Python Rapid Artificial Intelligence Ab Initio Molecular Dynamics

User Manual

Version 2.2a

June 8, 2022

Jingbai Li
Hoffmann Institute of Advanced Materials
Shenzhen Polytechnics, China

Steven A. Lopez
Northeastern University, Boston, U.S.A

Special acknowledgment to:

Pascal Friederich	Karlsruhe Institute of Technology, Germany
Patrick Reiser	Karlsruhe Institute of Technology, Germany

Citation:

J. Li, P. Reiser, B. R. Boswell, A. Eberhard, N. Z. Burns, P. Friederich, and S. A. Lopez, "Automatic Discovery of Photoisomerization Mechanisms with Nanosecond Machine Learning Photodynamics Simulations", *Chem. Sci.* **2021**.

Contents

1. What is PyRAI^{2}MD

Python Rapid Artificial Intelligence Ab Initio Molecular Dynamics (PyRAI^{2}MD) is a suite of Python scripts for nonadiabatic molecular dynamics simulation using machine-learning (ML) potentials. The primary aim of this project is to leverage the present nonadiabatic molecular dynamics (NAMD) techniques enabling nanosecond-scale simulations for medium-size molecular systems at high-level quantum chemical methods e.g., complete active space self-consistent field (CASSCF) with extended multistate second-order perturbative corrections (XMS-CASPT2).

PyRAI^{2}MD is designed as a user-friendly platform that integrate the trajectory surface hopping algorithms, and the state-of-the-art Neural Networks (NNs) models. PyRAI^{2}MD aims to simplify the job preparation procedures for newcomers of ML and NAMD.

PyRAI^{2}MD integrates a NAMD kernel and an ML kernel via an internal communication in memory. In turn, new features in NAMD simulations and ML models can be developed simultaneously.

2. Features

2.1. Nonadiabatic molecular dynamics

NVE, NVT, center of mass velocity removal, excessive kinetic energy
FSSH, ZNSH, NOSH

2.2. Machine-learning models

NNs
Model selection

2.3. External quantum chemical programs

Molcas
Local, slurm, customized basis set

BAGEL
Local, slurm

ORCA
In the future

GAMESS
In the future

GFN-xTB
In the future

MNDO
In the future

3. Installation

PyRAI^{2}MD is tested on Python 3.7.4. The installation of Python 3.7.4 can be found at

https://www.python.org/downloads/release/python-374/.

Using a Conda environment should work for PyRAI^{2}MD but haven't tested yet.

PyRAI^{2}MD depends on several Python3 packages. You can use pip to install them.

```
python3 -m pip install numpy              #Numpy - scientific computing library
python3 -m pip install scipy              #Scipy - scientific computing library
python3 -m pip install matplotlib         #Matplotlib - scientific plot library
python3 -m pip install tensorflow==2.3    #TensorFlow2 - machine-learning library
python3 -m pip install cython             #Cython - C-extension for python
```

To install PyRAI2MD, first download the codes

```
git clone https://github.com/lopez-lab/PyRAI2MD.git
```

Specify environment variable of PyRAI^{2}MD, where you find the main script pyrai2md.py

```
export PYRAI2MD=/path/to/PyRAI2MD
```

Compile the fssh library with Cython

```
cd $PYRAI2MD/Tools
bash compile_cython_fssh.sh
```

Test installation

```
python3 $PYRAI2MD/pyrai2md.py
```

Test functionalities, see X for more information of test calculations

```
cp $PYRAI2MD/Tools/test_PyRAI2MD.sh .
bash test_PyRAI2MD.sh
```

4. Getting started with PyRAI^{2}MD

4.1. Input structure

PyRAI^{2}MD reads a plain text file and does not require a specific extension. An input file looks like below:

```
&CONTROL
title      test
jobtype    train

&NN
train_data    data.json
```

The content is case insensitive, but each keyword (*blue*) must take one to read the input value (*red*) properly. The '&' defines a keyword section (*black*) and the empty line will be automatically skipped. Current available keyword sections include:

CONTROL	This section reads general information to set up calculations. It also controls the parameters used in adaptive sampling for the neural network active learning.
MOLECULE	This section reads molecular specifications including configuration interaction space, spin multiplicities. It also defines the interstate couplings, multiscale regions, periodic conditions, and external constrains.
MOLCAS	This section reads environment variables for setting up Molcas calculations.
BAGEL	This section reads environment variables for setting up BAGEL calculations.
MD	This section reads (nonadiabatic) molecular dynamics parameters. It controls the cutoff of the trajectories for the neural network active learning.
NN	This section reads the model information of neural networks.
SEARCH	This section reads the parameters used in grid search for optimizing neural network hyperparameters.
EG	This section reads the hyperparameters for energy+gradient model.
NAC	This section reads the hyperparameters for nonadiabatic coupling model.

SOC	This section reads the hyperparameters for spin-orbit coupling model**.**
EG2	This section reads the hyperparameters for the second energy+gradient model.
NAC2	This section reads the hyperparameters for the second nonadiabatic coupling model.
SOC2	This section reads the hyperparameters for the second spin-orbit coupling model**.**
FILE	This section reads molecular information to use PyRAI^{2}MD tool for training data extraction**.**

4.2. Run PyRAI^{2}MD

Before running PyRAI^{2}MD, remember to set the environment variables.

```
export PYRAI2MD=/path/to/PYRAI2MD
```

To run PyRAI^{2}MD:

```
python3 $PYRAI2MD/pyrai2md.py input
```

5. Keyword sections

5.1. CONTROL

The keywords, default values, and short descriptions are listed below.

