XACS
Xiamen Atomistic
Computing Suite
XACScloud.com

MLatom.com

# Surface-hopping dynamics

Pavlo O. Dral

Xiamen University, P.R. China

Visiting Professor in
Nicolaus Copernicus University, Poland

5 July 2024

**Transitions between electronic states also often have to be taken into account, particularly, when simulating photophysical and photochemical processes**

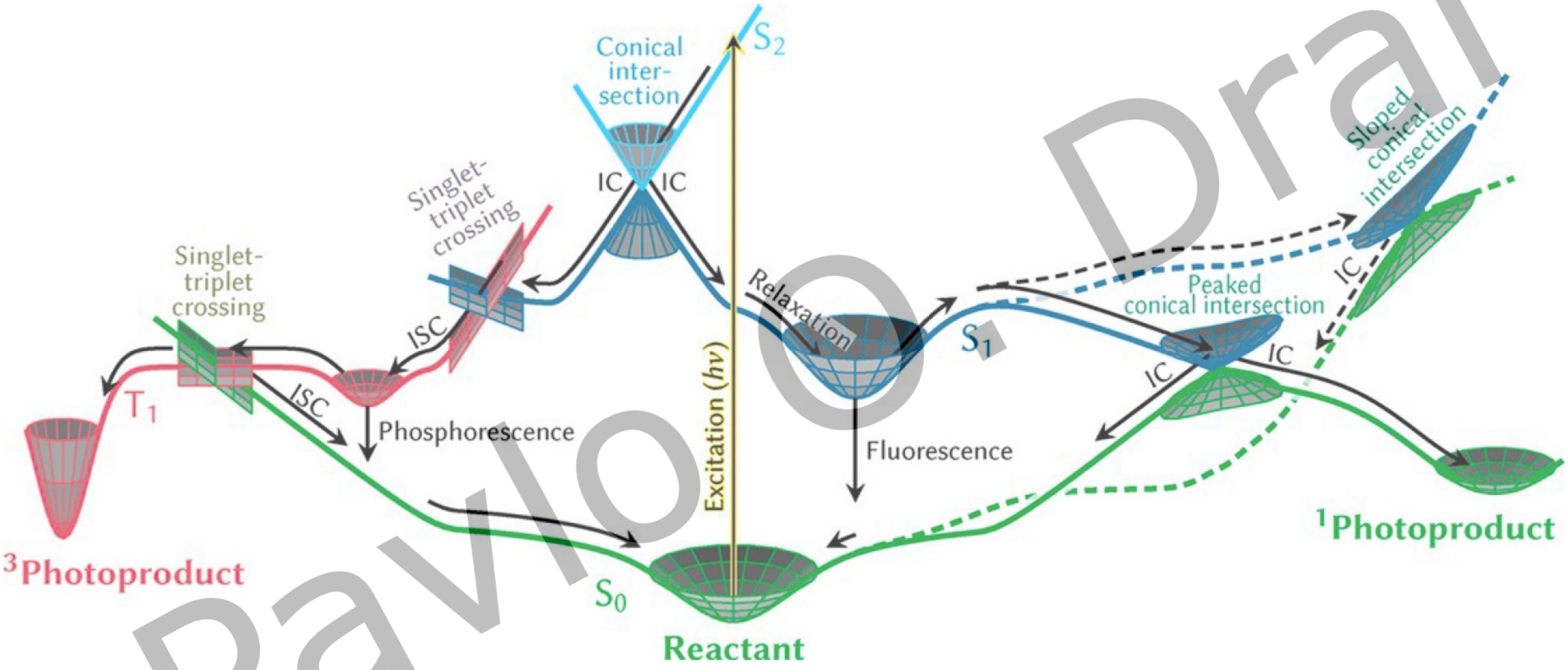
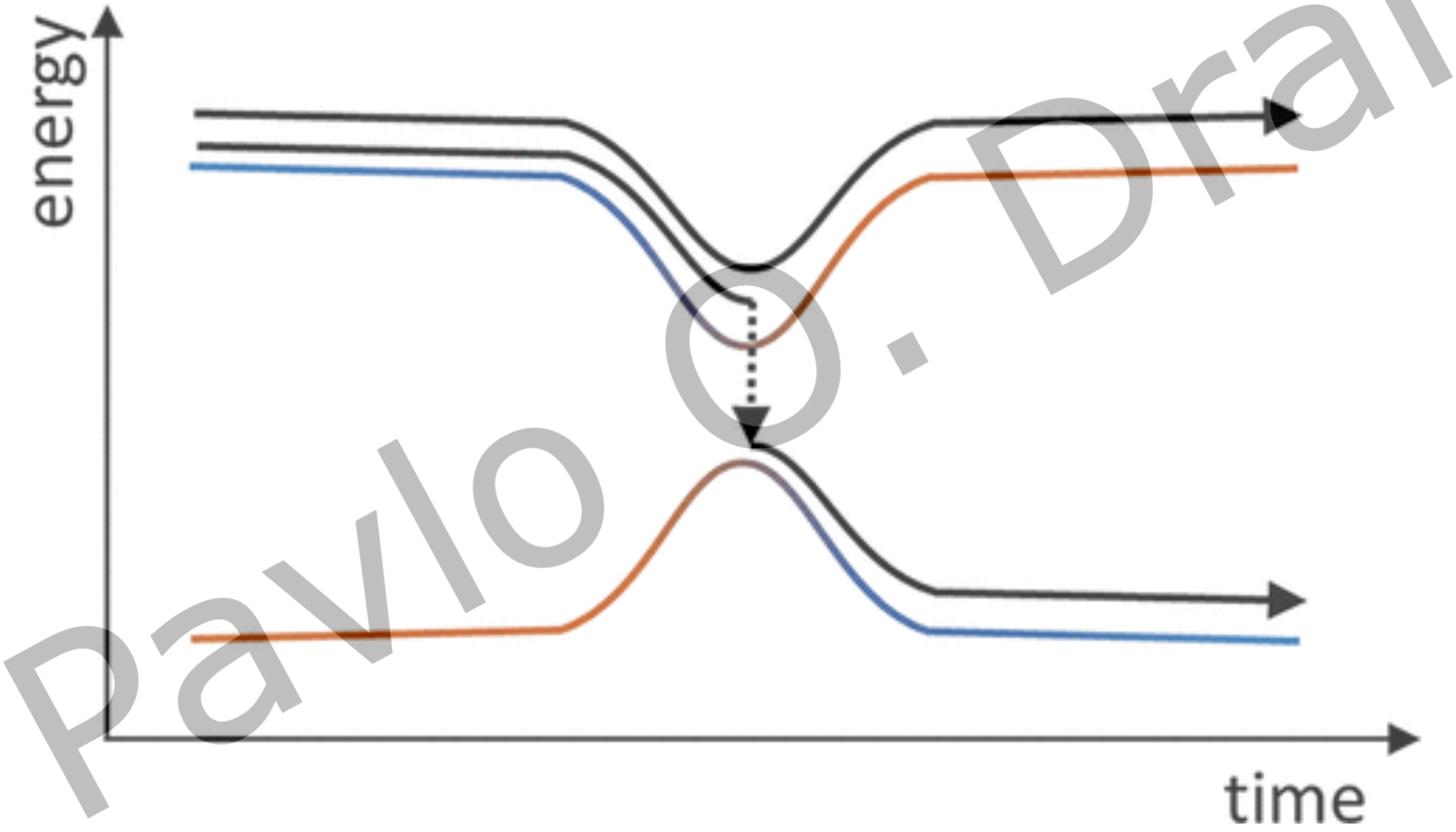


