Excited States and Nonadiabatic Dynamics CyberTraining School/Workshop 2022

Alexey Akimov

University at Buffalo, SUNY

July 4, 2022

General Workshop Goals

and

Overview of the CyberTraining Infrastructure

Objectives and Agenda

University at Buffalo
The State University of New York

CyberTraining: Pilot: Modeling Excited State Dynamics in Solar Energy Materials

Workshop Objectives

- Get **familiar with a variety of software packages** relevant to modeling of excited states and nonadiabatic dynamics
- Get an overview of **theoretical background** for corresponding computational methods
- Get a **practical experience** with these tools and packages

Keywords and topics:

- nonadiabatic dynamics
- excited states
- quantum dynamics
- quantum-classical methods
- charge transfer
- excitation energy transfer
- trajectory surface hopping
- coupled trajectories
- exact factorization
- TD-DFT, CASSCF, GW/BSE
- algorithms and methods
- software, programming, Python
- best practices, Git, GitHub

This year

- pyUNIxMD (Min)
- CT-MQC (Ibele)
- SHARC (Mai)
- SHARC/COBRAMM (Avagliano)
- OpenMolcas (Mai, Avagliano)
- ORCA (Mai)
- Hefei-NAMD (Zhao, Chu)
- Quantum Espresso (Zhao, Chu)
- BerkeleyGW and paratec (Zhang)
- DynEMol (Rego)
- Libra (Akimov)
- DFTB+ (Shakiba)
- CP2K (Shakiba)
- TBD (Kilin)

Last year

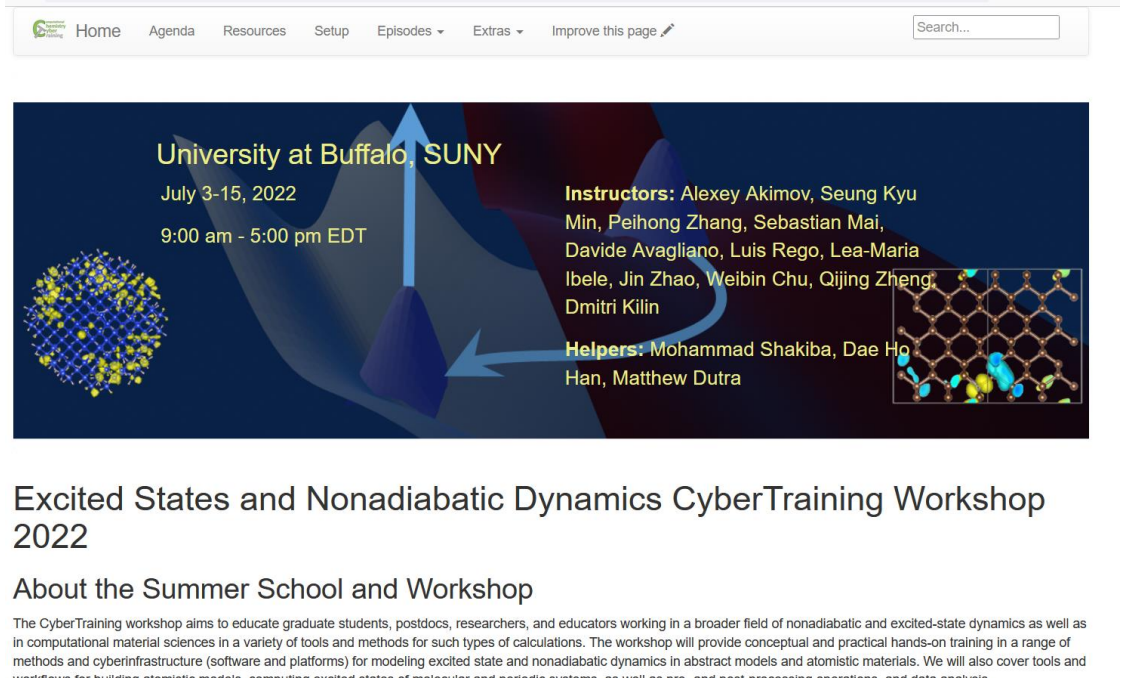
- Libra (Akimov)
- NEXMD (Tretiak)
- Newton-X (Barbatti)
- nano-qmflows (Infante, Zapata)
- CAT, auto-FOX (Infante, Zapata)
- COLUMBUS (Lischka)
- DFTB+
- CP2K
- Quantum Espresso
- ErgoSCF

The Plan & Resources

University at Buffalo
The State University of New York

All the details are here:

https://compchem-cybertraining.github.io/Cyber_Training_Workshop_2022/



Join Slack:

- Members can invite new members
- Private and public channels, direct (private) messages, conversations
- Any time, but no strings attached

https://join.slack.com/t/quantumdynamicshub/shared_invite/zt-mjbhjssx-GGhsbYHxeBMvhmumK_j7LA

VPN and Accounts:

- 2-factor authentication
- submit a ticket: https://ubccr.freshdesk.com/support/home

