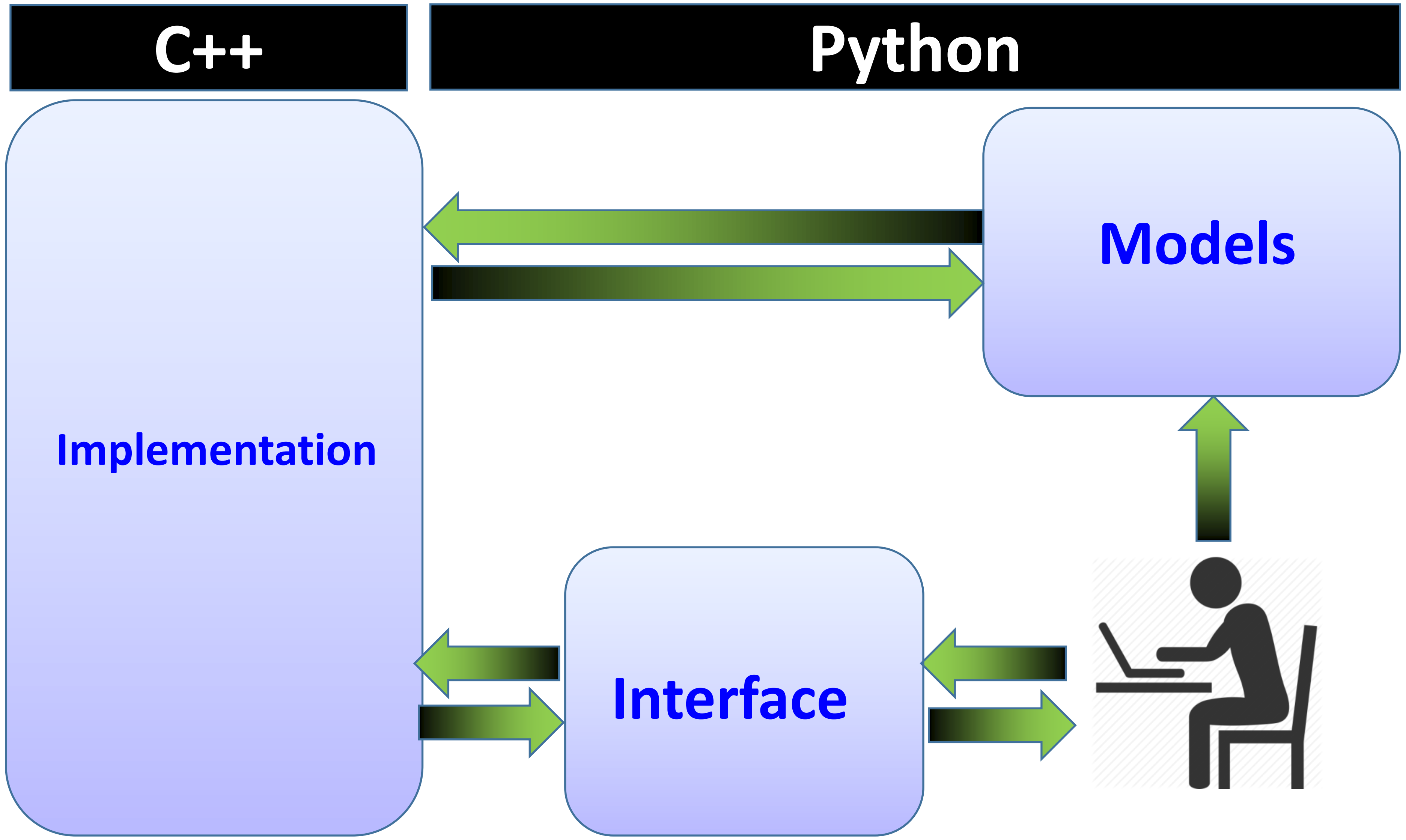
Benchmarked against PyQuante

```
p1 = PyQuante.PGBF.PGBF(a1,(R1.x,R1.y,R1.z),(n1,m1,k1))
p2 = PyQuante.PGBF.PGBF(a2,(R2.x,R2.y,R2.z),(n2,m2,k2))

val_ref = p1.kinetic(p2)
```