FIG. 3 A typical potential energy surface in NAMD. *Reproduced from S. Mai, L. González, Angew. Chem. Int. Ed. 59 (2020) 16832–16846, under CC BY 4.0.*

# Surface Hopping



J. C. Tully, *J. Chem. Phys.* **1990**, *93*, 1061
R. Crespo-Otero and M. Barbatti, *Chem. Rev.* **2018**, *118*, 7026

# Cost of Dynamics

Typical nonadiabatic excited-state simulations:

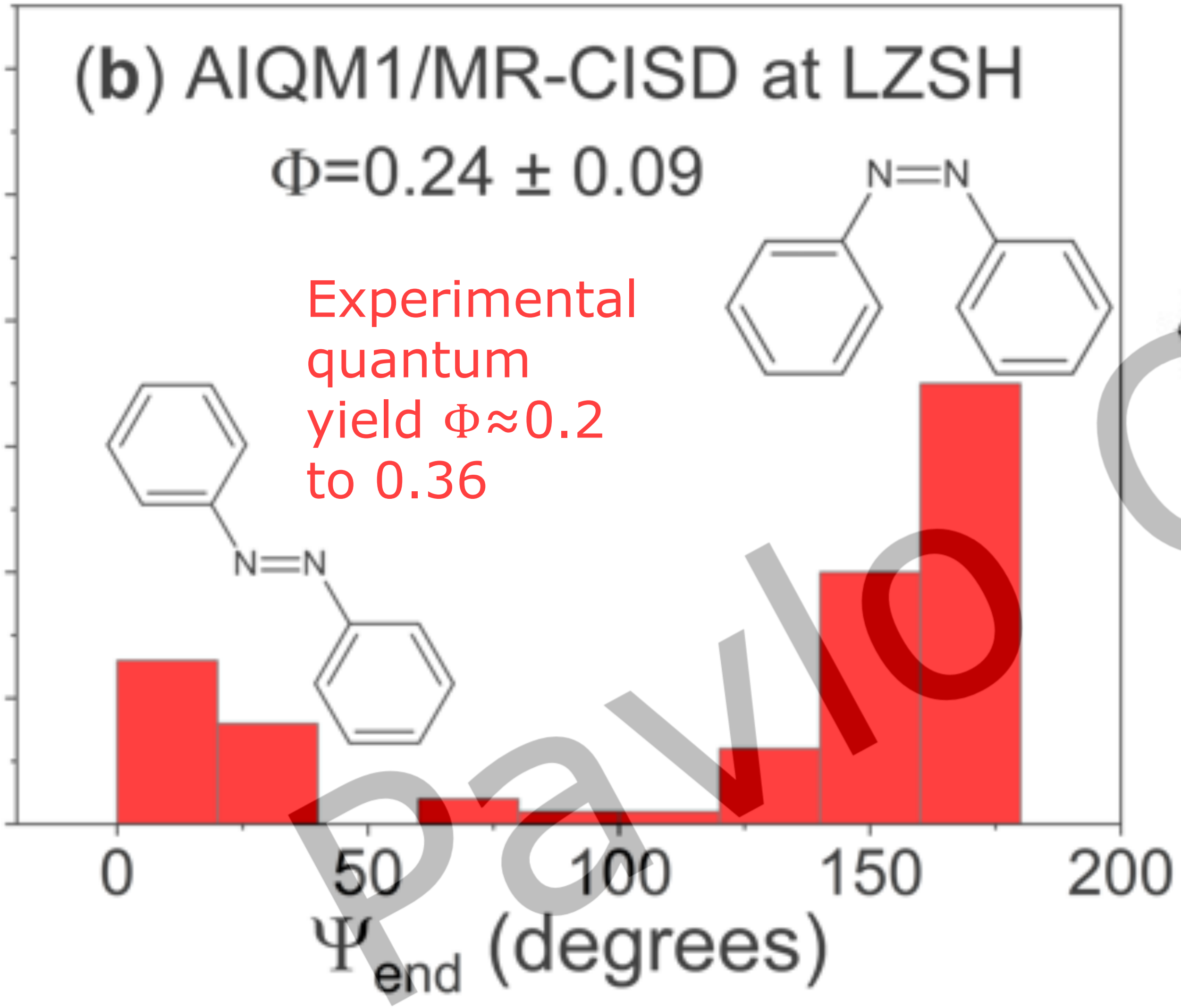
- 100 trajectories
- for 1 ps = 1000 fs
- 0.5 fs time step

Number of QM calculations:

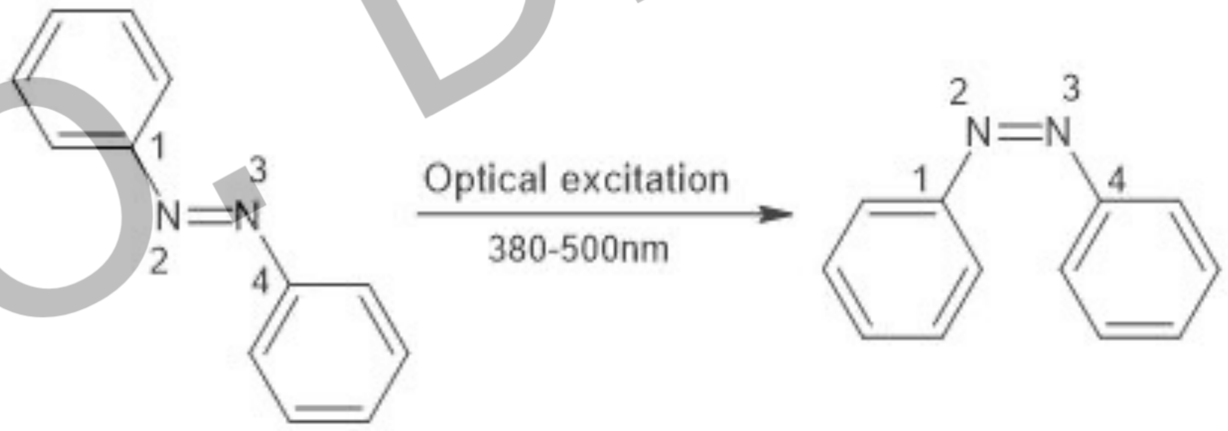
- 2000 per trajectory
- 200 000 in total