$CONTROL		
title	None	name for the output, user defined
ml_ncpu	1	number of cpu used for ml jobs
qc_ncpu	1	number of cpu used for qc jobs
gl_seed	1	random number seed
jobtype	sp	type of PyRAI^{2}MD job
qm	nn	neural networks as the electronic property calculator
----------------------- *keywords below are used for adaptive sampling* -----------------------		
abinit	molcas	molcas as the ab initio calculator
load	1	load existing model for adaptive sampling
pop_step	200	save average population for the first 200 steps
refine	0	refine data collected near the surface hopping structures, the default value skips this procedure
refine_num	4	number of data collected near the surface hopping structures for refinement
refine_end	200	the last MD step to stop the data refinement near surface hopping structures, the default value searches the surface hopping in the first 200 steps
maxiter	1	maximum number of iterations in the adaptive sampling
maxsample	1	Maximum number of sampled structures per trajectory
dynsample	0	use dynamically weighted thresholds, the default value uses constant thresholds to sample structures
maxdiscard	0	maximum discarded snapshots before adjusting thresholds
maxenergy	0.05	maximum energy threshold to stop trajectories, the unit is Hartree
minenergy	0.02	minimum energy threshold to record snapshots of a trajectory

dynenergy	0.1	weights to increase or decrease the current energy threshold according to the distance between the minimum and maximum energy threshold
inienergy	0.3	initial value of the maximum energy threshold
fwdenergy	1	number of iterations delayed before increasing the current energy threshold
bckenergy	1	number of iterations delayed before decreasing the current energy threshold
maxgrad	0.15	maximum gradient threshold to stop trajectories, the unit is Hartree·Bohr^{-1}
mingrad	0.06	minimum gradient threshold to record snapshots of a trajectory
dyngrad	0.1	weights to increase or decrease the current gradient threshold according to the distance between the minimum and maximum gradient threshold
inigrad	0.3	initial value of the maximum gradient threshold
fwdgrad	1	number of iterations delayed before increasing the current gradient threshold
bckgrad	1	number of iterations delayed before decreasing the current gradient threshold
maxnac	0.15	maximum nac threshold to stop trajectories, the unit is Bohr^{-1}
minnac	0.06	minimum nac threshold to record snapshots of a trajectory
dynnac	0.1	weights to increase or decrease the current nac threshold according to the distance between the minimum and maximum nac threshold
ininac	0.3	initial value of the maximum nac threshold
fwdnac	1	number of iterations delayed before increasing the current nac threshold
bcknac	1	number of iterations delayed before decreasing the current nac threshold
maxsoc	50	maximum soc threshold to stop trajectories, the unit is cm^{-1}
minsoc	20	minimum soc threshold to record snapshots of a trajectory
dynsoc	0.1	weights to increase or decrease the current soc threshold according to the distance between the minimum and maximum soc threshold
inisoc	0.3	initial value of the maximum soc threshold

fwdsoc	1	number of iterations delayed before increasing the soc energy threshold
bcksoc	1	number of iterations delayed before decreasing the soc energy threshold

Full descriptions for all available keywords are summarized below.

title	sets the name of the calculation, all temporary and logfiles will be named according to this value.
ml_ncpu	sets the number of cpu that will be used to run ML-related jobs using python multiprocessing. ML-related **jobtype** are train, adaptive, search.
qc_ncpu	sets the number of cpu that will be used to run QC-related jobs using python multiprocessing. QC-related **jobtype** is adaptive.
gl_seed	sets the global seed for random number generator. It affects the reproducibility of the surface hopping calculations during NAMD and adaptive sampling.

jobtype sets the type of PyRAI^{2}MD job. Available options are:

sp	single-point calculations,
md	NAMD simulation,
hop	surface hopping calculation,
adaptive	adaptive sampling,
train	training NNs,
prediction	predicting electronic properties using trained NNs,
search	NN hyperparameter optimization with grid search.

qm chooses the electronic property calculator. Available options are:

nn	uses trained NNs,
molcas	uses OpenMolcas for CASSCF calculations,
mlctkr	uses OpenMolcas/Tinker for QM/MM calculations,
bagel	uses BAGEL, for CASSCF and XMS-CASPT2 calculations

abinit	chooses the reference QC electronic property calculator. Available options are the same as **qm** except for nn. The chosen program will be used to recompute the QC-data for the collected structures during adaptive sampling.
load	reads a pretrained NNs for adaptive sampling. When it is set to 0, it will first training NNs before running the adaptive sampling.

pop_step	sets the number of MD steps to compute the average population over all trajectories propagated during adaptive sampling. Note that the step size depends on both the timestep and checkpointing frequency, which can be specified by size in &**MD** section.
refine	turns on additional structural sampling around the surface hopping points during adaptive sampling. It is turned off in default.
refine_num	sets the number of structures that will be collected around the surface hopping points during adaptive sampling.
refine_end	sets the last MD step to sample the structures if a surface hopping point is detected. Later hopping points will not be included to sample new structures. Note that the adaptive sampling only records the last a few MD steps to reduce the memory usage. Therefore, the sampling start from the recorded structures, which is not necessary to be the first MD step. The number of recorded MD steps can be adjusted by record in &**MD** section.
maxiter	sets the maximum number of iterations for adaptive sampling. The adaptive sampling will stop when it reach the maximum value or no longer find new structures.
maxsample	set the number of structures to be collected during the adaptive sampling. Note that this number does not include the number of structure refinement from **refine_num**.
dynsample	turns on the dynamically weighted adaptive sampling. The threshold values will be dynamically adjusted according to the numerical distance between the minimum and the maximum value. It is turn off in default.
maxdiscard	set the maximum number of discard structures in a trajectory. A structure will be discarded if it contains a non-physical bond length shorter than the sum of the van der Waals radius of each atom multiplied by 0.7. When the number of discarded structures exceed **maxdiscard**, the current threshold will be decreased to limit the exploration region of adaptive sampling. Otherwise, the current threshold will be increased to expand the exploration region of adaptive sampling. Note that the threshold adjustment can be delayed by fwd* and bck* keywords for the forward and backward direction.
maxenergy	sets the maximum value of the energy threshold to stop a trajectory.
minenergy	sets the minimum value of the energy threshold to record a trajectory.