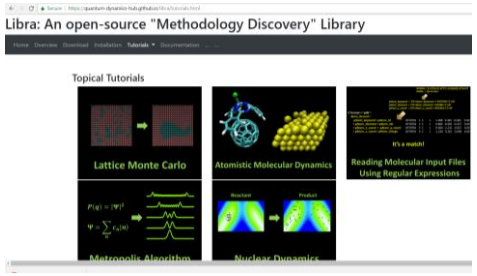
More Resources

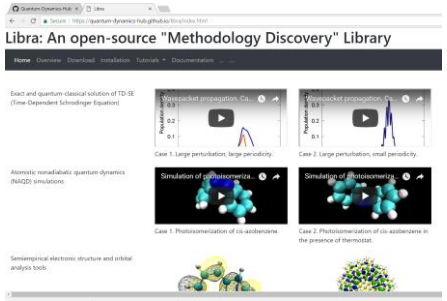
University at Buffalo
The State University of New York

Codes: https://github.com/Quantum-Dynamics-Hub
Training: https://github.com/compchem-cybertraining

Quantum Dynamics Hub
https://quantum-dynamics-hub.github.io/

Libra website: https://quantum-dynamics-hub.github.io/libra/index.html





Libra tutorials:
https://github.com/compchem-cybertraining/Tutorials_Libra

CP2k (and CP2k/Libra) tutorials:
https://github.com/compchem-cybertraining/Tutorials_CP2K

we'll be using them

Summer 2021 workshop:
https://compchem-cybertraining.github.io/Cyber_Training_Workshop_2021/

Winter school:
https://compchem-cybertraining.github.io/Libra_Winter_School_2022/

Summer 2022 (This!) workshop:
https://compchem-cybertraining.github.io/Cyber_Training_Workshop_2022/

Daily Schedule

University at Buffalo
The State University of New York

Daily

- Breakfast = hotel
- 9:00 am – 12:00 pm: Morning session (Recording)

- 12:00 – 1:30 pm Working lunch/rest – on your own at “Commons”, rest, discuss, collaborate (No Recording)

- 1:30 pm – 5:00 pm: Afternoon session (Recording)

- After 5:00 pm: collaborations/on your own, dinner on your own

Locations

- Classes are @: July 4 – 7 Clemens 120; 8 July – Alumni Arena, Alumni 90, July 11-15 Talbert 107

Logistic

University at Buffalo
The State University of New York

- We cover your stay
- Travel for the US participants, partially the international participants – we'll need the paperwork
- The forms will be distributed to you via Slack channel – please DON't sent them back via e-mail
- Stipends to cover the rest of expenses, please keep your receipts just in case.
- We'll need a confirmation that you aren't getting reimbursements from your institutions.
- A lot of paperwork later – likely it'll be just me handling most of the stuff
- Prizes: $300 (1 first prize), $200 (3 second prizes), $100 (5 first prizes) – the project competition. Online and in-person participants are eligible

Project

University at Buffalo
The State University of New York

Project rules

- Consist of: a) short written report, b) presentation at the last day of workshop; c) set of input/output files deposited on the GitHub repository
- Should actively involve one of the packages discussed over the workshop period
- Preferably not something you have an extensive experience with
- Doesn't have to be a full-scale research project, but can be a step towards this direction
- Projects completed using local or home institution resources are eligible
- Can be an application or a coding project
- The consistency in your course work during this school will contribute to your chances to win the awards
- The awards decisions will be made based on the committee evaluation.

- Submit your project via GitHub by July 21
- Oral presentation – tentatively July 22, via Zoom

Check out the past year projects:

https://github.com/compchem-cybertraining/Cyber_Training_Workshop_2021/tree/gh-pages/course_work

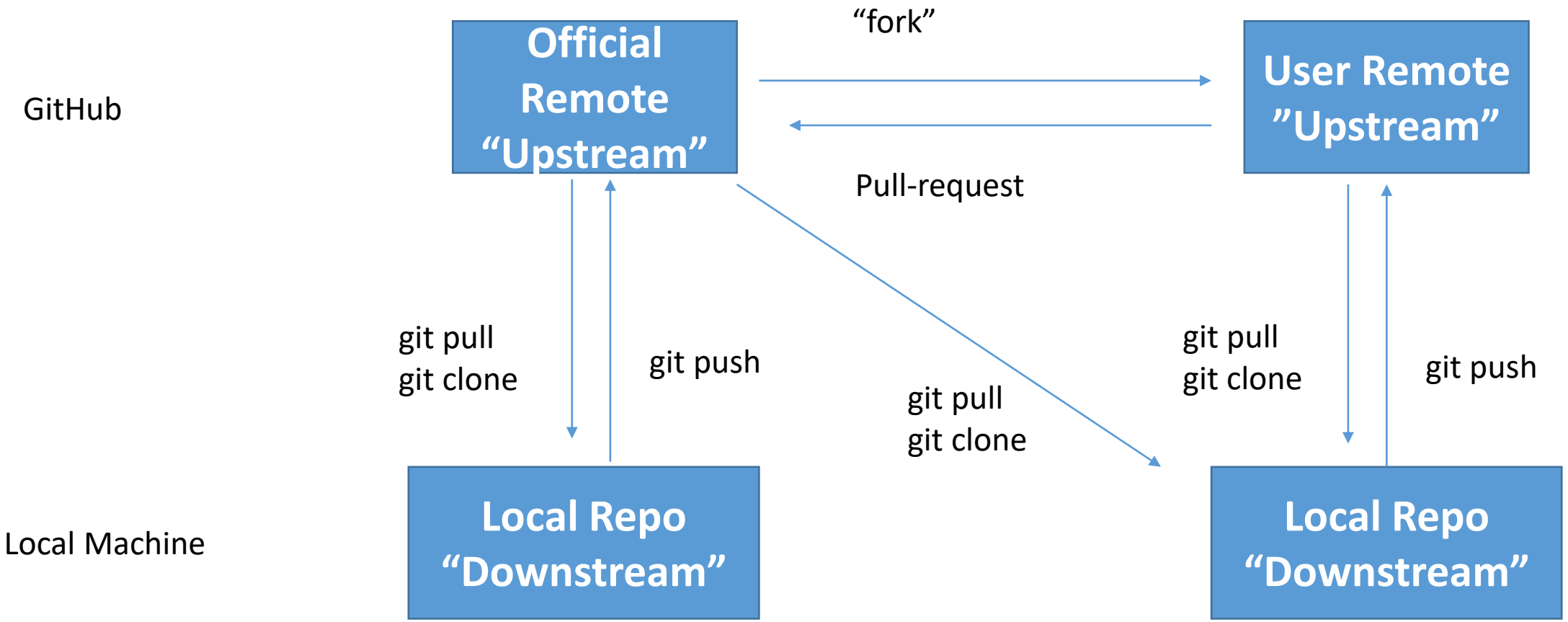
https://compchem-cybertraining.github.io/Cyber_Training_Workshop_2021/_episodes/13-projects