P. O. Dral, M. Barbatti, W. Thiel, *J. Phys. Chem. Lett.* **2018**, *9*, 5660

# AIQM1 surface-hopping dynamics

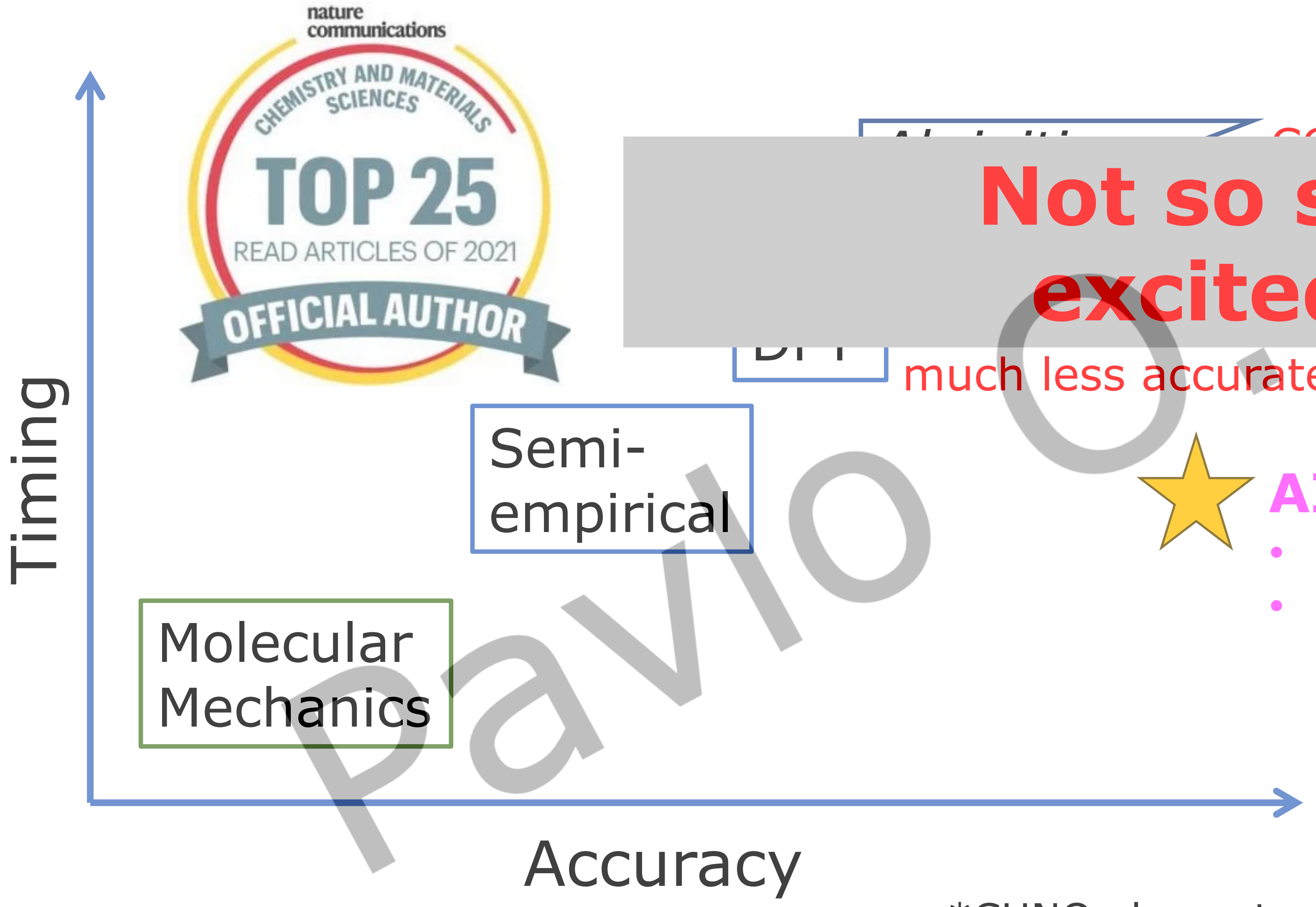


Experimental quantum yield Φ≈0.2 to 0.36



L. Zhang, S. V. Pios, M. Martyka, F. Ge, Y.-F. Hou, Y. Chen, L. Chen, J. Jankowska, M. Barbatti, P. O. Dral. Preprint on *arXiv*: https://arxiv.org/abs/2404.06189