dynenergy	sets the weights of the to increase or decrease the current energy threshold according to the distance between the minimum and maximum energy threshold. The adjustment is weights * (max - min) but the adjusted values will not exceed the minimum or maximum values.
inienergy	set the initial value of the energy threshold to be dynamically adjusted.
fwdenergy	set the number of delayed iterations to increase the current threshold.
bckenergy	set the number of delayed iterations to decrease the current threshold.
maxgrad	sets the maximum value of the gradient threshold to stop a trajectory.
mingrad	sets the minimum value of the gradient threshold to record a trajectory.
dyngrad	sets the weights of the to increase or decrease the current gradient threshold according to the distance between the minimum and maximum gradient threshold. The adjustment is weights * (max - min) but the adjusted values will not exceed the minimum or maximum values.
inigrad	set the initial value of the gradient threshold to be dynamically adjusted.
fwdgrad	set the number of delayed iterations to increase the current threshold.
bckgrad	set the number of delayed iterations to decrease the current threshold.
maxnac	sets the maximum value of the nac threshold to stop a trajectory.
minnac	sets the minimum value of the nac threshold to record a trajectory.
dynnac	sets the weights of the to increase or decrease the current nac threshold according to the distance between the minimum and maximum nac threshold. The adjustment is weights * (max - min) but the adjusted values will not exceed the minimum or maximum values.
ininac	set the initial value of the nac threshold to be dynamically adjusted.
fwdnac	set the number of delayed iterations to increase the current threshold.
bcknac	set the number of delayed iterations to decrease the current threshold.
maxsoc	sets the maximum value of the soc threshold to stop a trajectory.
minsoc	sets the minimum value of the soc threshold to record a trajectory.

dynsoc	sets the weights of the to increase or decrease the current soc threshold according to the distance between the minimum and maximum soc threshold. The adjustment is weights * (max - min) but the adjusted values will not exceed the minimum or maximum values.
inisoc	set the initial value of the nac threshold to be dynamically adjusted.
fwdsoc	set the number of delayed iterations to increase the current threshold.
bcksoc	set the number of delayed iterations to decrease the current threshold.

5.2. MOLECULE

The keywords, default values, and short descriptions are listed below.

&MOLECULE		
ci	1	definition of the configuration interaction space for each spin state
spin	0	definition of the spin multiplicity for each spin state
coupling	None	definition of the interstate couplings

Full descriptions for all available keywords are summarized below.

ci sets configuration interaction space for each spin state, i.e., the number of states in each spin multiplicity, 2 means two states of the first spin, i.e., S0, S1. It can take multiple integers if multiple spin states are iinvolved, e.g. 2 2 means two states in spin 1 and two states in spin 2. the spin multiplicities are defined by spin.

spin sets the total spin number for each spin state, 0 is singlet, 1 is triplet. It follows the same order as ci.

coupling reads pairwise indices to define the coupling between two states. Each pair should be separated by ','. The following example,

```
ci              2 2
spin            0 1
coupling        1 2, 2 3, 2 4, 3 4
```

defines that state 1 and 2 are singlet and state 3 and 4 are triplet. It includes the nac between state 1 and 2 (singlet) and state 3 and 4 (triplet) as well as the soc between state 2 and 3 (singlet-triplet) and state 2 and 4 (singlet-triplet). The order of index pairs does not matter and the coupling of the non-defined pairs (e.g, state 1 and 4) will be treated as zero.

5.3. MOLCAS

The Molcas calculation also needs an input template and guess orbital named with .StrOrb in the current folder. See X for examples of running Molcas calculations.

The keywords, default values, and short descriptions are listed below.

&MOLCAS		
molcas	None	path to Molcas executable
molcas_nproc	1	number of cpu for OpenMP parallelization
molcas_mem	200	number of memories for calculation
molcas_print	2	logfile printing level
molcas_project	None	project name
molcas_calcdir	$PWD	path to the temporary calculation folder
molcas_workdir	None	path to Molcas scratch folder
basis	2	additional basis set information
omp_num_threads	1	number of threads for OpenMP parallelization
use_hpc	0	submit calculation to remote cluster
keep_tmp	1	keep the temporary calculation folder

Full descriptions for all available keywords are summarized below.

molcas	sets the path to Molcas executable.
molcas_nproc	sets $MOLCAS_NPROC environment variable, the default value is 1.
molcas_mem	sets $MOLCAS_MEM environment variable, the default value is 2000 MB.
molcas_print	sets $MOLCAS_PRINT environment variable, the default value is 2.
molcas_project	sets $MOLCAS_PROJECT environment variable, the default value is taken from **title** in &CONTROL section
molcas_calcdir	sets the path to a temporary folder for Molcas calculation. The temporary folder will be named as tmp_MOLCAS. If no path is provided, the

	tmp_MOLCAS will be created in the current folder. Note this is the folder to run Molcas calculations, but not necessary to be the Molcas scratch folder, which is set by **molcas_workdir**.
molcas_workdir	sets $MOLCAS_WORKDIR environment variable. If no path is provided, it will be the same path as the tmp_MOLCAS folder set by **molcas_calc**. Note that Molcas is input/output intensive, the temporary files could be large and the calculation running in SLURM's /scratch could be slower than in a local disk. It is recommended to use a local folder such as /tmp or /srv/tmp. If you are not sure which folder to use, a shortcut is AUTO, which needs to be upper-case.
basis	reads atom annotation to use different basis sets if it is set to 1. It is turned off in default (2). To use different basis sets, you need to prepare a xyz file following the same atom order and annotate the atom with '_', e.g. "C_ X Y Z". The coordinates can be random. Then add the basis set in &GATEWAY in the Molcas input template, e.g. "ANO-S-MB, C_.ANO-S-VDZP", which will use ANO-S-VDZP for annotated atoms but ANO-S-MB for others.
omp_num_threads	sets OpenMP parallel threads for OpenMolcas, the default value is 1. Note that not all Molcas functions are parallelized.
use_hpc	submits the Molcas calculation to the job scheduler. It is turned off in default, thus the calculation is running as a subprocess in the current machine. For single calculation, it is recommended to run the Molcas calculation without **use_hpc** because it does not have to wait in the queue. However, if there are more Molcas calculations than available cpus or the disk space for all calculations is not enough, e.g. in adaptive sampling, it is better to use **use_hpc** to distribute the calculations to all available nodes via a job scheduler. To use this function, you need to prepare a submission script template with the same name as **title** in &CONTROL section, e.g. job_title.slurm and specify the all necessary #SBATCH variables.
keep_tmp	keep the temporary Molcas calculation folder. It is turned on in default. Set to 0 to turned off.