Passing Python functions

University at Buffalo
The State University of New York



How it works with Sampling

University at Buffalo
The State University of New York

```
vector<MATRIX> metropolis_gau
(Random& rnd, bp::object target_distribution,
MATRIX& dof, bp::object distribution_params,
int sample_size, int start_sampling, double gau_var){
```

Metropolis Algorithm

```
double p_old =
bp::extract<double>( target_distribution(s_old,
distribution_params) );

...
}
```

C++

```
def test():
   q  = MATRIX(ndof, 1)
   output = metropolis_gau( piab, q, params, ...)
```

User calls the sampling

Output

```
def piab(q, params):
```

User defines the probability density

Python

Example

University at Buffalo
The State University of New York

User defines how to run the MC sampling

```
q = MATRIX(1,1);  q.set(0, 0.5)
params = {"k":1.0, "m":2000.0, "states":[0], "coeffs":[1.0]}
Nsamp = 1000000; Nstart = 50000
sampling = metropolis_gau(rnd, HO_sup, q, params, Nsamp,Nstart, 0.05)
bin(sampling, -1.5, 2.0, 0.01, 0, 0, "_distrib-1.txt")
```

User defines what probability distribution function is to be sampled

```
def HO_sup(q, params):
    k = params["k"];    m = params["m"];
    states = params["states"];    coeffs = params["coeffs"]
    x = q.get(0)
    sz = len(states)
    p = 0.0
    for n in xrange(sz):
        p = p + coeffs[n] * ket_n(x, states[n], k, m)
    p = p * p
    return p
```

The dynamical algorithm is in C++, but…

Don’t need to implement the model in C++

Initial Conditions: Metropolis Sampling

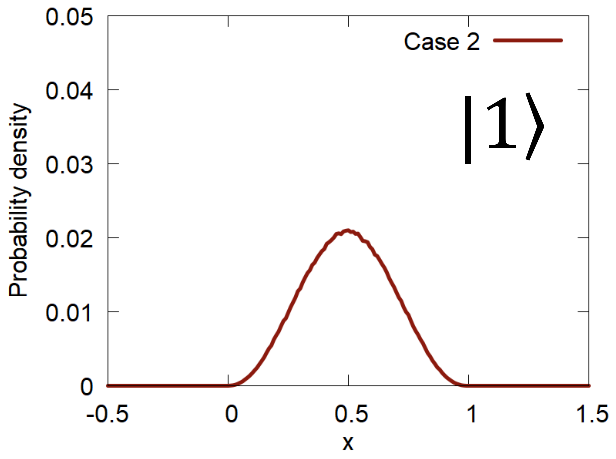
University at Buffalo
The State University of New York

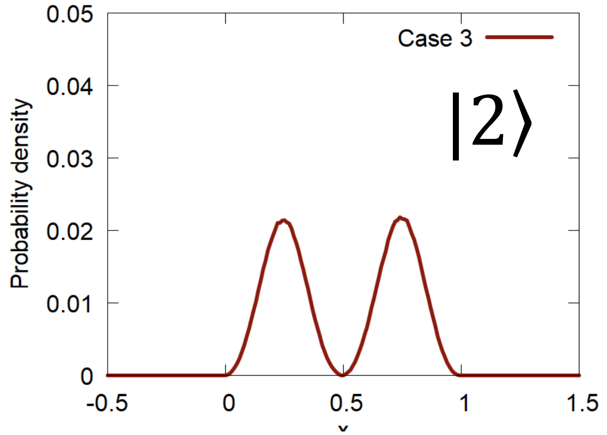
Particle in a box

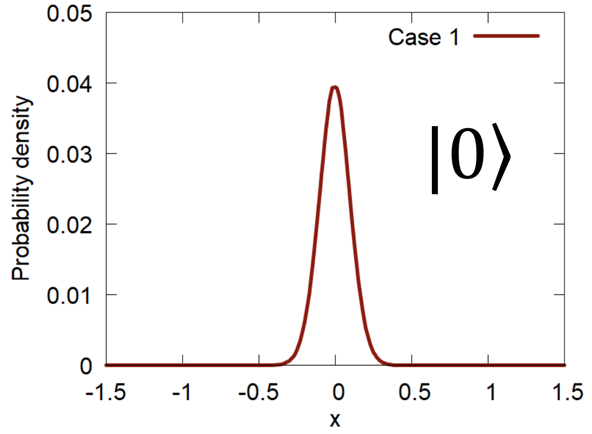
$$\psi_n(q) \sim sin\left(\frac{\pi n q}{L}\right)$$

