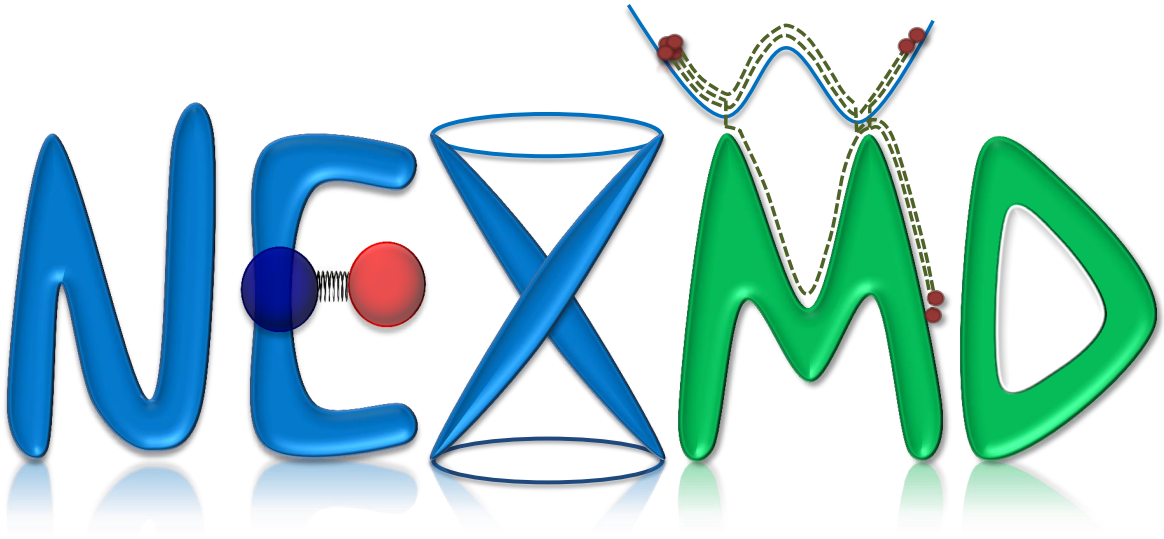
Non-adiabatic EXcited-state Molecular Dynamics (NEXMD) Reference Manual Version 2.0.2

immediate



Contents

1 Introduction **2**

2 NEXMD Main Features **3**

3 Compiling NEXMD **5**
3.1 Stack size 5
3.2 Avoiding edits to the Makefile 5
3.3 Advance configuration 5
3.4 Tests after compiling 5

4 Running NEXMD **6**

5 The input file **8**

6 Output Files **20**

7 Restarting Simulations **26**

8 Input examples **27**
8.1 Optimization 27
8.2 Normal modes calculation 30
8.3 TSH dynamics 34
8.4 Mean field Ehrenfest dynamics 37
8.5 AIMC dynamics 40

A Contributors **45**
A.1 Principal Investigators 45
A.2 Developers 45

B Observable calculation for AIMC **46**
B.1 Observable over the electronic subspace 46
B.2 Observable over the nuclear subspace 48

1 Introduction

NEXMD, from Non-adiabatic EXcited state Molecular Dynamics [1,2], is a software package designed for simulating the non-radiative electronic relaxation taking place in a molecular system after a photo-excitation. The intricate mechanisms taking place between the excited states manifold underpins processes in nature such as vision [3] or photosynthesis [4]. A detailed understanding of these mechanisms have also allowed a wide range of applications including but not limited to organic light emitting diodes [5], photovoltaics [6–8], field-effect transistors [9, 10], sensors [11–15], photocatalysts [16] and solar cells [17, 18]. The complex excited state electronic structure arising from strong electronic correlations and low dimensionality [19], combined with delocalized and polarizable π-electrons are key for the generation of mobile charge carriers [20]. Such systems typically undergo an efficient non-radiative relaxation [21] that can take place through several nonadiabatic pathways leading to overall dissipation of an excess of electronic energy into heat. In this context, many physical processes such as internal conversion [22], energy transfer [23], charge separation [24, 25], exciton self-trapping [26] or vibronic coherences [27–29] can be of relevance.

NEXMD is able to simulate these processes from a full-atom perspective treating molecules that have a few hundreds of atoms, including a few dozens of excited states and for a few picoseconds. The electronic structure is based in the Colective Electronic Oscillator (CEO) method [30], considering Configuration Interaction Singles [31] or the Time-Dependent Hartree-Fock (TDHF) [32,33] combined with semiempirical Hamiltonians models. The initial implementation (NEXMD Version 1.0), included the Trajectory Surface Hopping approach [34]. Recent implementations have added Ehrenfest dynamics [35] and the Ab-Inition Multiple Cloning sampling for Multiconfigurational Ehrenfest [36]. These methods in combination with other features present in NEXMD allow a comprehensive investigation of a wide range of processes and systems, including but not limited to polymers [37–43], dendrimers [29, 44–50], nanorings and nanobelts [51–56], light harvesting complexes [57–59] and energetic materials [60–62].

This document is a user manual covering technical details required for using NEXMD, such as available features, compilation and execution, input file extensive description and output files formatting. It also contains a section with input files examples intended to be used as templates for the specific features available within NEXMD. This manual is not intended to cover the theoretical background supporting the features of NEXMD. The interested reader can use instead a recent extensive review on NEXMD [2] or specialized papers on the development of a given feature or application. While this manual is a useful guide for the new user, more guided tutorials (specific to particular use cases) are available elsewhere.

2 NEXMD Main Features

NEXMD/NEXMD v2 have been principally developed to perform on-the-fly non-adiabatic excited state molecular dynamics simulations at TDHF or CIS level of electronic structure using semiempirical Hamiltonians. It also comprises several features that are useful when analyzing the photophysics of a molecular system. In this section we enumerate some of NEXMD features and corresponding input keywords, while the full list and the details for configuring the corresponding input parameters can be found in the Section 5.

- Electronic structure single point is done when both the classical steps `n_class_steps` and the optimization cycles `maxcyc` are set to zero. In the standard output (usually denoted as `md.out`), NEXMD will write the ground state and excited state energies along with the corresponding electric transition dipole moments and oscillator strengths. Furthermore the following options are available:
  - Mulliken charges: Can be set with `printcharges`.
  - Transition dipole moments between excited states: Can be set with `calcxdens`.
  - Transition density plots: Can be set with `out_data_cube`. This option will generate a set of `.DATA` files that can be converted to `.cube` files and plotted with any standard chemical software.
- Optimization can be done for the ground state or any excited state by setting the variable `maxcyc` to any value greater than zero. The number of classical steps has to be reduced to zero otherwise an error will show up in the standard output.
- Hessian and normal mode calculations can be set with `do_nm`. Both the number of classical steps `n_class_steps` and the optimization cycles `maxcyc` have to be set to zero.
- Molecular dynamics can be set when the number of classical steps `n_class_steps` is set to a value greater than zero. NEXMD includes the following types of molecular dynamics:
  - Ground state molecular dynamics is performed when the initial excited state `exc_state_init` is set to zero. If the number of excited states to calculate `\n_exc_states_propagate` is set to number greater than zero, the energies and transition dipole moments of the corresponding states will be calculated along the ground state dynamics.
  - Adiabatic excited state dynamics can be activated by setting the initial excited state `exc_state_init` to a value greater than zero and the Born-Oppenheimer flag `bo_dynamics_flag` to one. The number of excited states to propagate in the input variable `n_exc_states_propagate` has to be greater than `exc_state_init`.
  - Non-adiabatic excited state dynamics can be activated setting the initial excited state `exc_state_init` to a value greater than zero and the Born-Oppenheimer flag `bo_dynamics_flag` to zero. By default the non-adiabatic dynamic method

is the trajectory surface hopping. The type of non-adiabatic molecular dynamics can also be set to perform Ehrenfest or Ab-initio Multiple Cloning dynamics by means of the variable `NAMD_type`.

- Langevin thermostat can be activated setting `therm_type` to one for any of the types of molecular dynamics, although it is advised to only use it for ground state dynamics.
- Implicit solvation schemes are available by setting the input variable `solvent_model`. Vertical excitation and state specific methods are not advised for non-adiabatic dynamics since instantaneous quantum transitions instantaneously change the charge density, which in turn instantly changes polarized solvent field leading to an unphysical sudden change of the excited state solutions.
- Constraints can be activated between atom distances or for generalized directions, e.g. normal modes. Only one type of constraint is advised to be used in a calculation.
  - Constraints for atom distances can be activated with the input variable `npc`.
  - Constraints for generalized directions can be activated with the input variable `nmc`.

3 Compiling NEXMD

Expanding upon the readme, the `Makefile` for `NEXMD` is handwritten for both `GNU` and `Intel` Fortran compilers and libraries with their default names and locations. By default a serial binary is compiled and libraries are statically linked. The `Makefile` should work for most standard systems (sometimes only after loading the relevant compiler and library module)

Calling the `make` command in the NEXMD directory will call the default `Intel` Fortran compiler with `Intel` MKL libraries, equivalent to calling `make all` and `make ic_mkl`

The other tested option is `make gnu` which will call the default `GNU` Fortran compiler with BLAS and LAPack libraries in their default location.

There are commented out options, as examples for tinkering with the Makefile for advanced users.

3.1 Stack size

Similar to most programs, if `nexmd.exe` has insufficient space on the stack it will crash with no useful error messages.

Before running calculations, it is useful to ensure there is sufficient stack size, for `BASH`:

```
ulimit -s unlimited
```

or

```
ulimit -s hard
```

This may require super-user permissions.