5.4. BAGEL

The BAGEL calculation also needs an input template and orbital archive in the present folder. See X for examples of running Molcas calculations.

The keywords, default values, and short descriptions are listed below.

&BAGEL		
bagel	None	path to BAGEL executable
bagel_nproc	1	number of cpu for BAGEL parallelization
bagel_project	200	project name
bagel_workdir	2	path to BAGEL calculation folder
bagel_archive	None	name of BAGEL orbital archive
mpi	$PWD	path to the MPI library
blas	None	path to BLAS library
lapack	0	path to LAPACK library
boost	2	path to BOOST library
mkl	1	path to MKL library
arch	0	cpu architecture
omp_num_threads	1	number of threads for OpenMP parallelization
use_mpi	0	use MPI for parallelization
use_hpc	0	submit calculation to remote cluster
keep_tmp	1	keep the temporary calculation folder

Full descriptions for all available keywords are summarized below.

bagel	sets the path to BAGEL executable.
bagel_nproc	sets the number of cpu for BAGEL calculation with OpenMP parallelization
molcas_project	sets the name of BAGEL calculation, the default value is taken from **title** in &CONTROL section
bagel_workdir	sets the path to a temporary folder. It creates a sub folder tmp_BAGEL for BAGEL calculation. BAGEL is mainly running in memory. Therefore, it does not suffer from the input/output overhead issue.

bagel_archive	sets the name of BAGEL orbital archive if the orbital archive has a different name from **title** in &CONTROL section. In default, the name is taken from **title** in &CONTROL section.
mpi	sets the path to MPI
blas	sets the path to BLAS library
lapack	sets the path to LAPACK library.
boost	sets the path to BOOST library.
mkl	sets the path to Intel MKL library.
arch	specifies the cpu architecture, the default value is intel64.
omp_num_threads	sets OpenMP parallel threads for BAGEL, the default value is 1.
use_hpc	submits the BAGEL calculation to the job scheduler. It is turned off in default, thus the calculation is running as a subprocess in the current machine. For single calculation, it is recommended to run the BAGEL calculation without **use_hpc** because it does not have to wait in the queue. However, if there are more BAGEL calculations than available cpus or the disk space for all calculations is not enough, e.g. in adaptive sampling, it is better to use **use_hpc** to distribute the calculations to all available nodes via a job scheduler. To use this function, you need to prepare a submission script template with the same name as **title** in &CONTROL section, e.g. job_title.slurm and specify the all necessary #SBATCH variables.
keep_tmp	keep the temporary BAGEL calculation folder. It is turned on in default. Set to 0 to turned off.

5.5. MD

The keywords, default values, and short descriptions are listed below.

&MD		
initcond	0	sample initial condition
excess	0	excess kinetic energy in Hartree
scale	1	scale kinetic energy by a factor
target	0	set a target kinetic energy in Hartree
graddesc	0	gradient descent mode (zero velocity)
reset	0	remove center of mass velocity
resetstep	0	center of mass velocity reset interval
ninitcond	20	number of sampled initial conditions
method	wigner	initial condition sampling method
format	molden	frequency file format
temp	300	temperature in Kelvin
step	10	number of threads for OpenMP parallelization
size	20.67	step size in the atomic unit of time
root	1	initial state
activestate	0	only compute gradients of the current state
sfhp	nosh	surface hopping algorithm
nactype	ktdc	type of nac
phasecheck	0	apply phase correction to nac
gap	0.5	energy gap threshold to compute Zhu-Nakamura surface hopping between the same spin states
gapsoc	0.5	energy gap threshold to compute Zhu-Nakamura surface hopping between the different spin states
substep	20	number of substep in wave function integration in FSSH calculation
integrate	0	accumulate the nuclear amplitude transfer in FSSH calculation ***This is only for debug purpose***
deco	0.1	energy-based decoherence correction in Hartree
adjust	1	adjust velocity at surface hopping

reflect	1	reflect velocity at frustrated hopping
maxh	10	Maximum number of allowed surface hoppings
dosoc	0	compute Zhu-Nakamura surface hopping between the different spin states
thermo	off	apply a thermostat for NVT ensemble
thermodelay	200	delay time for applying a thermostat in the ground-state
silent	1	no output prints on screen
verbose	0	logfile printing level
direct	2000	number of MD steps that will be written in output
buffer	500	number of MD steps that will be skipped in output
record	0	number of the last MD snapshots that will be recorded for adaptive sampling
checkpoint	0	checkpoint a trajectory for a given number of MD steps
restart	0	restart calculation
addstep	0	add MD steps in a restart calculation

Full descriptions for all available keywords are summarized below.

initcond	generates initial conditions from a frequency file. It is turned off in default. Thus, it reads coordinates and velocities from .xyz and .velo files. In adaptive sampling, the initial conditions are always generated from a frequency file, no matter it is set to 1 or 0.
excess	adds extra kinetic energy beyond the initial kinetic energy then scales the initial velocity isotopically. It is sometimes useful to accelerate the MD and drive the trajectory uphill. The unit is Hartree. This option is the first adjustment to the kinetic energy.
scale	scales the initial kinetic energy isotropically by a factor. It is sometimes useful to accelerate the MD and drive the trajectory uphill. This option is the second adjustment to kinetic energy.
target	sets a target kinetic energy to scale the initial velocity isotopically. It is sometimes useful to accelerate the MD and drive the trajectory uphill. This option is the last adjustment to the kinetic energy.