Now switch to Jeanette’s presentation

GitHub & Git

overview

Workflow


GitHub
Official Remote "Upstream"
"fork"
User Remote "Upstream"
Pull-request
git pull
git clone
git push
git pull
git clone
git pull
git clone
git push
Local Machine
Local Repo "Downstream"
Local Repo "Downstream"

Libra overview

Instructors

University at Buffalo
The State University of New York

Alexey Akimov

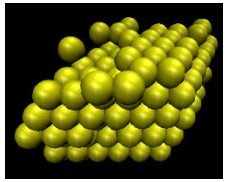
Mohammad Shakiba

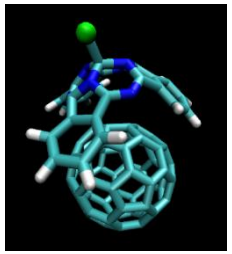
Please Introduce Yourself

- Name, position, affiliation, research group
- Research interests and expertise
- Anything else you would like to share with us

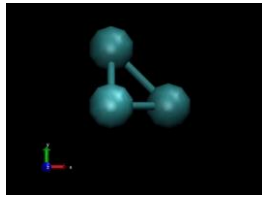
Libra History

Classical MD





Rigid body MD

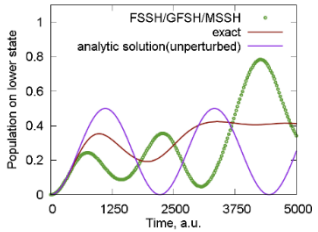


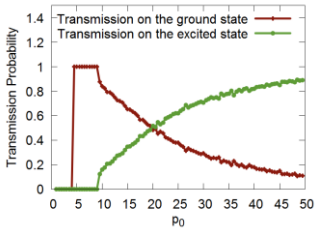
- Simplectic integrators for classical DOFs
- Thermostats
- Barostats
- Force fields
- ANN
- Chemical object representation

2007-2011 (LCCCS)

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

Ehrenfest & TSH



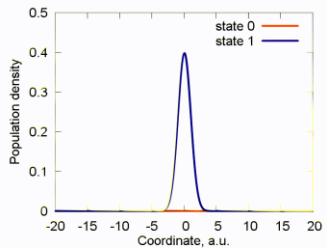


- Simplectic integrators for TDSE
- TSH, Ehrenfest, stand-alone scripts
- Decoherence methods
- Model problems
- Added HF and EHT to LCCCS
- Interface with VASP, then QE

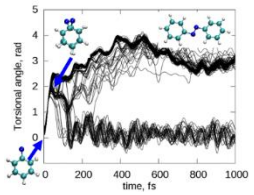
2011-2015 (Pyxaid)

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

DVR



Back-reaction



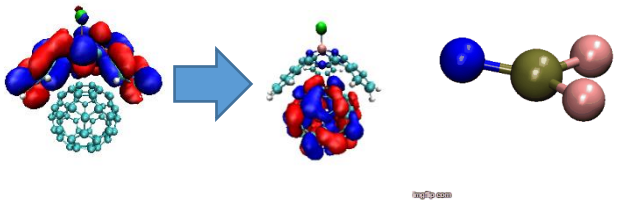
- Modularization and revision
- DVR methods
- Semiempirical Hamiltonians
- Molecular integrals
- Decoherence methods, TSH

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

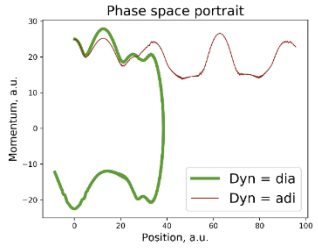
- SOC, multiple k-points, etc.