3.2 Avoiding edits to the Makefile

For `BASH`, the default `Makefile` can be used with non-default compiler locations by adding these locations to the path:

```
export PATH = <NEXMD directory>:$PATH
```

For `BASH`, it may be possible that custom libraries can be softlinked to the main repository.

```
ln -s <custom library file> <NEXMD directory>/<special_lib>
```

3.3 Advance configuration

Advanced users on custom systems may find that they need NEXMD compiled with custom compilers and libraries outside the default location. The `Makefile` contains empty variables under the flag `custom` for such custom compilers and libraries; for complete flexibility.

3.4 Tests after compiling

`<NEXMD directory>/tests` contains template input and bash scripts for NEXMD2 calculations on benzene; `<NEXMD directory>/tests/reference` contains reference output of the tests.

4 Running NEXMD

NEXMD is a command-line program which always uses a main input file named `input.ceon`, and rarely requires additional input files. The typical workflow is constructing the input in a fresh directory and calling the program as:

```
/path_to_nexmd_bin/nexmd.exe > md.out &
```

and it is possible to use a main input file with a non-standard name with

```
/path_to_nexmd_bin/nexmd.exe < custom_input.ceon > md.out & .
```

Several output files (in addition to the standard output which above is written to `md.out`) will be generated depending on the input parameters. More details about the output files content and format can be found in section 6 of this manual.

Example NEXMD `input.ceon` files with the expected output are present in the test directory found at `<NEXMD directory>/tests`. These examples cover most of the features present in NEXMD. It is advised to use these examples to test the NEXMD immediately after compilation. Furthermore, when building a new `input.ceon` file, it is also advised to use as template one of the `input.ceon` in the `tests` directory or in this manual, and modify only the necessary parameters for activating the desired features. This minimal changes are intended to avoid combinations of input keywords that are not benchmarked, not physically justified, or are partially ignored by the sequential reading of keywords as a safeguard to reduce wasting computational resources.

It is also important to notice that there are three mutually-exclusive calculations that NEXMD can perform, enforced by the sequential reading of the main input file. These are optimization, normal modes calculation and molecular dynamics propagation. A main input file suggesting to perform more than one of these calculations will result in an error written to standard output.

Among NEXMD output files, there is one named `restart.out` written for all the molecular dynamics flavors, which is an input file for restarting the dynamics. If such file is present, NEXMD will restart the simulation from the last molecular dynamics step. The program can be called in a similar way:

```
/path_to_nexmd_bin/nexmd.exe < input.ceon >> md.out &
```

or

```
/path_to_nexmd_bin/nexmd.exe < restart.out >> md.out &
```

Note the `>>` before `md.out` to append instead of overwriting the standard output file. More details about restarting can be found in section 7 of this manual.

The usual NEXMD workflow when doing research can be followed in Figure 1. Each of these steps is either an individual run, a set of individual runs, or the analysis of a set of output files.

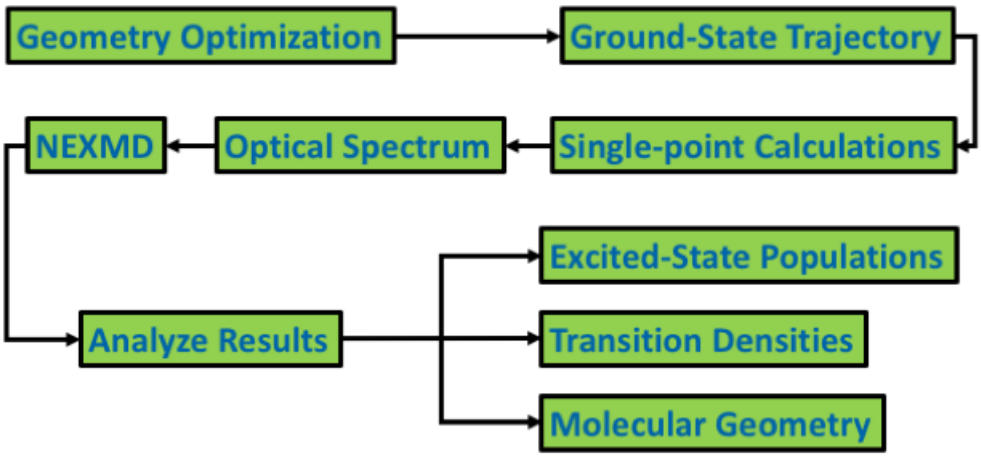


Figure 1: A schematic of the general procedure for simulating non-adiabatic dynamics.

5 The input file

The contents of the input file for NEXMD can be categorized into blocks: &qmmm for geometry optimization, ground-state parameters, excited-state parameters, and solvent models; &moldyn for molecular dynamics, thermostats, and constraints; &pairs and &modes for specifying constraints; &coord for initial geometries; &veloc for initial velocity and &coeff for excited-state coefficients. All input parameters are overviewed here, the numerical values in square brackets are default parameters.

```
&qmmm
   !***** Geometry Optimization
```

The following section contains input parameters pertaining to geometry optimization.

```
   maxcyc=0, ! Number of cycles for geometry optimization [0]
   ntpr=1, ! Print results every ntpr cycles [1]
   grms_tol=1.0d-2, ! Tolerance in eV/A (derivatives) [1.0d-2]
```

- maxcyc sets the maximum number of cycles for geometry optimization. If the number of cycles reaches maxcyc, an error message reads: Maximum number of iterations reached without convergence. Therefore, to optimize geometry, maxcyc must be set to a sufficiently large number greater than zero. If the geometry reaches the maximum number of cycles, you must review the situation carefully to determine if the input geometry was appropriate or if grms_tol (shown below) was set too low. A negative maxcyc is treated as 0. All files written according to the corresponding verbosity will be written for the optimized geometry.
- ntpr sets how often results for geometry optimization are printed to the standard output file. The output of the iteration is written as iter, energy in eV, and rms gradient in eV per Å. The latter quantity is explained in grms_tol, shown below.
- grms_tol sets the convergence criteria for geometry optimization. The units for grms_tol are in eV per Å, so that a smaller grms_tol is a smaller change in energy per change in bond length (i.e. a tighter convergence criteria).

```
   !***** Normal mode analysis (NMA)
```

The following section contains input parameters pertaining to normal mode calculations.

```
    do_nm=1, ! 0 for not doing NMA, greater than 0 for doing NMA [0]
    deltaX = 1.0d-4, ! Displacement for second derivatives required Hessian calculation (Angstr
[1.0d-4]
```

- do_nm sets NEXMD to perform Normal Mode Analysis. Two files will be generated: nma_modes.out and nma_freq.out containing respectively the normal modes directions and the normal modes energies (Hartree). The NMA calculation can be performed for the ground state or for any excited state. It is recommended that this calculation is performed for a minimum energy configuration. Non adiabatic corrections [63] have still not been added to the current version.

- `deltaX` sets the displacement (Å) for the numerical Hessian calculation.

```
!***** Ground-State and Output Parameters
```

The following section contains input parameters pertaining to the ground-state calculation and its associated outputs.

```
qm_theory='AM1', ! Integral type, check Amber's SQM for more options [AM1]
scfconv=1.0d-8, ! Ground-state SCF convergence criteria, eV [1.0d-6]
verbosity=0, ! QM/MM output verbosity (0-minimum, 5-maximum)
! [1 for dynamics and optimization, 5 for others]
printdipole=2, ! (0) Unrelaxed transitions, (1) Unrelaxed transitions plus
! total molecular, or (2) Unrelaxed/relaxed transitions plus
! total molecular [1 for dynamics and 2 for single-point]
itrmax=300, ! Max SCF iterations for ground state
! (negative to ignore convergence) [300]
```

- `qm_theory` sets the semi-empirical model Hamiltonian. A list of available Hamiltonians can be found in Ref. [64]. The majority of studies with `NEXMD` have used Austin Model 1 (AM1).

- `scfconv` sets the convergence criteria for the self-consistent field (SCF) ground-state energy in eV. In other words, the user requests that the ground-state energy be determined to within `scfconv` eV.

- `verbosity` sets the level of printing of QM/MM related outputs. The outputs of each level can be found in Ref. [64]. The verbosity should be set to at least `1` in order to obtain ground-to-excited state oscillator strengths. This is important for single-point calculations and obtaining an optical spectrum.

- `printdipole` sets the printing level for transition dipole moments. There are three options to choose from which are: `(0)`, x,y,z coordinates of ground-to-excited state transition dipole moments with respect to an arbitrary lab. frame, `(1)`, same as `(0)` plus the module of the total molecular transition dipole moment, and `(2)`, same as `(1)` plus relaxed and unrelaxed difference transition dipole moments.

- `itrmax` sets the maximum number of cycles for the ground-state SCF calculation. If the number of cycles reaches `itrmax`, the code will stop. A negative value for `itrmax` means the ground-state SCF calculation will go through |`itrmax`| cycles regardless of whether or not the convergence criteria in `scfconv` has been met.

```
!***** Excited-State Parameters
```

The following section contains input parameters pertaining to the excited-state calculation.

```
exst_method=1, ! CIS (1) or RPA (2) [1]
dav_guess=1, ! Restart Davidson from (0) Scratch, (1) Previous,
ftol0=1.0d-7, ! Acceptance tolerance (|emin-eold|) [1.0d-5]
dav_maxcyc=200, ! Max cycles for Davidson diagonalization
```

```
! (negative to ignore convergence) [100]
printcharges=0, ! Print (1) or do not print (0) Mulliken charges of QM atoms [0]
calcxdens=.false., ! Print (.true.) or do not print (.false.)
! excited-to-excited transition dipole moments [.false.]
```

- `exst_method` sets the approximate excited-state wavefunction, which is used to compute excited-state properties. There are two options to choose from in `NEXMD`. The first is the configuration interactions singles (`CIS`) wavefunction and the other is the random phase approximation wavefunction (`RPA`). The RPA wavefunction is a slight extension of the CIS wavefunction and includes more electron correlation effects. Therefore, it is common for RPA to be more computationally demanding than CIS. The CIS wavefunction is stable and has been used for the majority of studies with `NEXMD`.