graddesc	propagates a trajectory following the gradient descent by setting the velocities to zero during the MD. It is turned off in default.
reset	removes translation and rotation velocity at the center of mass. It is turned off in default. It helps avoid the “flying ice” artifact, which results from the draining of vibration energy to translation and rotation energy when velocity rescaling (e.g., thermostat) is frequently used.
resetstep	sets the interval of removing translation and rotation velocity at the center of mass. It is usually recommended to reset velocity every 2000 steps with a timestep of 0.5 fs. If it is set to 0, it only reset the initial velocity. This keyword must be used together with **reset**.
ninitcond	sets the number of initial conditions in sampling. The last condition is used in MD if the value is greater than 1. In adaptive sampling, this value determines the number of trajectories to collect new structures.
method	chooses the method to do initial condition sampling. It is recommended to do Wigner sampling using wigner. The Boltzmann sampling is also available with boltzmann.
format	sets the frequency file format. It supports the Molcas’ molden file ($*xxx*.freq.molden), BAGEL frequency calculation output file (need to rename as $*xxx*.freq.bagel), ORCA frequency calculation output file (need to rename as $*xxx*.freq.orca), Gaussian frequency calculation output file and fchk file with “Freq=SaveNormalModes” (need to rename as $*xxx*.freq.log and $*xxx*.freq.fchk).
temp	sets the temperature in Kelvin for initial condition sampling and thermostat. It is not used in microcanonical ensemble (i.e., NVE).
step	sets the number of MD steps.
size	sets the step size in the atomic unit of time. 1 au = 0.02418884254 fs.
root	sets the initial state in NAMD. It should not be larger than the total number of states defined by **ci** in &MOLECULE.
activestate	only computes the gradients of current state with QC calculations. It is turned off in default. It reduces the cost of FSSH dynamics because the gradients of other states are not used. However, the gradients of all states are needed in Zhu-Nakamura surface hopping. This keyword is not used in ML-NAMD as NNs predict gradients of all states.

sfhp	chooses the surface hopping algorithm. Available options are: fssh Tully’s the fewest switches surface hopping with explicit nac, gsh Zhu-Nakamura surface hopping, nosh turn off the surface hopping calculation.
nactype	chooses the type of nac for fssh calculation. Available options are: nac nonadiabatic coupling vectors, non-weighted by the state energy gap ktdc curvature driven time-dependent coupling, which approximates nonadiabatic coupling by the first-order derivative of energy in two adjacent MD step.
phasecheck	apply phase correction to nonadiabatic coupling by the overlap of nac vectors at two adjacent MD step. It is turned off in default. It is only used when **sfhp** is set to fssh and **nactype** is set to nac.
gap	sets the energy gap threshold to compute Zhu-Nakamura surface hopping between two states with same spin multiplicity. The surface hopping calculations are skipped when the energy gap is larger than this value. This keyword is not used when **sfhp** is set to fssh.
gapsoc	sets the energy gap threshold to compute Zhu-Nakamura surface hopping between two states with different spin multiplicities. The surface hopping calculations are skipped when the energy gap is larger than this value. This keyword is not used when **sfhp** is set to fssh.
substep	sets the number of substeps to integrate the electronic wave function in fssh calculation. It is not used when **sfhp** is set to gsh.
integrate	accumulate the nuclear amplitude in fssh calculation. ***This is only used for debug purpose and must not be used to produce results for publication.***
deco	applies the energy-based decoherence correction in fssh calculation. The unit is in Hartree. It is not used when **sfhp** is set to gsh.
adjust	scales the velocity at surface hopping events. Available options are: 0 do not scale velocity, 1 scale velocity isotropically, 2 scale velocity along the NAC direction.
reflect	changes the velocity direction when frustrated hopping happens. Available options are: 1 directly reflect velocity

	2 reflect the velocity component along the NAC vectors.
maxh	sets the maximum number of allowed surface hopping events.
dosoc	computes Zhu-Nakamura surface hopping between two states with different spin multiplicities. It requires additional calculations of spin-orbit coupling and is turned off in default.
thermo	controls the ensemble of trajectory. Available options are: off do not rescale velocity (NVE) 0 rescale velocity to conserve total energy (forced to NVE ensemble) 1 rescale velocity using Nóse-Hoover thermostat (NVT ensemble) 2 rescale velocity to conserve total energy in the excited state then applying Nóse-Hoover thermostat in the ground-state.
thermodelay	sets the number of MD step delayed for applying a thermostat in the ground-state. It is only used when set **thermo** is set to 2.
silent	turns off printing output on screen. It is turned on in default.
verbose	controls the printing level. 0 only prints energy and state populations, 1 prints coordinates, velocities, gradients, and NACs, 2 prints more calculations information (screen output only).
direct	sets the number of MD steps to be written in the output file. It starts from the first step.
buffer	sets the number of MD steps to be skipped in output file after direct writing steps.
record	sets the number of the latest MD steps in a trajectory to be cached in memory. The cached trajectories are used to sample uncertain data in adaptive sampling. Reduce this number if the molecular dynamics have a huge number of steps or the adaptive sampling does not have enough memory to proceed.
checkpoint	sets the number of MD steps to checkpoint a trajectory. The trajectory is stored in python pickle file (.pkl) and can be used to restart the calculation. It is turned off in default.
restart	reads the .pkl file to restart a calculation. It is turned off in default.
addstep	adds additional MD steps in the restarted calculation. Use this if you want

to continue to propagate a completed trajectory.

5.6. NN

The neural networks in PyRAI^{2}MD are implemented with TensorFlow/Keras API. The neural network is built upon fully connected feedforward multilayer perceptrons. It consist of an input layer, several hidden layers, and an output layer. Each layer is connected by multiple neurons with activation functions. The connection between layers is a linear function including weights and bias.

PyRAI^{2}MD offers a convenient interface to train a neural network and load a trained model for the prediction of energies, forces, non-adiabatic couplings, and spin-orbit couplings. PyRAI^{2}MD always trains two sets of neural networks, which can have completely different architectures or only different initial weights. This is useful to measure the prediction uncertainty when predicting data out of the training set. The energies and forces are combined in one model and the non-adiabatic couplings and spin-orbit couplings use an independent model. Users can choose to train either one or all of them.