# Quantum chemistry approximations



**Not so simple for excited states!**

Do not use B3LYP if you can use AIQM1!

**AIQM1***

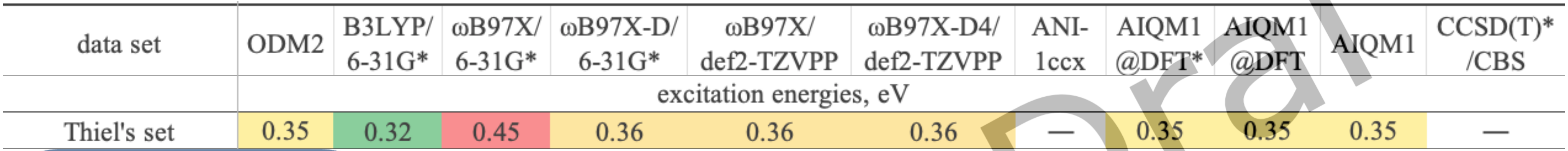
- **Accuracy of CCSD(T)**
- **Orders of magnitude faster than B3LYP**

AIQM1: P. Zheng, R. Zubatyuk, W. Wu, O. Isayev, P. O. Dral, *Nat. Commun.* **2021**, *12*, 7022

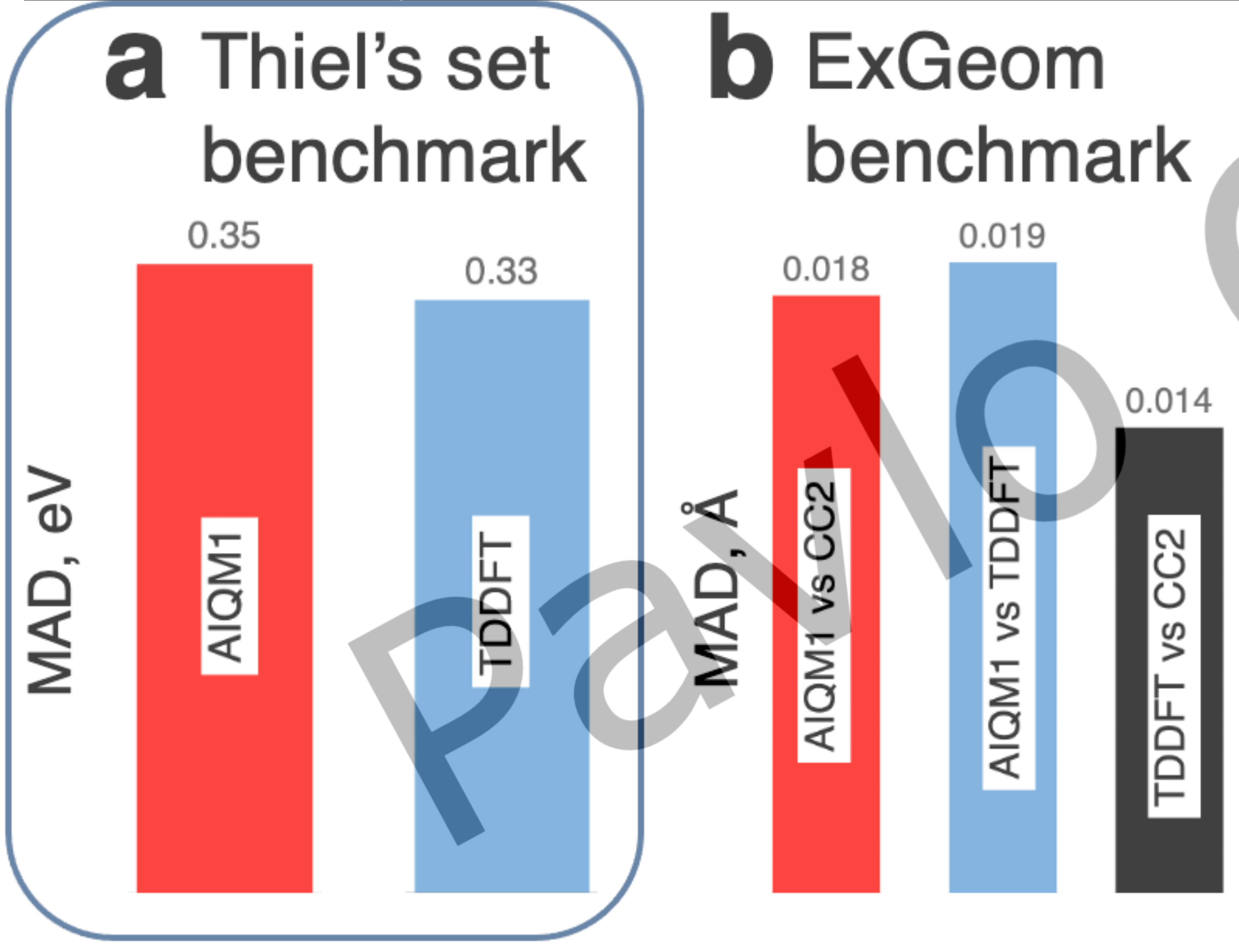
*CHNO elements only – extensions on the way