- `dav_guess` sets the initial guess for the Davidson algorithm. The Davidson algorithm is used to calculate excited-state eigenvalues and eigenvectors. One option is to start Davidson from `scratch`. However, this option may increase computation time. Another option is start Davidson from the results of the `previous` calculation. The latter option should be used for realistic simulations.

- `ftol0` sets the convergence criteria on excited-state energies. In other words, the user requests that excited-state energies be determined to within `ftol0` eV.

- `dav_maxcyc` sets the maximum number of cycles for the Davidson algorithm. If the number of cycles exceeds `dav_maxcyc`, an error message will read: `Number of Davidson iterations exceeded, exiting.` A negative value for `dav_maxcyc` means the excited-state calculation will go through |`dav_maxcyc`| cycles regardless of whether or not the convergence criteria in `ftol0` has been met.

- `printcharges` sets whether or not to print Mulliken charges of QM atoms to the standard output file.

- `calcxdens` sets whether or not to print excited-to-excited transition dipole moments. This option can only be set to `true` during single-point calculations. An error will occur if `calcxdens` is set to `true` during dynamics. A file called `muab.out` will be generated if `calcxdens=.true.`, which contains excited-to-excited transition dipoles in arbitrary units. The number of excited states included in `muab.out` depends on the number of excited states being propagated, which is controlled by `n_exc_states_propagate`, located in an upcoming section of the input file.

```
!***** Solvent Models and External Electric Fields
```

The following section contains input parameters pertaining to solvent models and external electric fields.

```
solvent_model=0, ! (0) None, (1) Linear response, (2) Vertical excitation,
! or (3) State-specific [0]
potential_type=1, ! (1) COSMO or (2) Onsager [1]
onsager_radius=2, ! Onsager radius, A (system dependent) [2]
```

```
ceps=10, ! Dielectric constant, unitless [10]
linmixparam=1 ! Linear mixing parameter for vertical excitation
! or state-specific SCF calculation [1]
cosmo_scf_ftol=1.0d-5, ! Vertical excitation or state-specific
! SCF tolerance, eV [1.0d-5]
doZ=.false. ! Use relaxed (.true.) or unrelaxed (.false) density for
! vertical excitation or state-specific COSMO or Onsager [.false.]
EF=0, ! (0) None or (1) Electric field in ground and excited state [0]
Ex=0, ! Electric field vector X, eV/A [0]
Ey=0, ! Electric field vector Y, eV/A [0]
Ez=0, ! Electric field vector Z, eV/A [0]
```

- `solvent_model` sets the solvent model between gas phase, linear response, vertical excitation or state specific with the values 0, 1, 2 and 3 respectively. A detailed description of these models can be found in Refs. [65, 66].

- `potential_type` sets the potential of the solvent model. The Onsager model assumes that the solute is placed in a spherical cavity inside the solvent. The latter is described as a homogeneous, polarizable medium of constant dielectric constant given by `ceps`. The solute dipole moment induces a dipole moment of opposite direction in the surrounding medium. Polarization of the medium in turn polarizes the charge distribution in the solvent. Treating this mutual polarization in a self-consistent manner leads to the Onsager reaction field model. COSMO (Conductor-like Screening Model) generalizes the Onsager potential where the cavity surface is defined by the shape of the solute. COSMO is a more complete description of the electrostatic interactions.

- `onsager_radius` defines the radius of the spherical cavity for the Onsager reaction field model.

- `ceps` sets the dielectric constant of the solvent. A list of solvents and their dielectric constants can be found in Ref. [67]. Be sure to reference Ref. [67].

- `linmixparam` sets the degree to which the last two SCF iterations are mixed. The mixed solution is used as an input for the following SCF iteration. The goal of introducing `linmixparam` is to significantly reduce the high cost of finding the SCF solution by inputting a solvent potential into the current iteration that is extrapolated from previous iterations. See Refs. [65, 66] for more information.

- `cosmo_scf_ftol` sets the convergence criteria of the vertical excitation or state-specific solvent model SCF calculation. In other words, the user requests that the energy be determined to within `cosmo_scf_ftol` eV. The `cosmo_scf_ftol` flag sets the tolerance for both the Onsager and COSMO potentials.

- `doZ` sets whether to use the relaxed or unrelaxed density for the vertical excitation or state-specific solvent model.

- `EF` sets whether or not there is an external electric field applied to the system.

- `Ex`, `Ey`, `Ez` sets the magnitude and direction of the external electric field in the x, y, and z axes, respectively. The user must also set `EF` to `1` if an external electric field is desired in the simulation.

General Note: The default tolerances are within accepted levels of convergence. However, these values may be increased depending on the size of the system or time of simulation.

```
&endqmmm

&moldyn
   !***** General Parameters
```

The following block contains general input parameters pertaining to molecular dynamics.

```
NAMD_type = 'tsh', ! NAMD method [tsh, aimc, mf]
natoms=12, ! Number of atoms
! (must be equal to the number of atoms in system)
rnd_seed=19345, ! Seed for the random number generator
bo_dynamics_flag=0, ! (0) Non-BO or (1) BO [1]
exc_state_init=6, ! Initial excited state (0 - ground state) [0]
n_exc_states_propagate=8, ! Number of excited states [0]
```

- `NAMD_type` sets the NAMD method: trajectory surface hoping (tsh), Ehrenfest (mf), and ab initio multiple cloning (aimc) [36, 68].

- `natoms` sets the number of atoms in the system being studied. This number is important for memory allocation and must be equal to or greater than the number of atoms in the system.

- `rnd_seed` sets the seed for the random number generator. For each `rnd_seed`, there is a well-defined sequence of random numbers. For non-adiabatic ensemble simulations, `rnd_seed` must be different from one trajectory to another. This ensures the stochastic nature of the simulation, which is important for both the nuclear Langevin dynamics (i.e. coupling of the system to a heat bath) and the surface hopping algorithm, [34] which governs electronic transitions between electronic states (i.e. non-adiabatic dynamics).

- `bo_dynamics` sets whether the dynamics is non-Born–Oppenheimer (non-adiabatic) or Born–Oppenheimer (adiabatic). If the simulation is non-adiabatic, this typically means the user is running an ensemble of trajectories. This may also be the case for an adiabatic simulation, depending on the study.

- `exc_state_init` sets the initial excited-state of the system. For a non-adiabatic ensemble simulation, a distribution of initial excited-states is needed to model a photo-excited wavepacket of different nuclear geometries. Therefore, `exc_state_init` may be different from one trajectory to another.

- `n_exc_states_propagate` sets the total number of excited-states to be propagated in the dynamics. The user must be careful not to include unnecessary higher-energy states if it is unlikely for the system to access those states as this may greatly increase computation time. The number of excited-states to include in the simulation is determined by the electronic structure and optical spectrum of the system.

```
!***** Dynamics Parameters
```

The following section contains more inputs for molecular dynamics.

```
time_init=0.0, ! Initial time, fs [0.0]
time_step=0.1, ! Time step, fs [0.1]
n_class_steps=10000, ! Number of classical steps [1]
n_quant_steps=4, ! Number of quantum steps for each classical step [4]
moldyn_deriv_flag=1, ! (0) None, (1) Analytical, or (2) Numerical [1]
num_deriv_step=1.0d-3, ! Displacement for numerical derivatives, A [1.0d-3]
```

- `time_init` sets the initial time of the trajectory. If the trajectory has not yet begun, then this number is set 0.0 fs. However, it is common for a trajectory to be restarted from where it left off. In these cases the initial time depends on when the previous simulation has ended. Restart input files will be discussed in Section 7.

- `time_step` sets the classical time-step of the trajectory. This is the time-step at which nuclear degrees of freedom are integrated.

- `n_class_steps` sets the total number of classical time-steps in the trajectory. An error message will read: `You must run dynamics (n_class_steps > 0) or geometry optimization (maxcyc > 0). Running both simultaneously is not possible.`

- `n_quant_steps` sets the number of quantum steps per classical step. The nuclear degrees of freedom are integrated with the Velocity Verlet (VV) algorithm, while the electronic degrees of freedom (i.e. the quantum coefficients) are integrated with the Runge-Kutta (RK) algorithm. The nuclear dynamics are more computationally stable than the quantum coefficients. The VV algorithm, while less computationally demanding than RK, is sufficient for nuclear dynamics. Quantum coefficients are more susceptible to computational instabilities and require a more rigorous method for integrating their equations of motion (i.e. the Schrödinger equation).

- `moldyn_deriv_flag` sets how gradients are calculated. The options here are numerical or analytical. Generally, analytical derivatives should be used because they are more computationally stable and less computationally expensive than numerical derivatives.

- `num_deriv_step` sets the derivative step-size in units of angstroms (Å) when the `moldyn_deriv_flag` is set to `numerical`.

```
!***** Constraints
```

The following section contains input parameters pertaining to constrained distances and normal modes.

```
npc=0, !Number of pairs of distances to be constrained [0]
nmc=0, !Number of normal modes to be constrained [0]
```

- `npc` is the number of pairs of distances tobe constrained [69]. In the current version the same atom can't belong to more than two constrained pairs. An extra block `&pairs` is needed to add the indexes of the pairs of atoms to be constrained. The current version can't constrain atom distances for minimization.