The keywords, default values, and short descriptions are listed below.

&NN		
modeldir	$PWD	path to save or load NN
train_data	None	path to load training data
pred_data	None	path to load prediction data
nn_eg_type	1	number of energy+gradient model
nn_nac_type	0	number of nac model
nn_soc_type	0	number of soc model
eg_unit	si	unit of energy+gradient model
nac_unit	si	unit of nac model
soc_unit	si	unit of soc model
permute_map	No	path to permutation map for data augmentation
silent	1	no output prints on screen

Full descriptions for all available keywords are summarized below.

modeldir	sets a path to save or load a NN model. The default location is the present folder. The model is saved in a folder named as "NN-$*xxx*".

train_data sets a path to load the training data from a JSON file. See X for the information of data format. If a file name is provided, it assumes that the file is in the current folder.

pred_data sets a path to load the prediction data from a JSON file. If a file name is provided, it assumes that the file is in the current folder. It is only used when **jobtype** is set to prediction.

nn_eg_type defines the number of energy+force models with different architectures. Available options are:

1	build two neural networks with the same architecture but being initialized with different weights. The hyperparameters are read from &**EG**.
2	build two neural networks with different architecture being initialized with different weights. The hyperparameters are read from &**EG** and &**EG2**, respectively.

nn_nac_type defines the number of nac models with different architectures. Available options are:

0	skip the nac model.
1	build two neural networks with the same architecture but being initialized with different weights. The hyperparameters are read from &**NAC**.
2	build two neural networks with different architecture being initialized with different weights. The hyperparameters are read from &**NAC** and &**NAC2**, respectively.

nn_soc_type defines the number of nac models with different architectures. Available options are:

0	skip the soc model.
1	build two neural networks with the same architecture but being initialized with different weights. The hyperparameters are read from &**SOC**.
2	build two neural networks with different architecture being initialized with different weights. The hyperparameters are read from &**SOC** and &**SOC2**, respectively.

eg_unit set the unit of energy and gradients used in training. Available options are:

au	energy in Hartree and gradient in Hartree·Bohr^{-1},
si	energy in eV and gradients in eV·Å^{-1}.

nac_unit set the unit of nac used in training. Available options are:

au	nac in Hartree·Bohr^{-1},

si	nac in eV·Å^{-1}.

soc_unit set the unit of nac used in training. Available options are:

si	soc in cm^{-1}.

permute_map read a text file that defined the permutations of atom indexing. Each line should only include one set of permutation. “1 5 3 2 4 6” means first switch the index of atom 2 and atom 5 then switch the index of atom 4 and the atom 2.

silent turns off printing output on screen. It is turned on in default.

5.7. SEARCH

The keywords, default values, and short descriptions are listed below.

&SEARCH		
depth	None	a list to search number of hidden layers
nn_size	None	a list to search number of neurons per hidden layer
batch_size	None	a list to search batch size
reg_l1	None	a list to search l1 factor
reg_l2	None	a list to search l2 factor
dropout	None	a list to search dropout ratio
use_hpc	0	unit of energy+gradient model
retrieve	0	read results from training logfiles

Full descriptions for all available keywords are summarized below.

depth searches a list of parameters for hidden layers, e.g., 2 3 4 5.

nn_size searches a list of parameters for number of neurons per hidden layer, e.g., 100 200 300.

batch_size searches a list of parameters for batch size, e.g, 64 128.

reg_l1	searches a list of parameters for l1 factor, e.g., 1e–5 1e–6 1e–7. It is used when **use_reg_activ**, **use_reg_weight**, or **use_reg_bias** is set to l1 or l1_l2 in &**EG**, &**EG2**, &**NAC**, &**NAC2**, &**SOC**, and &**SOC2** sections.
reg_l2	searches a list of parameters for l1 factor, e.g., 1e–5 1e–6 1e–7. It is used when **use_reg_activ**, **use_reg_weight**, or **use_reg_bias** is set to l2 or l1_l2 in &**EG**, &**EG2**, &**NAC**, &**NAC2**, &**SOC**, and &**SOC2** sections.
dropout	searches a list of parameters for dropout ratio, e.g., 0.001 0.002 0.003.
use_hpc	submits the NN training to the job scheduler. It is turned off in default, thus the training is running as a subprocess in the current machine. For training a few NNs on a nodes with many cpu, it is recommended to not use **use_hpc** because it does not have to wait in the queue. However, if there are hundreds of training in a grid search, it is better to use **use_hpc** to distribute the calculations to all available nodes via a job scheduler. To use this function, you need to prepare a submission script template with the same name as **title** in &CONTROL section, e.g. job_title.slurm and specify the all necessary #SBATCH variables.
retrieve	reads the logfiles of NN trainings in a completed grid-search and regenerate a logfile containing a summary of training results. No training calculation is performed. It is used when the grid search completed normally but the failed to print results. It is turned off in default.

5.8. EG and EG2

The keywords, default values, and short descriptions are listed below.

&EG and &EG2		
invd_index	None	path to inverse distance indices file
depth	4	number of hidden layers
nn_size	100	number of neurons per hidden layer
batch_size	64	number of data in one batch
activ	leaky_softplus	activation function
activ_alpha	0.03	activation function coefficient alpha
loss_weights	1 1	weights of energy and gradient loss
use_dropout	False	turn on dropout
dropout	0.005	dropout ratio

use_reg_activ	None	turn on regularization on activation function
use_reg_weight	None	turn on regularization on weights
use_reg_bias	None	turn on regularization on bias
reg_l1	1e–5	l1 factor
reg_l2	1e–5	l2 factor
use_step_callback	True	turn on stepwise learning rate schedular
scale_x_mean	False	shift x values to mean
scale_x_std	False	scale x values to std
scale_y_mean	True	shift y values to mean
scale_y_std	True	scale y values to std
normalization_mode	1	normalize hidden layer weights
val_split	0.1	split ratio of validation data
epo	2000	number of epochs
epostep	10	number of epochs for validation
learning_rate	1e–3	initial learning rate
learning_rate_step	1e–3 1e–4 1e–5 1e–6	stepwise learning rates
epoch_step_reduction	500 500 500 500	number of epochs for stepwise learning rate reduction