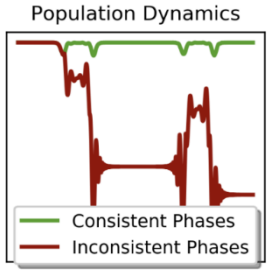
2018 (Pyxaid2, Libra-X)

Libra History

Phase correction for NACs

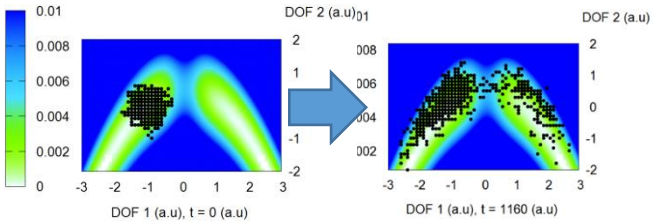
Akimov *JPCL* **2018** 9, 6096-6102





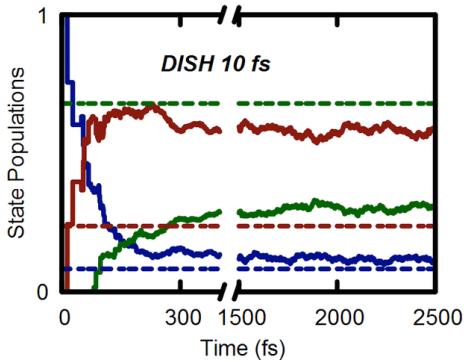
Entangled trajectories

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



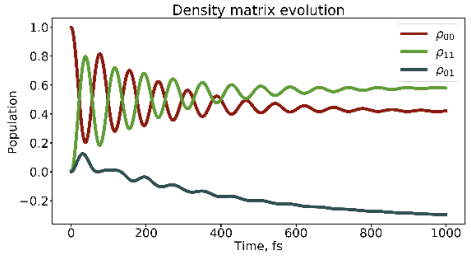
Bastida's Boltzmann-corrected Ehrenfest, mSDM

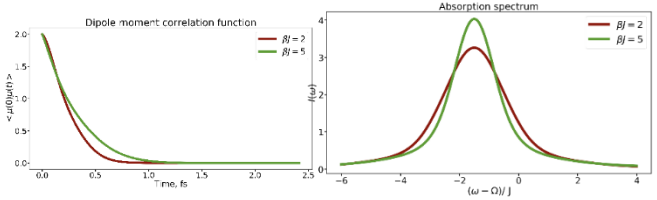
Smith; Akimov *JCP* **2019**, 151, 124107



HEOM

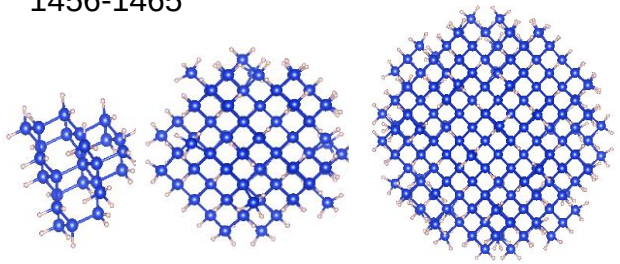
Temen, Jain, Akimov *Int. J. Quant. Chem.*, **2020**, 120, e26373





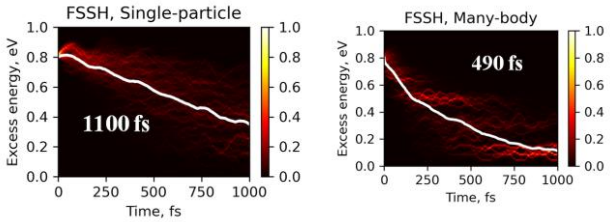
Belyaev-Lebedev LZ method

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

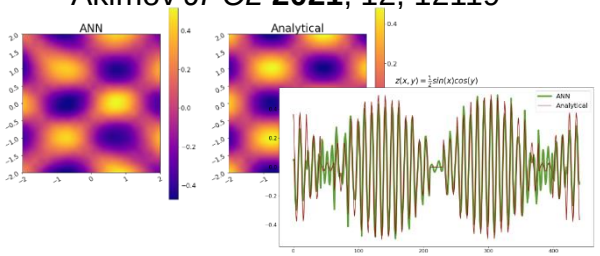


Revised DISH, new workflows

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) src/dyn/heom
https://github.com/amber-jain-group-iitb/heom_amber