- `nmc` is the number of normal modes to be constrained [70]. The current version can't constrain modes and distances at the same time. Two extra files are needed: `reference.xyz` and `nma_modes.out`. `reference.xyz` contains the reference structure for which normal modes where calculated in `xyz` format. At each time step, NEXMD will translate and rotate in order to minimize the root mean squared deviation to the structure stored in `reference.xyz`. The `nma_modes` contains the 3*`natom` normal modes directions (stored in columns). Any set of generalized directions can be constrained as long as the directions are orthonormalized. An extra block `&modes` is needed to add the indexes of the normal modes to be constrained. The current version can't constrain normal modes for minimization.

```
!***** Non-Adiabatic Parameters TSH
```

The following section contains input parameters pertaining to trajectory surface hopping dynamics.

```
decoher_type=2, ! Type of decoherence: Reinitialize (0) Never,
! (1) At successful hops, (2) At successful plus frustrated hops...
dotrivial=1, ! Do unavoided (trivial) crossing routine (1) or not (0) [1]
quant_step_reduction_factor=2.5d-2, ! Quantum step reduction factor [2.5d-2]
iredpot=1, ! Reduce excited states for tsh, 1 for yes and 0 for no [0]
nstates=2, ! Number of excited states to reduce [2]
```

- `decoher_type` sets the method for decoherence. The surface hopping method treats nuclear degrees of freedom classically and electronic degrees of freedom quantum mechanically. Due to the classical treatment of nuclei, there is an overcoherence between quantum states. In realistic simulations there should always be some form of decoherence. Method `0` does not introduce any form of decoherence and should only be used for code testing or benchmarking purposes. By reinitializing the quantum coefficients after hops, this introduces a form of decoherence that is instantaneous. Method `1` collapses the wavefunction at all successful hops, whereas `2` collapses the wavefunction at all successful plus frustrated hops. Frustrated hops are those that were unable to satisfy energy conservation and were rejected. In general, method `2` should be used for realistic simulations.

- `dotrivial` sets whether or not to reduce the time-step in the vicinity of trivial crossings. See description under `quant_step_reduction_factor` for more details. Trivial crossings are identified by the method described in Ref. [71].

- `quant_step_reduction_factor` sets how much to reduce the quantum step in the vicinity of an unavoided or trivial crossing. At trivial crossings, the energy gap between the adjacent states is vanishingly small. The coupling between these states is spiky localized in time. Therefore, in order to resolve the non-adiabatic coupling between these states and determine whether or not an electronic transition occurs, the time-step must be reduced. It is defined as `quant_step_reduction_factor` × `quant_time_step`, where `quant_time_step` is determined by `time_step` and `n_quant_steps`.

- `iredpot` is set to 1 to reduce the total number of excited states calculated after a hop for trajectory surface hopping dynamics. This is particularly useful for big systems with tens of states. If `iredpot` is set to 0 the number of states will remain constant during the complete simulation. Otherwise a reduction of states algorithm will be appliyed. [72]

- `nstates` sets how many states will remain above the current state after a hop in order to allow hopping up. The reduction of states is irreversible and some extra states should be calculated above the current state in order to account for hopping up.

```
!***** Non-Adiabatic Parameters AIMC
```

The following section contains input parameters pertaining to AIMC dynamics. AIMC dynamics may generate several trajectories from the same initial condition. Two types of files will be generated: those ending in `.out` will refer to a given trajectory with a four digit label and those ending in `.dat` will correspond to the complete ensemble. Ehrenfest dynamics works just as AIMC but without cloning, so no ensemble files will be produced and the trajectory files will not be labeled.

```
AIMC_dclone_1=1.5, ! AIMC threshold for the first criterion [1.5]
AIMC_dclone_2=0.2617, ! AIMC threshold for the second criterion [0.2617]
AIMC_dclone_3=0.005, ! AIMC threshold for the third criterion [0.005]
AIMC_max_clone=4, ! Max number of consecutive branching for AIMC [4]
nclones0=0, ! Initial count of the number of consecutive branching for AIMC [0]
S0_S1_threshold=0.2, ! For dropping trajectories when S0/S1 energy gap is to low [0.2]
```

- `AIMC_dclone_1` is the threshold for the first cloning criterion, see [36] for details.

- `AIMC_dclone_2` is the threshold for the second cloning criterion, see [36] for details. The default numerical value corresponds to 15 degrees in radians.

- `AIMC_dclone_3` is the threshold for the third cloning criterion, see the supplementary information of [73] for details.

- `AIMC_max_clone` prevent the AIMC algorithm to have an uncontrolled number of cloining events by allowing a limited number of consecutive clones. The default value 4 allows a maximum of $4^2 = 16$ clones per initial condition. Convergence tests for dendrimer systems have shown that after approximatelly 13 clones per initial condition there is no significant improvement of results [68].

- `nclones0` is the initial number of clones that already happened for a given trajectory. This variable is only used for restarting purposes.

- `S0_S1_threshold` is the threshold for dropping trajectories when the energy gap between S_0 and S_1 is to low. AIMC dynamics simulation will continue for the remaining trajectories (if any). The remaining trajectories will be renormalized in order to continue the propagation. Information about the time and label of the dropped out trajectory will be printed in the `dropped.out` file. `S0_S1_threshold` is also valid for trajectory surface hopping dynamics, for which the simulation will simply stop if the threshold is reached. The default value is 0.2 eV.

```
!***** Thermostat Parameters
```

The following section contains input parameters pertaining to the thermostat.

```
therm_type=1, ! Thermostat type: (0) Newtonian, (1) Langevin,
therm_temperature=300, ! Thermostat temperature, K [300]
therm_friction=20, ! Thermostat friction coefficient, 1/ps [20]
```

- `therm_type` sets the type of thermostat to be used in the simulation. This determines the equation of motion governing nuclear dynamics. There are two options for this input, one of which is `Newtonian`. This is the same as introducing no thermostat, i.e., simulations at constant energy, The other option is `Langevin`. The Langevin equation of motion is a stochastic differential equation that introduces terms for viscosity and a Gaussian random force that controls temperature in such a way that obeys the canonical ensemble. In realistic simulations, the thermostat should be set to `Langevin`.

- `therm_temperature` sets the temperature of the thermostat in units of Kelvin (K).

- `therm_friction` sets the friction parameter for the Langevin thermostat in units of inverse picoseconds (ps^{-1}). This input generally depends on the viscosity of the solvent that is being modeled. However, in most cases, the default parameter should be used.

```
!***** Output & Log Parameters
```

The following section contains input parameters pertaining to output data.

```
verbosity=2, ! NEXMD output verbosity (0-minimum, 3-maximum)
! [2 for dynamics, 3 for optimization and single-point]
out_data_steps=1, ! Number of steps to write data [1]
out_coords_steps=10, ! Number of steps to write the restart file [10]
out_data_cube=0, ! Write (1) or do not write (0) view files to generate cubes [0]
out_count_init=0, ! Initial count for view files [0]
printTdipole=0, ! For printing the transition dipole moment [0]
printTDM=0, ! For printing the complete transition density matrix [0]
```

- `verbosity` sets the level of printing of NEXMD related outputs.

- `out_data_steps` sets number of steps to write data. For example, if `time_step=0.1` and `out_data_steps=2`, data will be written to output files every 0.2 fs.
- `out_coords_steps` sets the number of data steps to also write a restart file. Note that the rate at which the restart file is written also depends on `out_data_steps`. For example, if `time_step=0.1`, `out_data_steps=2`, and `out_coords_steps=2`, the restart file will be written every 0.4 fs. In general, the restart file is written every `time_step` × `out_data_steps` × `out_coords_steps` fs.
- `out_data_cube` sets whether or not to generate `.DATA` files. These files are later used to generate cube files. If `out_data_cube` is set to `1`, `.DATA` files are generated for every excited state and for every time step. For example, the file `view0003-0007.DATA` refers to the 3$^{\text{rd}}$ time step and 7$^{\text{th}}$ excited state.
- `out_count_init` is the initial value of the iterator used for skipping the writing of the restart file.
- `printTdipole` is set to 1 in order to print the transition dipole moments from the ground state to each excited state considered in a separate file `tdipole.out`.
- `printTDM` is set to 1 for printing the complete transition density matrix instead of only the diagonal with the default value 0. The matrix will be written as a vector for each time step. For trajectory surface hopping dynamics it will be only written for the current state. For Ehrenfest or AIMC dynamics it will be written for all states involved and the second column will label the state. Use with caution: huge amount of data can be produced when writing the complete density matrix.

```
&endmoldyn
```

The following block contains input parameters pertaining to the indexes of the pairs of atoms distances to freeze. If `npc` is set to zero this entire block can be omitted.

```
&pairs
   1 2
   3 4
&endpairs
```

- Between `&pairs` and `&endpairs` are the indexes of the pairs of atoms to freeze. In this example the distances between the first and second, and the third and fourth will be frozen.

The following block contains input parameters pertaining to the indexes of the normal modes to freeze. If `nmc` is set to zero this entire block can be omitted.

```
&modes
   9
   7
&endmodes
```

- Between `&modes` and `&endmodes` are the indexes of modes in the corresponding `nma_modes.out` file to freeze. In this example the modes 9th and 17th will be frozen.