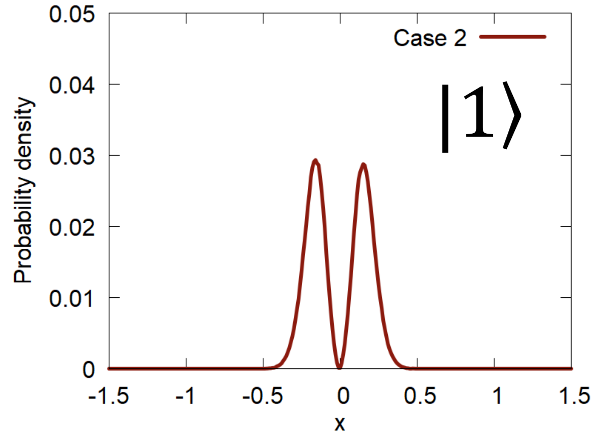
Harmonic oscillator

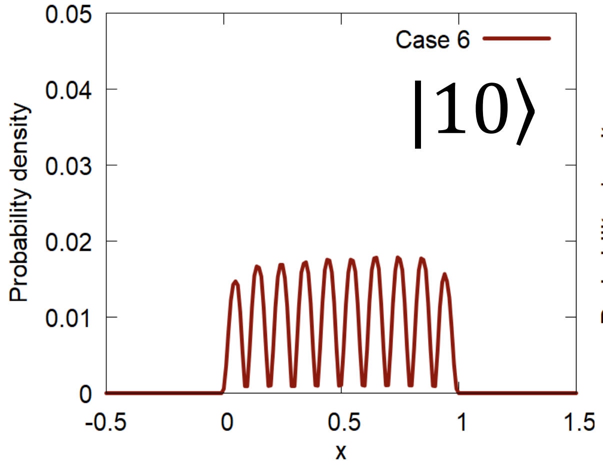
$$\psi_n(q) \sim H_n(q\sqrt{\alpha})\exp\left(-\frac{\alpha q^2}{2}\right)$$