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

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

Sophya Garaschchuk – quantum trajectory guided Gaussians (QTAG) in progress

Craig Martens – quantum trajectory surface hopping (QTSH) in progress

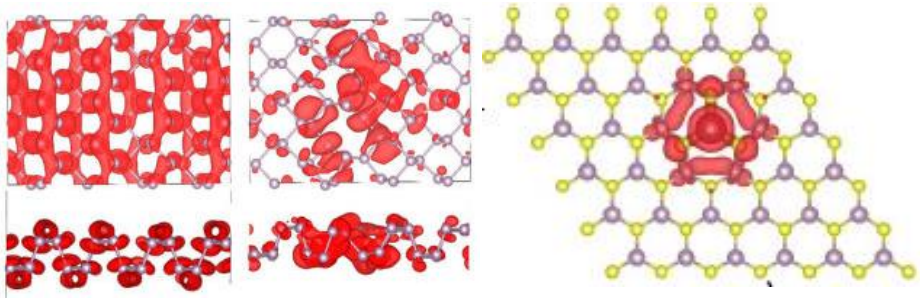
… and more

Practical: Libra in materials research

University at Buffalo
The State University of New York

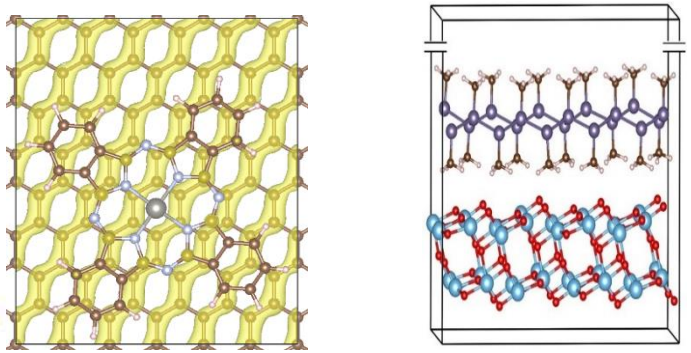
2D systems

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

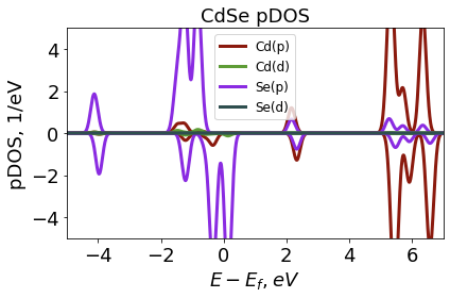


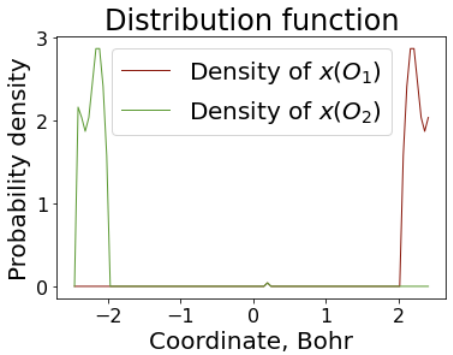
2D heterojunctions

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



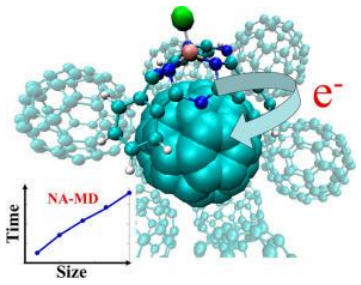
Auxiliary Analysis Tools





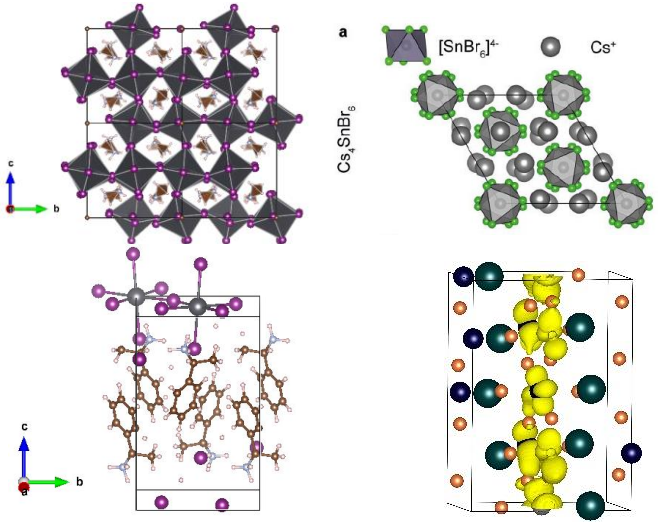
Organic heterojunctions

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



Perovskites

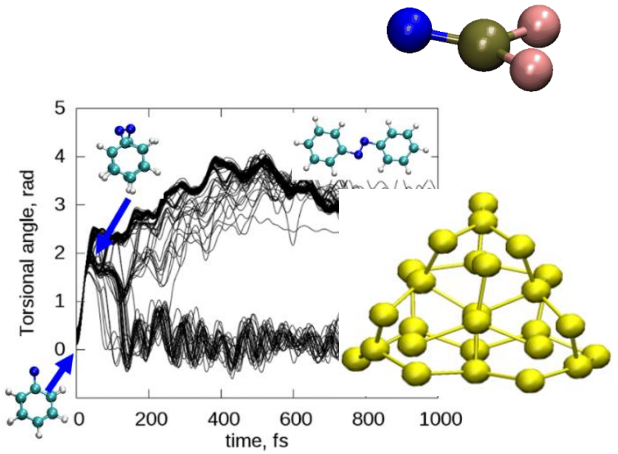
Nijamudheen, A.; AVA *JPCL* **2018**, *9*, 248