The following blocks contains input parameters pertaining to the coordinates and velocities of the atoms that constitute the molecule being studied.

```
&coord
  6      -7.9798271101     0.6776918081    -0.0532285388
  6      -7.0849928010     1.7602597759     0.0294961792
  6      -5.7058415294     1.5490364812     0.0312760931
  6      -5.2231419594     0.2195333448    -0.0446043010
  6      -6.1050960756    -0.8685564920     0.0220869421
  6      -7.5344099241    -0.6444487634     0.0248126135
  1      -9.0268081830     0.8587716724    -0.0794794940
  1      -7.4774606514     2.7566353436     0.1635393862
  1      -5.0939335779     2.4479885163     0.1481876938
  1      -4.1292016456     0.0999373674    -0.1580811639
  1      -5.6916991654    -1.8878557992     0.1151090966
  1      -8.2838388636    -1.4313798704     0.1051927200
&endcoord

&veloc
     3.3718248255   -5.6032885851   -1.1970845430
     2.5106648755    2.0978837936   -1.0696411897
    -5.9135180273   -3.7505826950    1.1689299883
     7.7194332369    4.8702351843    0.6576546539
    -7.1851218597   -2.0113572464   -0.6329683366
    -1.7276579899    0.3919019235   -0.0257452789
   -17.0279163131    9.9875659542    5.3513734186
    -4.7222747943   18.9640275032   11.9601977632
    10.9539809532   17.0164104392   -9.7113209726
    25.7548696749    2.2116651958   -0.5444198125
   -16.5303708308   -2.3313274630   -3.2147489925
    16.2776787026    2.2582071549    9.3572624705
&endveloc
```

- Between `&coord` and `&endcoord` are coordinates of the atoms in angstroms (Å). The first column identifies each atom with its atomic number. The following three columns are the x, y, and z coordinates, respectively. The coordinates shown above are those of a benzene molecule.

- Between `&veloc` and `&endveloc` are three columns showing velocities of each atom along the x, y, and z axes, respectively. The units of velocity are Å/ps.

The following blocks contains input parameters pertaining to the quantum density matrix associated with the nuclear states (the Tully density matrix for TSH).

```
&coeff
  0.00  0.00
  0.00  0.00
  0.00  0.00
  0.00  0.00
  0.00  0.00
  1.00  0.00
  0.00  0.00
  0.00  0.00
&endcoeff
```

- Between `&coeff` and `&endcoeff` are two columns showing the magnitude and phase of the excited-state coefficients, respectively. The number of rows should be equal to the number of excited states being propagated, `n_exc_states_propagate`. In this example, eight excited-states are being propagated and the system is initially fully excited in the sixth excited-state. This block needs to be the last in the `input.ceon` file for restarting to work properly.

6 Output Files

There are four extensions for data output:

- `.xyz` is only used for coordinates and makes this file readable by several molecular visualization tools.
- `.out` is generally used for output corresponding to a given trajectory. For AIMC type of dynamics a four digit label is added for each one of these files.
- `.dat` is used for ensemble output of AIMC dynamics. The combined information of the data contained in these files and the one in the `coefficient.out` and `gamma.out` files is required for calculating any expectation value when using AIMC. More details on expectation values calculations for AIMC can be found elsewhere [36].
- `.DATA` is used for generating files which can be later converted to `.cube` files read by standard visualization software.

The following section contains a description of all output files. Data depending on time are written to their respective files at a rate which was specified in `out_data_steps` under `Output & Log Parameters`.

- `coeff-n.out` contains the current state as a function of time, as well as the populations of all excited-states being propagated. The first two columns are the current state and time in femtoseconds, respectively. The remaining columns are for excited-state populations from $|1\rangle$ to $|N\rangle$, respectively, where N is the number of states being propagated. The last column is the sum of all populations, which should be approximately 1.0. The first column for Ehrenfest and AIMC type of dynamics has no meaning.
- `coeff-q.out` contains the phase of coefficient corresponding to the electronic wavefunction written as a superposition of adiabatic states. The first column is for time in femtoseconds and the remaining ones correspond to states from $|1\rangle$ to $|N\rangle$, where N is the number of states being propagated.
- `coefficient.out` contains the coefficients corresponding to the electronic wavefunction written as a superposition of adiabatic states. These consists in a complex number for each excited state being propagated as a function of time. The first two columns stand for real and imaginary part. From the third to the fifth column we have: real part without classical action, imaginary part without classical action and classical action. See equations (10 – 14) from [36] for details on this change of variables.
- `cm.out` contains the coordinates of the center of mass with respect to the initial time step.
- `coords.xyz` and `velocity.out` contain the coordinates in angstroms (Å) and velocities in angstroms per femtosecond (Å/ps) of the system as a function of time, respectively.
- `coords_opt.xyz` contains the intermediate coordinates in angstroms (Å) during an energy minimization for `verbosity` greater than 1.

- `cross-steps.out` contains the overlap between the hypothetical crossing states. It is written when a reduction of the quantum step is required. The first column is the time in femtoseconds. The next column is the time at the reduced time step, also in femtoseconds. The third column is either a 1 or 2, indicating a potential or a confirmed trivial crossing, respectively. The next two columns are the states involved in the crossing. The remaining columns contain the overlap of states at the reduced quantum step, the original overlap of states that triggered the trivial crossing routine, and the energy difference between the states, in Hartrees, at the reduced quantum step.

- `dropout.out` contains the labels and times of trajectories that reached a S_0/S_1 energy gap lower than `S0_S1_threshold`. This will stop the simulation for trajectory surface hopping and Ehrenfest dynamics. For AIMC dynamics the simulation will continue through the remaining branches (if any) after a renormalization of the molecular wavefunction.

- `energy-ev.out` contains the kinetic energy, potential energy, and total energy of the system, and their respective changes from the initial time step all in eV.

- `electronic_overlaps.dat` contains the electronic overlaps coefficients propagated according to equation (22) from [36]. Given the symmetry properties of these overlaps: $\langle\phi_I^{(n)}|\phi_J^{(m)}\rangle = \langle\phi_J^{(m)}|\phi_I^{(n)}\rangle$ and $\langle\phi_I^{(n)}|\phi_J^{(n)}\rangle = \delta_{IJ}$, they are only written after the generation of the first clone and only the lower triangular part of the tensor ($n < m$) is written. The first column corresponds to the number of trajectories, the second column corresponds to the time in femtoseconds, the third and fourth column corresponds to the label of trajectories (1 for `0000`, 2 for `0001` and so on), the fifth and sixth columns correspond to the label of excited states and the seventh column corresponds to the electronic overlap value.

- `gamma.out` is only written for AIMC dynamics. It contains the kinetic part of the classical action (see equation 7 from [36] for details).

- `gradients.out` prints a list of real numbers where the first value is the current state, the second value is the time in femtoseconds for trajectory surface hopping dynamics, followed by the gradient components of that state. For Ehrenfest and AIMC dynamics gradient components of all states are included.

- `Heff_last.out` stores the last effective Hamiltonian for AIMC dynamics. This file is only needed for restarting.

- `hessian.out` contains the Hessian matrix (in atomic units), which is only written for nuclear normal mode calculations.

- `hops.out` contains all the successful hops and trivial crossings throughout the trajectory. The first column contains the time, the second column contains a number, where 0 corresponds to a successful hops, 2 corresponds to a trivial crossing, and 1 corresponds to points where a trivial crossing was possible, a point in time that warranted a

reduction in the quantum step. Note that in the case of 1 there is no crossing between states.

- `hops-trial.out` contains attempted and successful hops throughout the trajectory. Hops may be rejected due to energy conservation. This occurs when dispensable nuclear kinetic energy does not exceed the energy barrier between the residing and target surfaces. Therefore, rejected hops can only occur when the target surface lies above the residing surface. The first column is the time at which attempted hops occur in femtoseconds, the second column is the state before the hop or attempt of hop, the third column is the target surface, and the fourth column labels the type of hop, where `0` is a successful hop, `1` is a rejected hop with no decoherence event, and `2` is a rejected hop with a decoherence event. The latter two depend on whether the `decoher_type` was set to `1` or `2` under `Non-Adiabatic Parameters` of the `input.ceon` file. The wavefunction also decoheres at hops of type `0` if `decoher_type=1` or `2`.

- `muab.out` contains the excited-to-excited transition dipoles moments in atomic units. This file is generated when `calcxdens=.true.` during single-point calculations. First and second columns label states $|i\rangle$ and $|j\rangle$, respectively. The third column is the energy difference, $E_{ji} = E_j - E_i$ in eV. The following three columns are excited-to-excited dipoles along the x, y, and z axes, respectively. The last column is the total dipole moment.

- `nacr.out` contains the nonadiabatic coupling vector. The first column is for time in femtoseconds, the second and third columns are for the corresponding states. The remaining columns contain all components of the nonadiabatic coupling vector for all atoms: $x_1, y_1, z_1, x_2, \ldots, z_n$, where n is the number of atoms. The sequence of the atoms corresponds to the same sequence used in the `input.ceon` file between `&coord` and `&endcoord`. For trajectory surface hopping dynamics this only written between the states related at a hop,i.e., for hops indicted by the index 0 in the hops.out. For Ehrenfest and AIMC dynamics it is written at all times and pair of states. Given the symmetry properties of the nonadiabatic coupling vector: $\mathbf{d}_{\alpha\beta} = -\mathbf{d}^*_{\beta\alpha}$, only lower triangular part of the tensor ($\alpha < \beta$) is written.