Full descriptions for all available keywords are summarized below.

invd_index	sets a path to a file containing the pairwise indices for counting inverse distance. Each line should contain a pair of atom indices. If it is not used, all pairwise distances will be included.
depth	sets the number of hidden layers.
nn_size	sets the number of neurons per hidden layer.
batch_size	sets the number of training data in one batch.
activ	sets the activation function. leaky_softplus is used in default.
activ_alpha	sets the alpha coefficient in leaky_softplus activation function.

loss_weight	sets the weights of energy and gradient loss in the total loss function. It reads two values, e.g., 1 1
use_dropout	turn on dropout during the training.
dropout	sets the dropout ratio. Note that dropout should not be used together with **use_reg_activ**, **use_reg_weight**, or **use_reg_bias**.
use_reg_activ	turn on regularization on activation function. Available options are: l1 l1 regularization, l2 l2 regularization, l1_l2 l1 and l2 regularization.
use_reg_weights	turn on regularization on hidden layer weights. Available options are: l1 l1 regularization l2 l2 regularization l1_l2 l1 and l2 regularization
use_reg_bias	turn on regularization on hidden layer bias. Available options are: l1 l1 regularization l2 l2 regularization l1_l2 l1 and l2 regularization
reg_l1	sets a l1 factor. It is used when **use_reg_activ**, **use_reg_weight**, or **use_reg_bias** is set to l1 or l1_l2.
reg_l2	sets a l2 factor. It is used when **use_reg_activ**, **use_reg_weight**, or **use_reg_bias** is set to l2 or l1_l2.
use_step_callback	turn on the stepwise learning rate schedular. It is turned on in default.
scale_x_mean	shift x values to their mean value. It is not recommended because x values are inverse distances.
scale_x_std	shift x values to their standard deviation. It is not recommended because x values are inverse distances.
scale_y_mean	shift y values to their mean value. It is used in default to standardize the target data.
scale_y_std	shift y values to their standard deviation. It is used in default to standardize the target data.

normalization_mode	normalize the weights of hidden layer to avoid gradient explosion during the training.
learning_rate	sets the initial learning rate.
val_split	sets the split ratio of validation data. The training data is evenly partitioned into k-fold accordingly. The first and second fold are used as the validation set for first and second model, respectively. The rest of data are used to training the models.
epo	sets the number of epochs.
epostep	sets the number of epochs to validate the model.
learning_rate_step	sets the stepwise reduced learning rates for each portion of epochs.
epoch_step_reduction	sets the number of epochs for each portion of learning rates reduction.

5.9. NAC and NAC2

The keywords, default values, and short descriptions are listed below.

&NAC and &NAC2		
invd_index	None	path to inverse distance indices file
depth	4	number of hidden layers
nn_size	100	number of neurons per hidden layer
batch_size	64	number of data in one batch
activ	leaky_softplus	activation function
activ_alpha	0.03	activation function coefficient alpha
phase_less_loss	False	use phaseless loss for nac
use_dropout	False	turn on dropout
dropout	0.005	dropout ratio
use_reg_activ	None	turn on regularization on activation function
use_reg_weight	None	turn on regularization on weights
use_reg_bias	None	turn on regularization on bias

reg_l1	1e–5	l1 factor
reg_l2	1e–5	l2 factor
use_step_callback	True	turn on stepwise learning rate schedular
scale_x_mean	False	shift x values to mean
scale_x_std	False	scale x values to std
scale_y_mean	True	shift y values to mean
scale_y_std	True	scale y values to std
normalization_mode	1	normalize hidden layer weights
val_split	0.1	split ratio of validation data
epo	2000	number of epochs
epostep	10	number of epochs for validation
learning_rate	1e–3	initial learning rate
learning_rate_step	1e–3 1e–4 1e–5 1e–6	stepwise learning rates
epoch_step_reduction	500 500 500 500	number of epochs for stepwise learning rate reduction

Full descriptions for all available keywords are summarized below.

invd_index	sets a path to a file containing the pairwise indices for counting inverse distance. Each line should contain a pair of atom indices. If it is not used, all pairwise distances will be included.
depth	sets the number of hidden layers.
nn_size	sets the number of neurons per hidden layer.
batch_size	sets the number of training data in one batch.
activ	sets the activation function. leaky_softplus is used in default.
activ_alpha	sets the alpha coefficient in leaky_softplus activation function.
phase_less_loss	use phaseless loss for nac.
use_dropout	turn on dropout during the training.

dropout	sets the dropout ratio. Note that dropout should not be used together with **use_reg_activ**, **use_reg_weight**, or **use_reg_bias**.
use_reg_activ	turn on regularization on activation function. Available options are: l1 l1 regularization, l2 l2 regularization, l1_l2 l1 and l2 regularization.
use_reg_weights	turn on regularization on hidden layer weights. Available options are: l1 l1 regularization l2 l2 regularization l1_l2 l1 and l2 regularization
use_reg_bias	turn on regularization on hidden layer bias. Available options are: l1 l1 regularization l2 l2 regularization l1_l2 l1 and l2 regularization
reg_l1	sets a l1 factor. It is used when **use_reg_activ**, **use_reg_weight**, or **use_reg_bias** is set to l1 or l1_l2.
reg_l2	sets a l2 factor. It is used when **use_reg_activ**, **use_reg_weight**, or **use_reg_bias** is set to l2 or l1_l2.
use_step_callback	turn on the stepwise learning rate schedular. It is turned on in default.
scale_x_mean	shift x values to their mean value. It is not recommended because x values are inverse distances.
scale_x_std	shift x values to their standard deviation. It is not recommended because x values are inverse distances.
scale_y_mean	shift y values to their mean value. It is used in default to standardize the target data.
scale_y_std	shift y values to their standard deviation. It is used in default to standardize the target data.
normalization_mode	normalize the weights of hidden layer to avoid gradient explosion during the training.
learning_rate	sets the initial learning rate.

val_split	sets the split ratio of validation data. The training data is evenly partitioned into k-fold accordingly. The first and second fold are used as the validation set for first and second model, respectively. The rest of data are used to training the models.
epo	sets the number of epochs.
epostep	sets the number of epochs to validate the model.
learning_rate_step	sets the stepwise reduced learning rates for each portion of epochs.
epoch_step_reduction	sets the number of epochs for each portion of learning rates reduction.