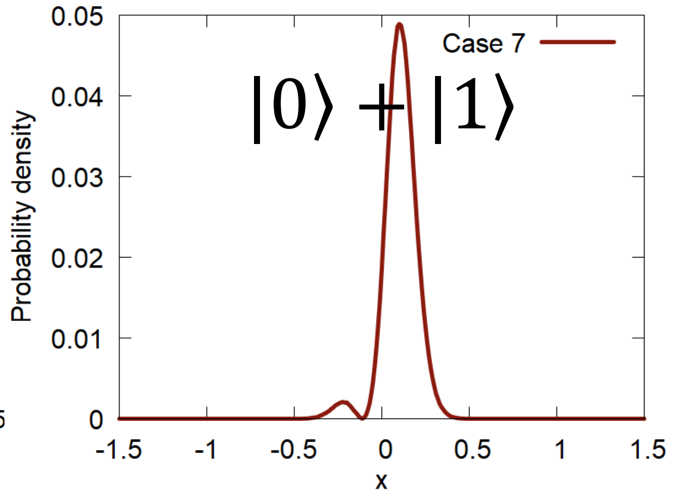


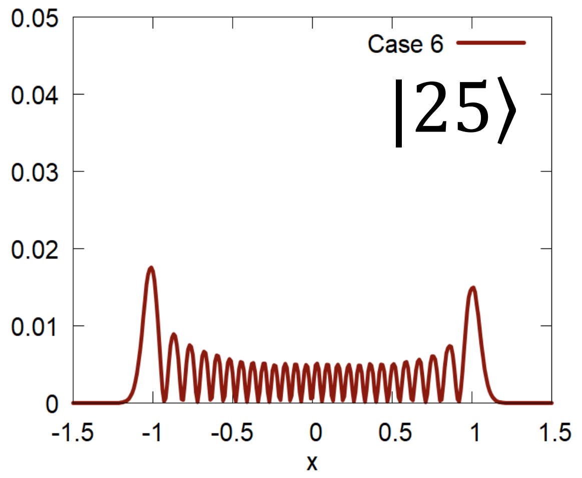


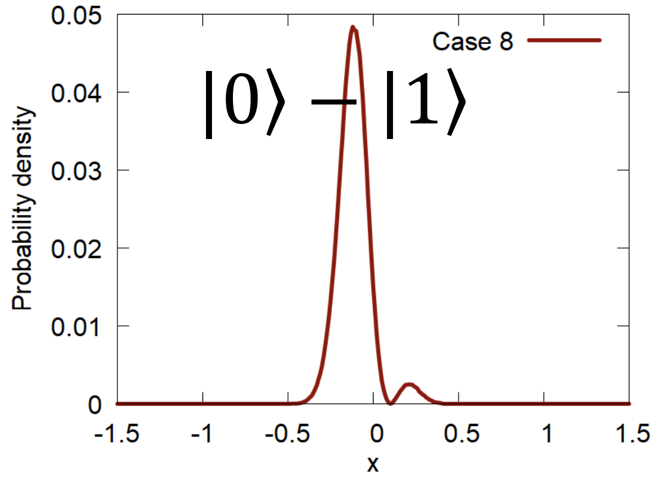












Why Contribute?

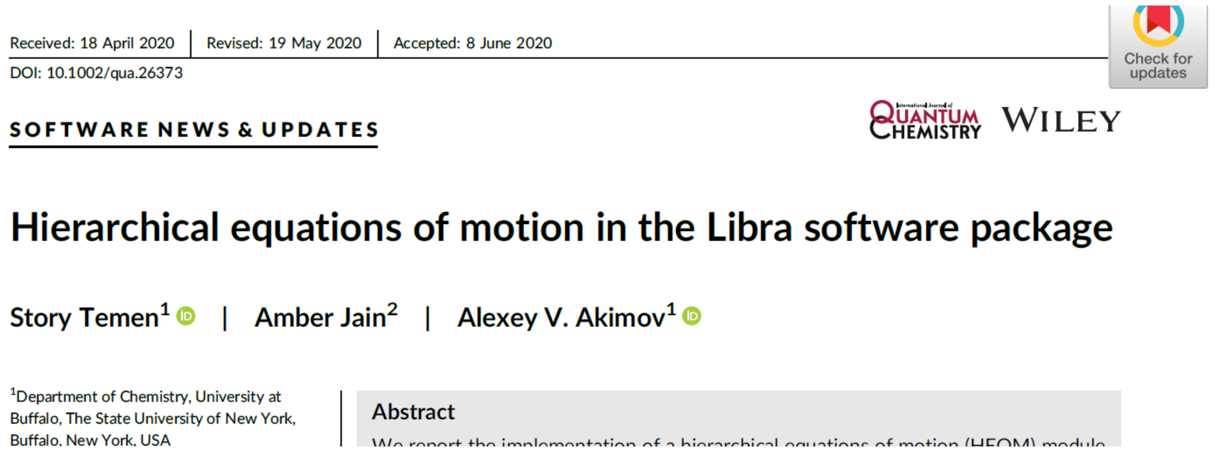
University at Buffalo
The State University of New York

- Make your methods broadly available (mutually advertise)
- Be listed on the developers/contributors lists, get credentials, get citations

- Make your methods compatible with other methods, enable easier interfacing – this facilitates new methods developments
- Take advantage of other methods/functions/data types available in the same code – learn once then swim
- Take advantage of improvements of other parts of the code
- Mutually ensure best standards and facilitate bug discovery/testing

How to Contribute?

University at Buffalo
The State University of New York



Received: 18 April 2020 | Revised: 19 May 2020 | Accepted: 8 June 2020
DOI: 10.1002/qua.26373