- `nact.out` contains the non-adiabatic coupling terms between all pairs of states. The non-adiabatic coupling between states $|\alpha\rangle$ and $|\beta\rangle$ is defined as $\dot{\mathbf{R}} \cdot \mathbf{d}_{\alpha\beta}$, where $\dot{\mathbf{R}}$ is velocity and $\mathbf{d}_{\alpha\beta}$ is the non-adiabatic coupling vector between states $|\alpha\rangle$ and $|\beta\rangle$. The first column is time in femtoseconds. The remaining columns are consecutive rows of the non-adiabatic coupling matrix. For example, if 2 states were being propagated, the output would be $\dot{\mathbf{R}} \cdot \mathbf{d}_{\alpha\beta}$ for $\alpha\beta = 11$, 12, 21, and 22, respectively. The diagonal terms of the non-adiabatic coupling are zero and the non-adiabatic coupling vector matrix is anti-Hermitian such that $\mathbf{d} = -\mathbf{d}^\dagger$ or $\mathbf{d}_{\alpha\beta} = -\mathbf{d}^*_{\beta\alpha}$.

- `nma_freq.out` contains the energies of the nuclear normal modes. The first column label the nuclear normal mode and the second contains the corresponding energy (in atomic units). For converting from atomic units to cm^{-1} we can take into account that 1 eV = 8065.54 cm^{-1} and 1 $a.u.$ = 27.211396 eV, therefore 1 $a.u.$ = 219474.60289384 cm^{-1}.

- `nma_modes.out` contains a matrix with the normalized directions for the nuclear normal modes (each column for each nuclear normal mode).

- `nuclear_coeff.dat` contains the coefficients corresponding to the nuclear part of the Multiconfigurational Ehrenfest wavefunction used in AIMC (see equation (1) from [36] for details). These consists of a complex number for each trajectory. The first column contains the number of trajectories and the second column contains the time in femtoseconds. The remaining columns contains the real and imaginary parts of the nuclear coefficients. The ordering is the same as the `.out` files labeling.

- `order.out` lists the diabatic order of the excited states with respect to the initial order at the first time step. The first column is the time in femtoseconds. The next N columns are the N excited states labeled according to the order at the first time step. The remaining N columns indicate, with a 2, whether a trivial crossing has occurred at a specific state at a certain time step, or, with a 1, whether the quantum time step was reduced. More details on trivial crossing identification can be found elsewhere [74]. les

- `pes.out` contains PESs of all states being propagated as a function of time. The first two columns are time in femtoseconds and ground-state energy in eV, respectively. The remaining columns are excited-state energies in eV from $|1\rangle$ to $|N\rangle$, respectively, where N is the number of states being propagated. It is also written for ground state dynamics if the `n_exc_states_propagate` is greater than 1.

- `pop.dat` contains the expectation value of the electronic populations calculated according to equation (25) from [36]. In the current version, this is the only expectation value calculated by NEXMD and should be used as reference when calculating any other expectation value.

- `restart.out` is a file identical to the `input.ceon`, but with the information corresponding to the last restarting point saved. These files are written at a rate of `out_coords_steps` under `Output & Log Parameters`, times the rate of data writing.

- `tdipole.out` contains the transition dipole moment from the ground state to all excited states propagated. It is only written if `printTdipole` is set to 1. It is not written by default. It is also written for ground state dynamics if the `n_exc_states_propagate` is greater than 1. The first column contains the time in femtoseconds, the second column contains the corresponding state, columns from the third to the fifth contain respectively the x, y and z component of the corresponding transition dipole moment and the sixth column contains the summation of the three components squared.

- `temperature.out` contains the temperature of the system as a function of time. The temperature is calculated from the total nuclear kinetic energy. The first column is time in femtoseconds, the second column is the temperature of the system in Kelvin, and the third column is the set temperature of the thermostat in Kelvin.

- `transition-densities.out` contains the transition density matrix written in the atomic orbital basis as a function of time. The number of atomic orbitals for each atom

is 1 for hydrogen and 4 for heavier atoms. The sequence of the atoms corresponds to the same sequence used in the `input.ceon` file between `&coord` and `&endcoord`. If `printTDM` is set to 1 the complete matrix will be written in the same line row after row. This has to be used with caution since huge amounts of data may be generated. By default `printTDM` is set to 0 and only diagonal components are written, which are enough to track the excitation localization in real space. For trajectory surface hopping dynamics it is only written for the residing surface and the first column is for the time in femtoseconds. For Ehrenfest and AIMC dynamics it is written for all surfaces, the first column label the surface and the second column is for time in femtoseconds. The remaining columns are for the transition density matrix components.

- `view_????_????.DATA` are only generated when `out_data_cube` is set to one. The first four digit label denotes time step while the second four digit label denotes the corresponding excited state. These files contains several blocks with information about the number of atoms, the number of orbitals, the number of occupied molecular orbitals, the number of eigenvectors printed, the atomic coordinates (in Å) and the diagonal of the transition density. From this information we can generate `.cube` files containing information about the transition density localization in real space. These `.cube` files can be read by standard visualization tools. In order to generate the `.cube` files we can use the `correr` script located in the `visualization` directory.

Table 1 summarizes the conditions for which each file is generated. `verbosity` here refers to the one in the `&moldyn` block from the `input.ceon` file.

File name	verbosity	GS	tsh	mf	aimc	Other
coeff-n.out	1	No	Yes	Yes	Yes	-
coeff-q.out	2	No	Yes	Yes	Yes	Always for aimc
coefficient.out	0	No	Yes	Yes	Yes	-
cm.out	3	Yes	Yes	Yes	Yes	-
coords.xyz	0	Yes	Yes	Yes	Yes	Minimization final output
coords_opt.xyz	2	-	-	-	-	Only for minimization
cross-steps.out	2	No	Yes	Yes	Yes	-
dropout.out	0	No	No	No	Yes	-
energy-ev.out	0	Yes	Yes	Yes	Yes	-
electronic_overlaps.dat	0	No	No	No	Yes	-
gamma.out	0	No	No	No	Yes	-
gradients.out	3	Yes	Yes	Yes	Yes	-
Heff_last.out	0	No	No	No	Yes	For restarting aimc
hessian.out	-	-	-	-	-	Only for normal modes
hops.out	0	No	Yes	Yes	Yes	-
hops-trial.out	2	No	Yes	No	No	-
muab.out	-	-	-	-	-	Only for calcxdens=.true.
nacr.out	2	No	Yes	Yes	Yes	Always for mf and aimc
nact.out	1	No	Yes	Yes	Yes	-
nma_freq.out	-	-	-	-	-	Only for normal modes
nma_modes.out	-	-	-	-	-	Only for normal modes
nuclear_coeff.dat	0	No	No	No	Yes	-
order.out	2	No	Yes	Yes	Yes	-
pes.out	1	Yes	Yes	Yes	Yes	For GS only if n_exc_states_propagate > 0
pop.dat	0	No	No	No	Yes	-
restart.out	0	Yes	Yes	Yes	Yes	-
tdipole.out	0	Yes	Yes	Yes	Yes	Only if printTdipole > 0, for GS only if n_exc_states_propagate > 0
temperature.out	0	Yes	Yes	Yes	Yes	-
transition-densities.out	1	No	Yes	Yes	Yes	-
velocity.out	0	Yes	Yes	Yes	Yes	-
view_????_????.DATA	0	No	Yes	Yes	Yes	Only if out_data_cube $= 1$

Table 1: NEXMD conditioning for writing each output file.

7 Restarting Simulations

Non-adiabatic trajectories may not finish within the user-defined number of classical steps for several reasons such as (1) the computing system may have a time-limit that is less than the time to complete a trajectory, (2) the defined wall-time may not be long enough to complete the trajectory, or (3) a problem in the computing system may cause jobs to stop before completion. In any case, trajectories may be restarted from the last time-step, as long as the last excited-state, nuclear coordinates and velocities, and quantum coefficients are known. These quantities are available in the file `restart.out`, which has exactly the same structure as an `input.ceon` but with the data corresponding to the restarting point. The file `restart.out` is created automatically by NEXMD. If it is corrupted for some reason, for instance if the program stopped exactly while writing it and is incomplete, it might be reconstructed from the last step. For restarting we can launch NEXMD exactly as we did the first time. NEXMD will look for a `restart.out` file and, if it is there, NEXMD will read data from it and continue from where it stopped. Any exceeding lines in the output files, if any, will be deleted. For the simulation to start from the beginning there can't be any file with the name `restart.out` (or `restart_????.out` for AIMC) in the directory.

It is also important to use a three spaces indentation in the `input.ceon` file for the variables in the `restart.out` file to be updated.

If for some reason the user prefers restarting trajectories from a step different from the one stored in the `restart.out` file, it can be done by editing the `restart.out` file and setting the corresponding initial time, simulation steps, coordinates, velocities and electronic coefficients. This should be done with caution because NEXMD will remove any exceeding lines from the output files when restarting. For AIMC dynamics, all restarts should be synchronized.

If any of the output files is corrupted, i.e. it has less lines than what it should for restarting at time set in the input, the simulation will stopped ant the corresponding corrupted file will be notified in the `md.out` file.

8 Input examples

In this section several examples of `input.ceon` files are given to be used as templates. I general, when adapting these for a new molecule, the `coords` block and the number of atoms `natom` need to be updated. Other input variables and blocks can be updated or added depending on the desired application. Regarding indentation, it is important to use three spaces for variables to be updated in the `restart.out` file. More templates examples can be found in the `tests` directory.