5.10. SOC and SOC2

The keywords, default values, and short descriptions are listed below.

&EG and &EG2		
invd_index	None	path to inverse distance indices file
depth	4	number of hidden layers
nn_size	100	number of neurons per hidden layer
batch_size	64	number of data in one batch
activ	leaky_softplus	activation function
activ_alpha	0.03	activation function coefficient alpha
use_dropout	False	turn on dropout
dropout	0.005	dropout ratio
use_reg_activ	None	turn on regularization on activation function
use_reg_weight	None	turn on regularization on weights
use_reg_bias	None	turn on regularization on bias
reg_l1	1e–5	l1 factor
reg_l2	1e–5	l2 factor
use_step_callback	True	turn on stepwise learning rate schedular
scale_x_mean	False	shift x values to mean
scale_x_std	False	scale x values to std

scale_y_mean	True	shift y values to mean
scale_y_std	True	scale y values to std
normalization_mode	1	normalize hidden layer weights
val_split	0.1	split ratio of validation data
epo	2000	number of epochs
epostep	10	number of epochs for validation
learning_rate	1e–3	initial learning rate
learning_rate_step	1e–3 1e–4 1e–5 1e–6	stepwise learning rates
epoch_step_reduction	500 500 500 500	number of epochs for stepwise learning rate reduction

Full descriptions for all available keywords are summarized below.

invd_index	sets a path to a file containing the pairwise indices for counting inverse distance. Each line should contain a pair of atom indices. If it is not used, all pairwise distances will be included.
depth	sets the number of hidden layers.
nn_size	sets the number of neurons per hidden layer.
batch_size	sets the number of training data in one batch.
activ	sets the activation function. leaky_softplus is used in default.
activ_alpha	sets the alpha coefficient in leaky_softplus activation function.
use_dropout	turn on dropout during the training.
dropout	sets the dropout ratio. Note that dropout should not be used together with **use_reg_activ**, **use_reg_weight**, or **use_reg_bias**.
use_reg_activ	turn on regularization on activation function. Available options are: l1 l1 regularization, l2 l2 regularization, l1_l2 l1 and l2 regularization.
use_reg_weights	turn on regularization on hidden layer weights. Available options are: l1 l1 regularization l2 l2 regularization

	l1_l2 l1 and l2 regularization
use_reg_bias	turn on regularization on hidden layer bias. Available options are: l1 l1 regularization l2 l2 regularization l1_l2 l1 and l2 regularization
reg_l1	sets a l1 factor. It is used when **use_reg_activ**, **use_reg_weight**, or **use_reg_bias** is set to l1 or l1_l2.
reg_l2	sets a l2 factor. It is used when **use_reg_activ**, **use_reg_weight**, or **use_reg_bias** is set to l2 or l1_l2.
use_step_callback	turn on the stepwise learning rate schedular. It is turned on in default.
scale_x_mean	shift x values to their mean value. It is not recommended because x values are inverse distances.
scale_x_std	shift x values to their standard deviation. It is not recommended because x values are inverse distances.
scale_y_mean	shift y values to their mean value. It is used in default to standardize the target data.
scale_y_std	shift y values to their standard deviation. It is used in default to standardize the target data.
normalization_mode	normalize the weights of hidden layer to avoid gradient explosion during the training.
learning_rate	sets the initial learning rate.
val_split	sets the split ratio of validation data. The training data is evenly partitioned into k-fold accordingly. The first and second fold are used as the validation set for first and second model, respectively. The rest of data are used to training the models.
epo	sets the number of epochs.
epostep	sets the number of epochs to validate the model.
learning_rate_step	sets the stepwise reduced learning rates for each portion of epochs.
epoch_step_reduction	sets the number of epochs for each portion of learning rates reduction.

5.11. FILE

The keywords, default values, and short descriptions are listed below.

&FILE		
natom	0	number of atoms
file	None	path to a list file to read QC calculation results

Full descriptions for all available keywords are summarized below.

natom	sets the number of atoms for reading the coordinates from the QC calculation logfiles.
file	read the path to a list file for extracting the QC data from the calculation logfiles. If a file name is provided, it assumes that the list file is in the current folder. In the list file, each line should contain a path to a QC calculation folder.

5. Nonadiabatic molecular dynamics

5.1. Fewest switches surface hopping

5.2. Zhu-Nakamura surface hopping

6. Machine learning models

6.1. Preparing training data

6.2. Creating a neural network

First, we create a model to predict energies, forces, and non-adiabatic couplings (if requested). The input example below shows the frequently used keywords for creating modes.

jobtype determines the type of calculation. It takes 'train' for training neural networks, 'prediction' for predicting e
nergies, forces, and non-adiabatic couplings, 'adaptive' for adaptive sampling of conformational space using molecular dynamics trajectories, and 'md' for molecular dynamics simulation.

PyRAI^{2}MD has a flexible training scheme depending on the available computing resources. When **ml_ncpu = 1**, all models will be trained sequentially. When **ml_ncpu <=4**, all models will be trained in subprocess so they can use all given numbers of CPUs. If **ml_ncpu > 4**, the extra CPU resources will be used to parallelize the training, which is automatically managed by TensorFlow.

6.3. Training a neural network

6.5. Adaptive sampling

7. External quantum chemical program

7.1. Molcas

7.2. BAGEL

7.3. ORCA

7.4. GAMESS

7.5. GFN-xTB

7.6. MNDO