Check for updates

SOFTWARE NEWS & UPDATES

International Journal of Quantum Chemistry WILEY

Hierarchical equations of motion in the Libra software package

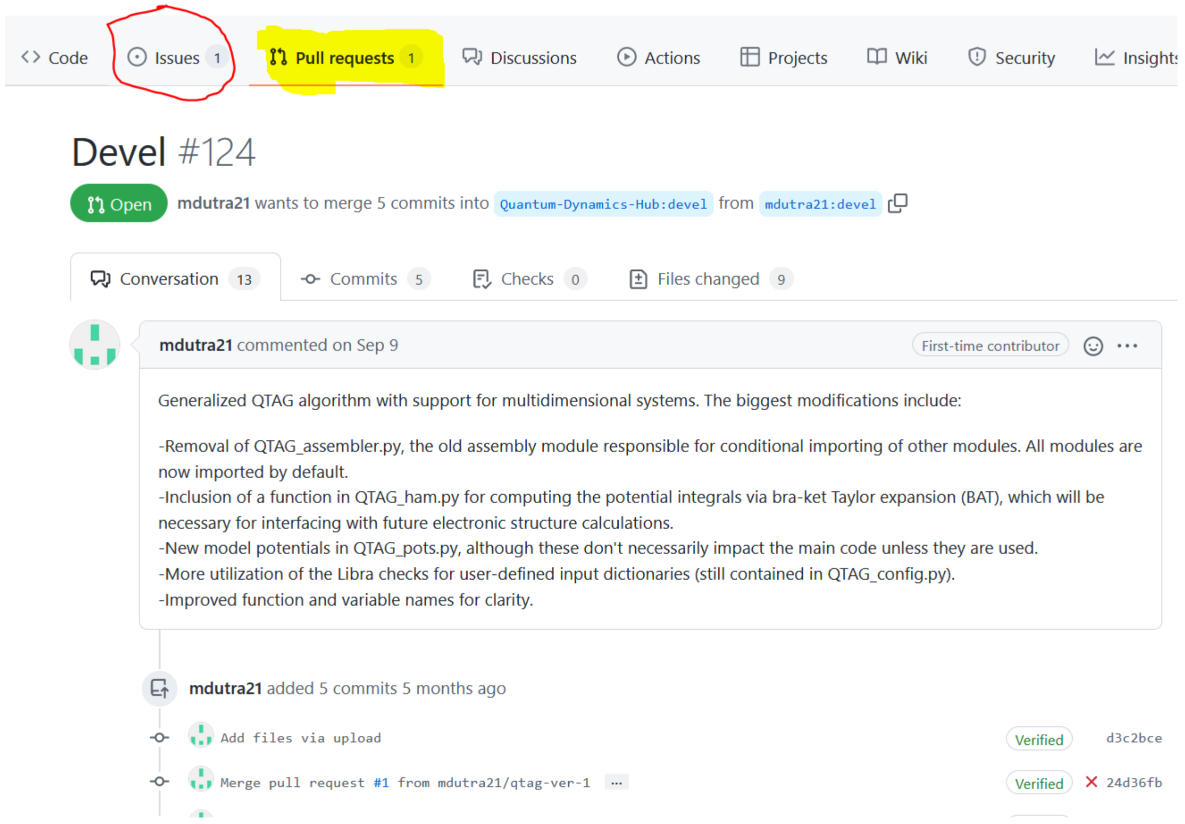
Story Temen[1] | Amber Jain[2] | Alexey V. Akimov[1]

[1]Department of Chemistry, University at Buffalo, The State University of New York, Buffalo, New York, USA

Abstract

We report the implementation of a hierarchical equations of motion (HEOM) module

https://github.com/amber-jain-group-iitb/heom_amber



Devel #124

Open mdutra21 wants to merge 5 commits into `Quantum-Dynamics-Hub:devel` from `mdutra21:devel`

mdutra21 commented on Sep 9 First-time contributor

Generalized QTAG algorithm with support for multidimensional systems. The biggest modifications include:

-Removal of QTAG_assembler.py, the old assembly module responsible for conditional importing of other modules. All modules are now imported by default.
-Inclusion of a function in QTAG_ham.py for computing the potential integrals via bra-ket Taylor expansion (BAT), which will be necessary for interfacing with future electronic structure calculations.
-New model potentials in QTAG_pots.py, although these don't necessarily impact the main code unless they are used.
-More utilization of the Libra checks for user-defined input dictionaries (still contained in QTAG_config.py).
-Improved function and variable names for clarity.

mdutra21 added 5 commits 5 months ago

```
Add files via upload                              Verified    d3c2bce
Merge pull request #1 from mdutra21/qtag-ver-1 …   Verified  ✕ 24d36fb
```

Before:

- hard-coded inputs (recompile for all parameters)
- Fortran into executable

After:

- General-purpose code, any inputs
- Python/C++, integrate with the plotting scripts, etc.

- create a pull-request
- open an issue
- start a discussion (haven't tried yet)