8.1 Optimization

Optimization can be done either in the ground state or in a given excited state. The following input can be used as a template for optimization in the ground state. For optimizing in a given excited state, the input variables `exc_state_init` and `n_exc_states_propagate` need to be set to the corresponding state.

```
&qmmm
   !***** Geometry Optimization
   maxcyc=300, ! Number of cycles for geometry optimization [0]
   ntpr=1, ! Print results every ntpr cycles [1]
   grms_tol=1.0d-2, ! Tolerance in eV/A (derivatives) [1.0d-2]

   !***** Normal Modes Analysis
   do_nm=0, ! Flag for doing Normal Modes Analysis [0]
   deltaX=1.0d-4, ! Displacement for the Hessian calculation, A [1.0d-4]

   !***** Ground-State and Output Parameters
   qm_theory='AM1', ! Integral type, check Amber's SQM for more options [AM1]
   scfconv=1.0d-6, ! Ground-state SCF convergence criteria, eV [1.0d-6]
   verbosity=0, ! QM/MM output verbosity (0-minimum, 5-maximum)
   ! [1 for dynamics and optimization, 5 for others]
   printdipole=1, ! (0) Unrelaxed transitions, (1) Unrelaxed transitions plus
   ! total molecular, or (2) Unrelaxed/relaxed transitions plus
   ! total molecular [1 for dynamics, 2 for optimization and single-point]
   printbondorders=0, ! (0) No or (1) Yes [0]
   ! *** UNDER DEVELOPMENT, DO NOT USE ***
   density_predict=0, ! (0) None, (1) Reversible MD,
   ! or (2) XL-BOMD [0] *** ALL ARE UNDER DEVELOPMENT, DO NOT USE ***
   itrmax=3000, ! Max SCF iterations for ground state
   ! (negative to ignore convergence) [300]

   !***** Excited-State Parameters
   exst_method=1, ! CIS (1) or RPA (2) [1]
   dav_guess=1, ! Restart Davidson from (0) Scratch, (1) Previous,
   ! or (2) XL-BOMD [1] *** (2) IS UNDER DEVELOPMENT, DO NOT USE ***
   ftol0=1.0d-6, ! Acceptance tolerance (|emin-eold|) [1.0d-5]
   ftol1=1.0d-6, ! Acceptance tolerance for residual norm [1.0d-5]
   ! *** UNDER DEVELOPMENT, DO NOT USE ***
```

```
   dav_maxcyc=200, ! Max cycles for Davidson diagonalization
   ! (negative to ignore convergence) [100]
   printcharges=0, ! Print (1) or do not print (0) Mulliken charges of QM atoms [0]
   calcxdens=.false., ! Print (.true.) or do not print (.false.)
   ! excited-to-excited transition dipole moments [.false.]

   !***** Solvent Models and External Electric Fields
   solvent_model=0, ! (0) None, (1) Linear response, (2) Vertical excitation,
   ! or (3) State-specific [0]
   potential_type=1, ! (1) COSMO or (2) Onsager [1]
   onsager_radius=2, ! Onsager radius, A (system dependent) [2]
   ceps=10, ! Dielectric constant, unitless [10]
   linmixparam=1, ! Linear mixing parameter for vertical excitation
   ! or state-specific SCF calculation [1]
   cosmo_scf_ftol=1.0d-5, ! Vertical excitation or state-specific
   ! SCF tolerance, eV [1.0d-5]
   doZ=.false., ! Use relaxed (.true.) or unrelaxed (.false) density for
   ! vertical excitation or state-specific COSMO or Onsager [.false.]
   index_of_refraction=100, ! Dielectric constant for linear response
   ! solvent in excited-state, unitless [100] *** UNDER DEVELOPMENT, DO NOT USE ***
   EF=0, ! (0) None or (1) Electric field in ground- and excited-state [0]
   Ex=0, ! Electric field vector X, eV/A [0]
   Ey=0, ! Electric field vector Y, eV/A [0]
   Ez=0, ! Electric field vector Z, eV/A [0]
&endqmmm

&moldyn
   !***** General Parameters
   natoms=12, ! Number of atoms
   ! (must be equal to the number of atoms in system)
   rnd_seed=272184, ! seed for the random number generator
   bo_dynamics_flag=0, ! (0) Non-BO or (1) BO [1]
   exc_state_init=0, ! initial excited state (0 - ground state) [0]
   n_exc_states_propagate=0, ! Number of excited states [0]

   !***** Dynamics Parameters
   time_init=0.0, ! Initial time, fs [0.0]
   time_step=0.5, ! Time step, fs [0.1]
   n_class_steps=0, ! Number of classical steps [1]
   n_quant_steps=4, ! Number of quantum steps for each classical step [4]
   moldyn_deriv_flag=1, ! (0) None, (1) Analytical, or (2) Numerical [1]
   num_deriv_step=1.0d-3, ! Displacement for numerical derivatives, A [1.0d-3]

   !***** Non-Adiabatic Parameters
   decoher_type=2, ! Type of decoherence: Reinitialize (0) Never,
   ! (1) At successful hops, (2) At successful plus frustrated hops...
   ! (3) Persico/Granucci, or (4) Truhlar [2]
```

```
   ! *** (3) AND (4) ARE UNDER DEVELOPMENT, DO NOT USE ***
   decoher_e0=0.0, ! Decoherence parameter E0, Hartrees [0.1]
   ! (only for decoher_type = 3 or 4) *** UNDER DEVELOPMENT, DO NOT USE ***
   decoher_c=0.0, ! Decoherence parameter C, unitless [0.1]
   ! (only for decoher_type = 3 or 4) *** UNDER DEVELOPMENT, DO NOT USE ***
   iredpot=1, !For state reduction
   nstates=3, !How many states to reduce
   dotrivial=1, ! Do unavoided (trivial) crossing routine (1) or not (0) [1]
   quant_step_reduction_factor=2.5d-2, ! Quantum step reduction factor [2.5d-2]
   NAMD_type='tsh', ! Type of molecular dynamics ('tsh','mf' or 'aimc') ['tsh']
   AIMC_dclone_2=0.01,
   nclones0=0, ! Clones count for 'aimc' (must be declared here for restarting 'aimc') [0]

   !***** Thermostat Parameters
   therm_type=1, ! Thermostat type: (0) Newtonian, (1) Langevin,
   ! or (2) Berendsen [1] *** (2) IS UNDER DEVELOPMENT, DO NOT USE ***
   therm_temperature=300, ! Thermostat temperature, K [300]
   therm_friction=20, ! Thermostat friction coefficient, 1/ps [20]
   berendsen_relax_const=0.4, ! Bath relaxation constant for Berendsen
   ! thermostat, ps [0.4] *** UNDER DEVELOPMENT, DO NOT USE ***
   heating=0, ! Equilibrated (0) or heating (1) [0]
   ! *** UNDER DEVELOPMENT, DO NOT USE ***
   heating_steps_per_degree=100, ! Number of steps per degree
   ! during heating [100] *** UNDER DEVELOPMENT, DO NOT USE ***

   !***** Output & Log Parameters
   verbosity=3, ! NEXMD output verbosity (0-minimum, 3-maximum)
   ! [2 for dynamics, 3 for optimization and single-point]
   out_data_steps=1, ! Number of steps to write data [1]
   out_coords_steps=20, ! Number of steps to write the restart file [10]
   out_data_cube=0, ! Write (1) or do not write (0) view files to generate cubes [0]
   out_count_init=0, ! Initial count for view files [0]
   printTdipole=1, ! Flag for printing transition dipole moments [0]
   nmc=0, ! Number of normal modes to freeze
&endmoldyn

&coord
     6  -35.8948664799 9.2886259308 10.0544203477
     6  -36.3696092323 8.1368720916 9.4874253256
     6  -34.5363309682 9.5695703477 9.9555021126
     6  -35.4923084434 7.1597227488 8.9838424098
     6  -33.7126430908 8.6974582688 9.3044121015
     6  -34.1833404232 7.4945484254 8.7620358687
     1  -36.5854647341 9.9399336196 10.5379614226
     1  -37.4508240693 8.0646070979 9.4360164204
     1  -34.0700572906 10.5024940235 10.5000652405
     1  -35.9507481749 6.2012351180 8.7652116130
```

```
     1  -33.4383733642 6.8263742785 8.2469555967
     1  -32.6549106837 8.8726068085 9.1230185816
&endcoord

&veloc
    -0.8053143393  -4.1878833732  -3.6313775837
     1.0294675393   2.7627100918   4.4006100054
    -4.4870427177  -1.1534294697  -3.2087971501
    -2.8524435043   0.6142857400  -1.4239264979
     1.2815455099   4.6430462174   3.4289638426
     7.5572197841  -6.9819959784  -3.4951206895
     0.6887252580  17.5438701930  37.7476956202
   -13.3392762907  21.5658675938   4.2738139260
   -10.8410626110  -8.7990139589  -0.2271823192
   -27.6435417594  20.1653179965   7.1606228597
    28.5550727505  -2.3586673777   2.9304364808
     1.8799376326   3.5691627518  -4.6863835611
&endveloc

&coeff
0.0 0.0
0.0 0.0
1.0 0.0
0.0 0.0
&endcoeff
```