P. O. Dral, *J. Phys. Chem. Lett.* **2020**, *11*, 2336

# Excited states with AIQM1/MRCI

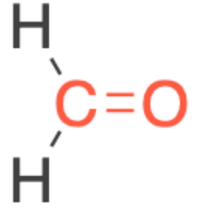


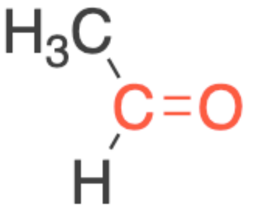
| data set | ODM2 | B3LYP/<br>6-31G* | ωB97X/<br>6-31G* | ωB97X-D/<br>6-31G* | ωB97X/<br>def2-TZVPP | ωB97X-D4/<br>def2-TZVPP | ANI-<br>1ccx | AIQM1<br>@DFT* | AIQM1<br>@DFT | AIQM1 | CCSD(T)*<br>/CBS |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | excitation energies, eV | | | | | | | | | | |
| Thiel's set | 0.35 | 0.32 | 0.45 | 0.36 | 0.36 | 0.36 | — | 0.35 | 0.35 | 0.35 | — |



TDDFT:
Linear-response TD B3LYP/TZVP

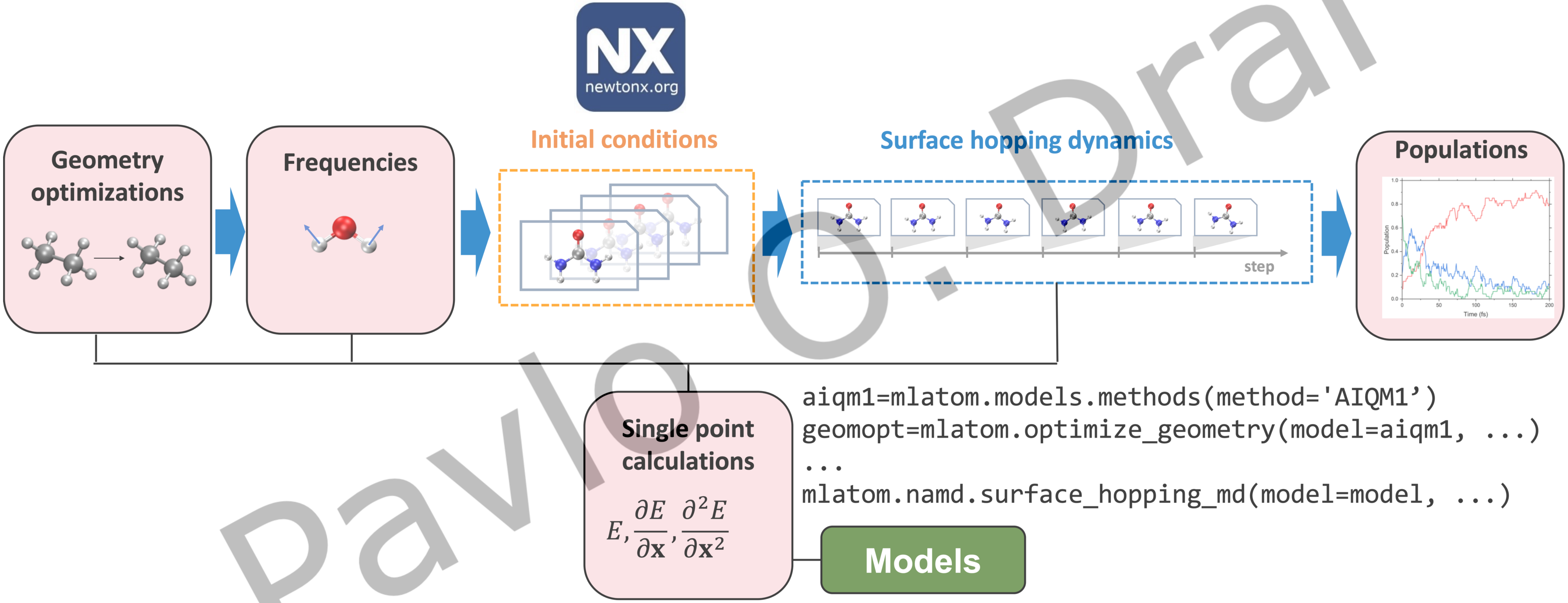
**c** bond length
Å





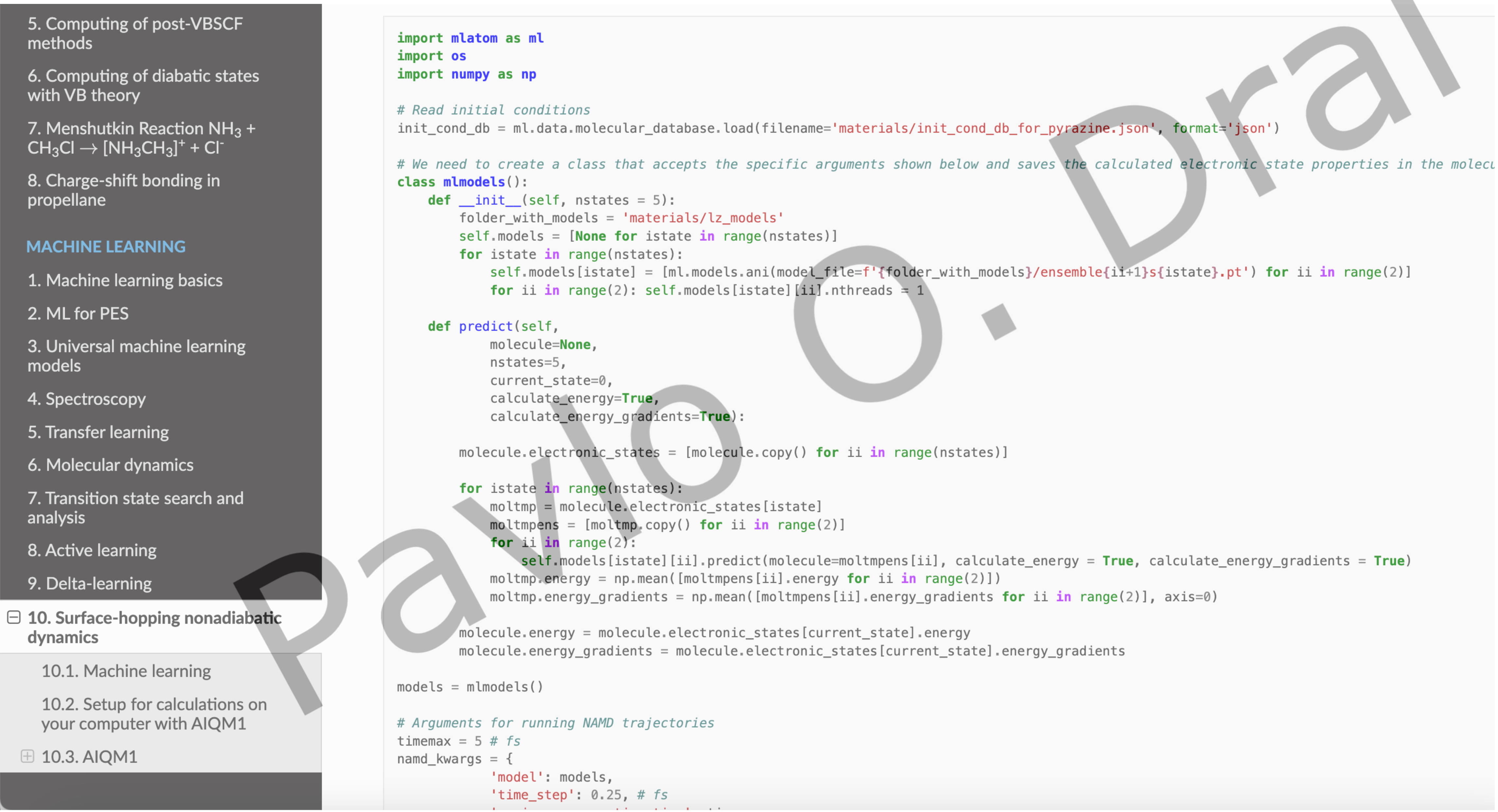
| | 1nπ* | 3nπ* | 3ππ* | 1nπ* |
|---|---|---|---|---|
| exp. | 1.323 | 1.307 | 1.423 | 1.320 |
| AIQM1 | 1.339 | 1.304 | 1.608 | 1.342 |
| TDDFT | 1.296 | 1.298 | – | 1.304 |
| CC2 | 1.361 | 1.343 | 1.469 | 1.387 |