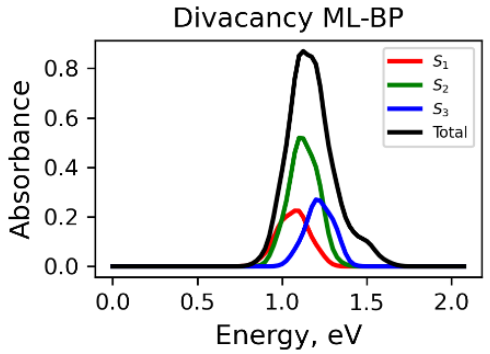


Quantum Dots & Molecules

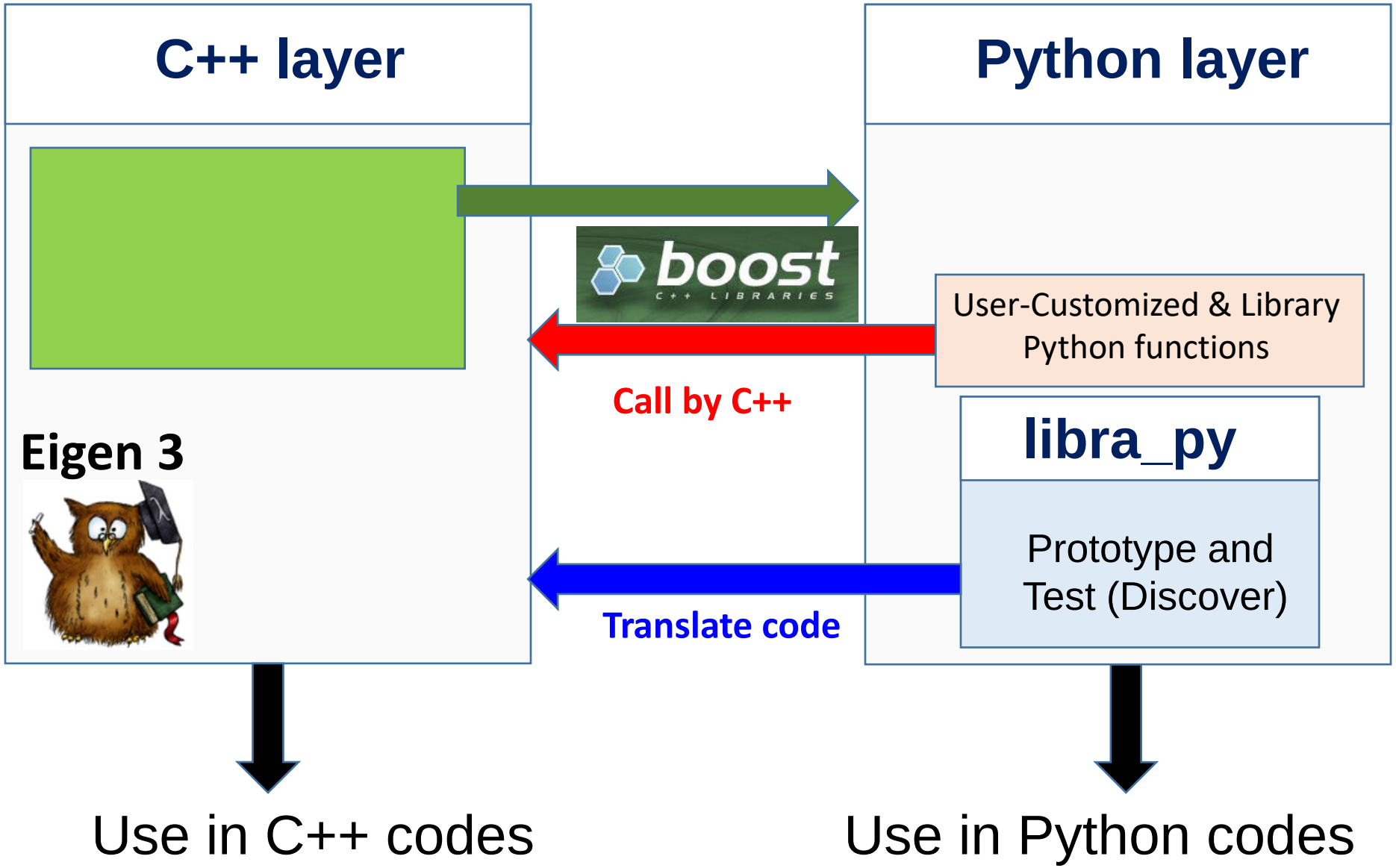
Lin, Y.; AVA *JPCA.* **2016**, 120, 9028

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





C++/Python Interoperability



Modularity: API Diversity

University at Buffalo
The State University of New York

- 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

University at Buffalo
The State University of New York

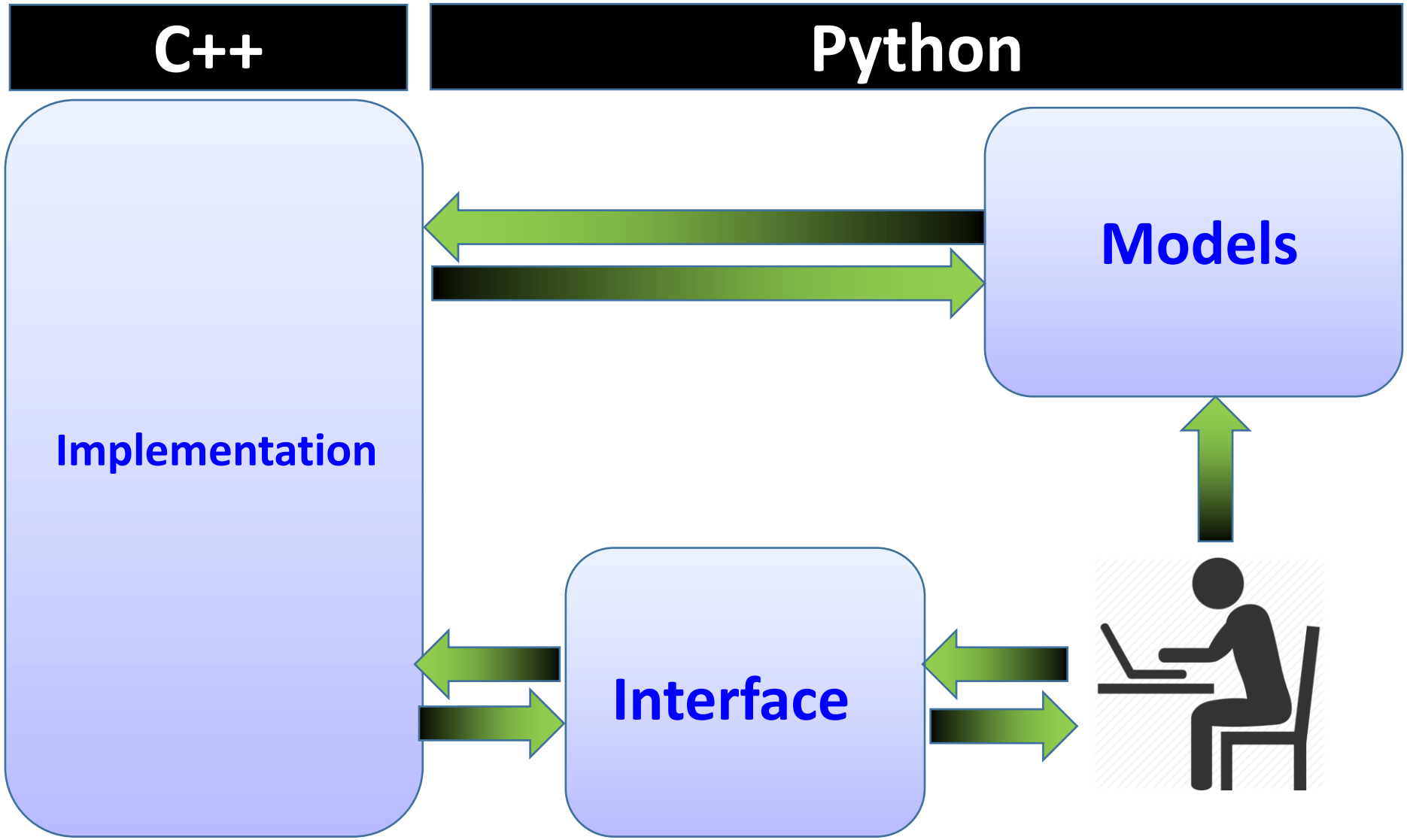
Computing kinetic energy between Gaussians

```
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



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…

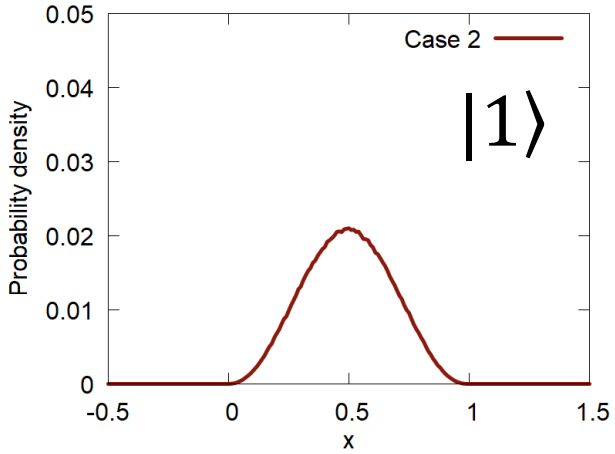
Don’t need to implement the model in C++

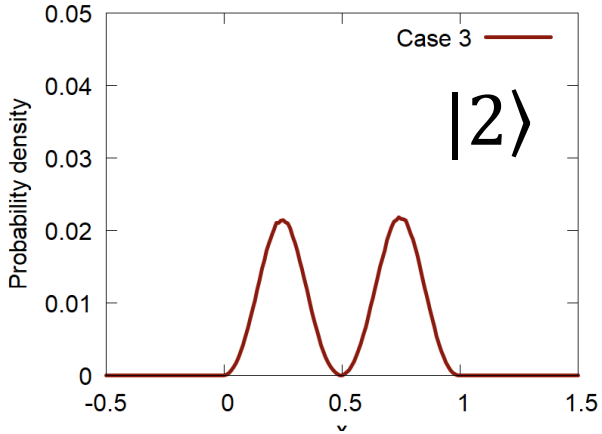
Initial conditions: Metropolis Sampling

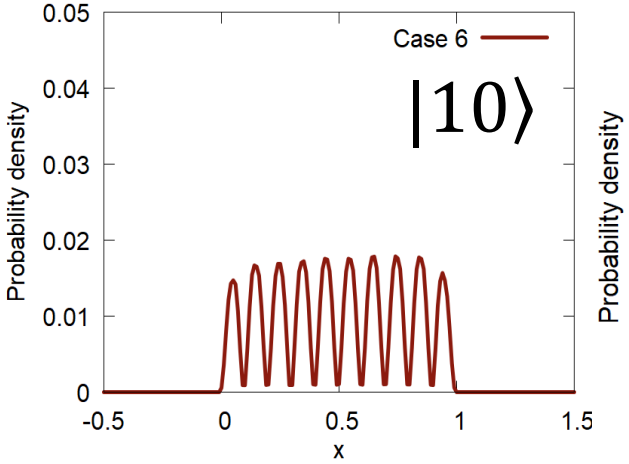
University at Buffalo
The State University of New York

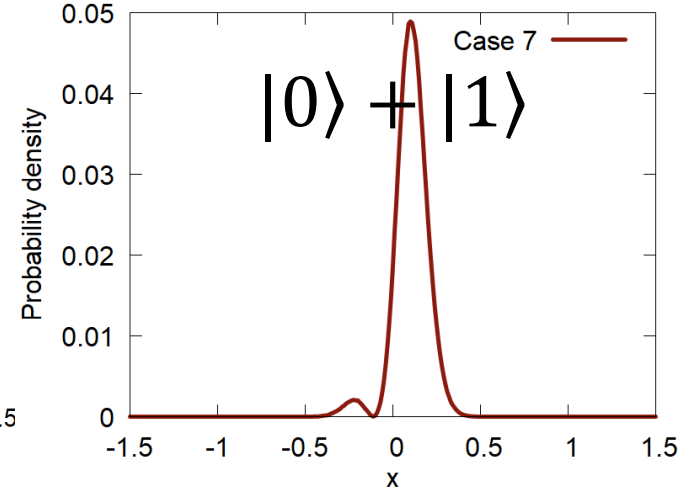
Particle in a box

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



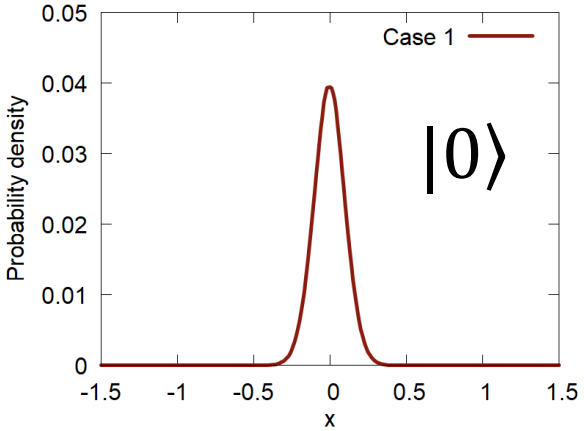


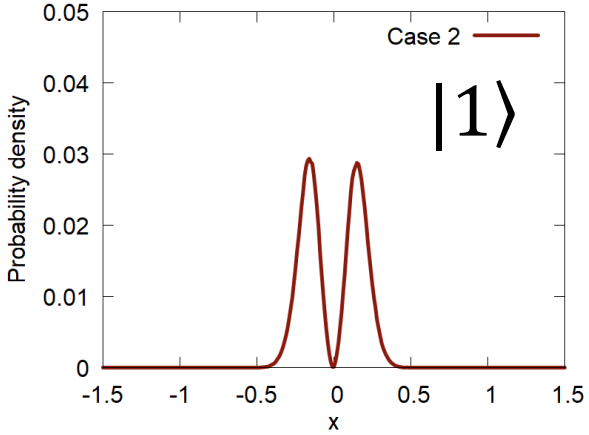


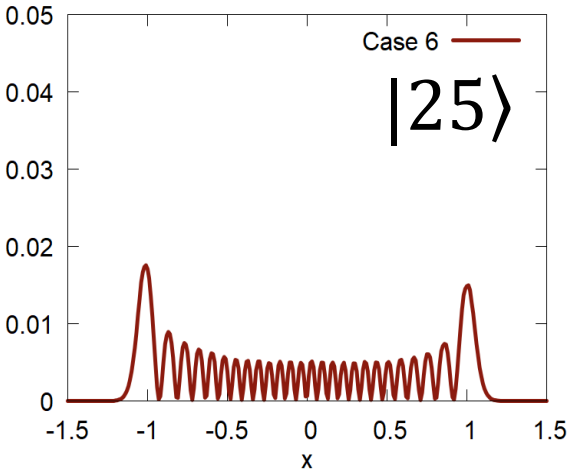


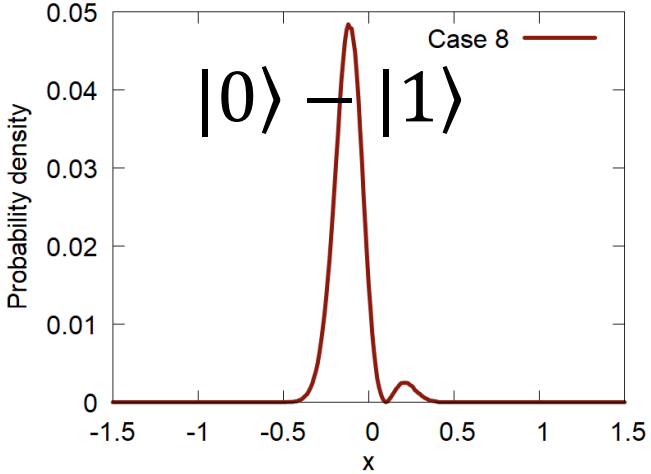
Harmonic oscillator

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









How it works with dynamics

University at Buffalo
The State University of New York

C++

Transformations

Ham_adi
d1Ham_adi
…

Ham_dia
d1Ham_dia
…

Python

User-defined Hamiltonian

Dynamical Algorithm

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

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

ham.compute_adiabatic(int lvl)

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

User calls dynamics

Output

Keep Dynamical Workflow Fixed

University at Buffalo
The State University of New York

User defines how to run the dynamical simulation

```
for i in xrange(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

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

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.

Code | Issues 1 | Pull requests 1 | Discussions | Actions | Projects | Wiki | Security | Insights

Devel #124

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

Conversation 13 | Commits 5 | Checks 0 | Files changed 9

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

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