P. Zheng, R. Zubatyuk, W. Wu, O. Isayev, P. O. Dral, *Nat. Commun.* **2021**, *12*, 7022



```
aiqm1=mlatom.models.methods(method='AIQM1’)
geomopt=mlatom.optimize_geometry(model=aiqm1, ...)
...
mlatom.namd.surface_hopping_md(model=model, ...)
```

L. Zhang, S. V. Pios, M. Martyka, F. Ge, Y.-F. Hou, Y. Chen, L. Chen, J. Jankowska, M. Barbatti, P. O. Dral. Preprint on *arXiv*: https://arxiv.org/abs/2404.06189



```python
import mlatom as ml
import os
import numpy as np

# Read initial conditions
init_cond_db = ml.data.molecular_database.load(filename='materials/init_cond_db_for_pyrazine.json', format='json')

# We need to create a class that accepts the specific arguments shown below and saves the calculated electronic state properties in the molecu
class mlmodels():
    def __init__(self, nstates = 5):
        folder_with_models = 'materials/lz_models'
        self.models = [None for istate in range(nstates)]
        for istate in range(nstates):
            self.models[istate] = [ml.models.ani(model_file=f'{folder_with_models}/ensemble{ii+1}s{istate}.pt') for ii in range(2)]
            for ii in range(2): self.models[istate][ii].nthreads = 1

    def predict(self,
            molecule=None,
            nstates=5,
            current_state=0,
            calculate_energy=True,
            calculate_energy_gradients=True):

        molecule.electronic_states = [molecule.copy() for ii in range(nstates)]

        for istate in range(nstates):
            moltmp = molecule.electronic_states[istate]
            moltmpens = [moltmp.copy() for ii in range(2)]
            for ii in range(2):
                self.models[istate][ii].predict(molecule=moltmpens[ii], calculate_energy = True, calculate_energy_gradients = True)
            moltmp.energy = np.mean([moltmpens[ii].energy for ii in range(2)])
            moltmp.energy_gradients = np.mean([moltmpens[ii].energy_gradients for ii in range(2)], axis=0)

        molecule.energy = molecule.electronic_states[current_state].energy
        molecule.energy_gradients = molecule.electronic_states[current_state].energy_gradients

models = mlmodels()

# Arguments for running NAMD trajectories
timemax = 5 # fs
namd_kwargs = {
            'model': models,
            'time_step': 0.25, # fs
```