8.2 Normal modes calculation

Normal modes calculations can be done either in the ground state or in a given excited state. The following input can be used as a template for normal modes calculations in the ground state. For calculating normal modes in a given excited state, the input variables `exc_state_init` and `n_exc_states_propagate` need to be set to the corresponding state.

```
&qmmm
   !***** Geometry Optimization
   maxcyc=0, ! Number of cycles for geometry optimization [0]
   ntpr=1, ! Print results every ntpr cycles [1]
   grms_tol=1.0d-2, ! Tolerance in eV/A (derivatives) [1.0d-2]

   !***** Normal Modes Analysis
   do_nm=1, ! Flag for doing Normal Modes Analysis [0]
   deltaX=1.0d-4, ! Displacement for the Hessian calculation, A [1.0d-4]

   !***** Ground-State and Output Parameters
   qm_theory='AM1', ! Integral type, check Amber's SQM for more options [AM1]
   scfconv=1.0d-6, ! Ground-state SCF convergence criteria, eV [1.0d-6]
   verbosity=0, ! QM/MM output verbosity (0-minimum, 5-maximum)
```

```
   ! [1 for dynamics and optimization, 5 for others]
   printdipole=1, ! (0) Unrelaxed transitions, (1) Unrelaxed transitions plus
   ! total molecular, or (2) Unrelaxed/relaxed transitions plus
   ! total molecular [1 for dynamics, 2 for optimization and single-point]
   printbondorders=0, ! (0) No or (1) Yes [0]
   ! *** UNDER DEVELOPMENT, DO NOT USE ***
   density_predict=0, ! (0) None, (1) Reversible MD,
   ! or (2) XL-BOMD [0] *** ALL ARE UNDER DEVELOPMENT, DO NOT USE ***
   itrmax=3000, ! Max SCF iterations for ground state
   ! (negative to ignore convergence) [300]

   !***** Excited-State Parameters
   exst_method=1, ! CIS (1) or RPA (2) [1]
   dav_guess=1, ! Restart Davidson from (0) Scratch, (1) Previous,
   ! or (2) XL-BOMD [1] *** (2) IS UNDER DEVELOPMENT, DO NOT USE ***
   ftol0=1.0d-6, ! Acceptance tolerance (|emin-eold|) [1.0d-5]
   ftol1=1.0d-6, ! Acceptance tolerance for residual norm [1.0d-5]
   ! *** UNDER DEVELOPMENT, DO NOT USE ***
   dav_maxcyc=200, ! Max cycles for Davidson diagonalization
   ! (negative to ignore convergence) [100]
   printcharges=0, ! Print (1) or do not print (0) Mulliken charges of QM atoms [0]
   calcxdens=.false., ! Print (.true.) or do not print (.false.)
   ! excited-to-excited transition dipole moments [.false.]

   !***** Solvent Models and External Electric Fields
   solvent_model=0, ! (0) None, (1) Linear response, (2) Vertical excitation,
   ! or (3) State-specific [0]
   potential_type=1, ! (1) COSMO or (2) Onsager [1]
   onsager_radius=2, ! Onsager radius, A (system dependent) [2]
   ceps=10, ! Dielectric constant, unitless [10]
   linmixparam=1, ! Linear mixing parameter for vertical excitation
   ! or state-specific SCF calculation [1]
   cosmo_scf_ftol=1.0d-5, ! Vertical excitation or state-specific
   ! SCF tolerance, eV [1.0d-5]
   doZ=.false., ! Use relaxed (.true.) or unrelaxed (.false) density for
   ! vertical excitation or state-specific COSMO or Onsager [.false.]
   index_of_refraction=100, ! Dielectric constant for linear response
   ! solvent in excited-state, unitless [100] *** UNDER DEVELOPMENT, DO NOT USE ***
   EF=0, ! (0) None or (1) Electric field in ground- and excited-state [0]
   Ex=0, ! Electric field vector X, eV/A [0]
   Ey=0, ! Electric field vector Y, eV/A [0]
   Ez=0, ! Electric field vector Z, eV/A [0]
&endqmmm

&moldyn
   !***** General Parameters
   natoms=12, ! Number of atoms
```

```
! (must be equal to the number of atoms in system)
rnd_seed=272184, ! seed for the random number generator
bo_dynamics_flag=0, ! (0) Non-BO or (1) BO [1]
exc_state_init=0, ! initial excited state (0 - ground state) [0]
n_exc_states_propagate=0, ! Number of excited states [0]

!***** Dynamics Parameters
time_init=0.0, ! Initial time, fs [0.0]
time_step=0.5, ! Time step, fs [0.1]
n_class_steps=0, ! Number of classical steps [1]
n_quant_steps=4, ! Number of quantum steps for each classical step [4]
moldyn_deriv_flag=1, ! (0) None, (1) Analytical, or (2) Numerical [1]
num_deriv_step=1.0d-3, ! Displacement for numerical derivatives, A [1.0d-3]

!***** Non-Adiabatic Parameters
decoher_type=2, ! Type of decoherence: Reinitialize (0) Never,
! (1) At successful hops, (2) At successful plus frustrated hops...
! (3) Persico/Granucci, or (4) Truhlar [2]
! *** (3) AND (4) ARE UNDER DEVELOPMENT, DO NOT USE ***
decoher_e0=0.0, ! Decoherence parameter E0, Hartrees [0.1]
! (only for decoher_type = 3 or 4) *** UNDER DEVELOPMENT, DO NOT USE ***
decoher_c=0.0, ! Decoherence parameter C, unitless [0.1]
! (only for decoher_type = 3 or 4) *** UNDER DEVELOPMENT, DO NOT USE ***
iredpot=1, !For state reduction
nstates=3, !How many states to reduce
dotrivial=1, ! Do unavoided (trivial) crossing routine (1) or not (0) [1]
quant_step_reduction_factor=2.5d-2, ! Quantum step reduction factor [2.5d-2]
NAMD_type='tsh', ! Type of molecular dynamics ('tsh','mf' or 'aimc') ['tsh']
AIMC_dclone_2=0.01,
nclones0=0, ! Clones count for 'aimc' (must be declared here for restarting 'aimc') [0]

!***** Thermostat Parameters
therm_type=1, ! Thermostat type: (0) Newtonian, (1) Langevin,
! or (2) Berendsen [1] *** (2) IS UNDER DEVELOPMENT, DO NOT USE ***
therm_temperature=300, ! Thermostat temperature, K [300]
therm_friction=20, ! Thermostat friction coefficient, 1/ps [20]
berendsen_relax_const=0.4, ! Bath relaxation constant for Berendsen
! thermostat, ps [0.4] *** UNDER DEVELOPMENT, DO NOT USE ***
heating=0, ! Equilibrated (0) or heating (1) [0]
! *** UNDER DEVELOPMENT, DO NOT USE ***
heating_steps_per_degree=100, ! Number of steps per degree
! during heating [100] *** UNDER DEVELOPMENT, DO NOT USE ***

!***** Output & Log Parameters
verbosity=3, ! NEXMD output verbosity (0-minimum, 3-maximum)
! [2 for dynamics, 3 for optimization and single-point]
out_data_steps=1, ! Number of steps to write data [1]
```

```
   out_coords_steps=20, ! Number of steps to write the restart file [10]
   out_data_cube=0, ! Write (1) or do not write (0) view files to generate cubes [0]
   out_count_init=0, ! Initial count for view files [0]
   printTdipole=1, ! Flag for printing transition dipole moments [0]
   nmc=0, ! Number of normal modes to freeze
&endmoldyn

&coord
  6  -35.8814924202   9.3092594343  10.0495128321
  6  -36.3910500896   8.1166355772   9.5354312245
  6  -34.5185102313   9.5871903867   9.9468424959
  6  -35.5378223612   7.2036938721   8.9152879523
  6  -33.6659884351   8.6765369590   9.3218621227
  6  -34.1749609477   7.4831882448   8.8098398674
  1  -36.5543473101  10.0319039396  10.5333213508
  1  -37.4647947247   7.8951824921   9.6216408339
  1  -34.1159154220  10.5231172920  10.3604264029
  1  -35.9394844117   6.2650023728   8.5067583920
  1  -33.5017536922   6.7614452042   8.3251412092
  1  -32.5933569087   8.9008929842   9.2308023570
&endcoord

&veloc
    -0.8053143393  -4.1878833732  -3.6313775837
     1.0294675393   2.7627100918   4.4006100054
    -4.4870427177  -1.1534294697  -3.2087971501
    -2.8524435043   0.6142857400  -1.4239264979
     1.2815455099   4.6430462174   3.4289638426
     7.5572197841  -6.9819959784  -3.4951206895
     0.6887252580  17.5438701930  37.7476956202
   -13.3392762907  21.5658675938   4.2738139260
   -10.8410626110  -8.7990139589  -0.2271823192
   -27.6435417594  20.1653179965   7.1606228597
    28.5550727505  -2.3586673777   2.9304364808
     1.8799376326   3.5691627518  -4.6863835611
&endveloc

&coeff
0.0 0.0
0.0 0.0
1.0 0.0
0.0 0.0
&endcoeff
```

8.3 TSH dynamics

The following input can be used as a template for TSH dynamics. The initial excited state `exc_state_init` has to be set to a value greater than zero and lower or equal to the number of excited states considered `n_exc_states_propagate`. When working with an ensemble it is also advised to use different random seed `rnd_seed` values. If the initial distribution of quantum amplitudes in the `coeff` block is set to a pure state, it has to match the initial excited state `exc_state_init`.

```
&qmmm
   !***** Geometry Optimization
   maxcyc=0, ! Number of cycles for geometry optimization [0]
   ntpr=1, ! Print results every ntpr cycles [1]
   grms_tol=1.0d-2, ! Tolerance in eV/A (derivatives) [1.0d-2]

   !***** Normal Modes Analysis
   do_nm=0, ! Flag for doing Normal Modes Analysis [0]
   deltaX=1.0d-4, ! Displacement for the Hessian calculation, A [1.0d-4]

   !***** Ground-State and Output Parameters
   qm_theory='AM1', ! Integral type, check Amber's SQM for more options [AM1]
   scfconv=1.0d-6, ! Ground-state SCF convergence criteria, eV [1.0d-6]
   verbosity=2, ! QM/MM output verbosity (0-minimum, 5-maximum)
   ! [1 for dynamics and optimization, 5 for others]
   printdipole=1, ! (0) Unrelaxed transitions, (1) Unrelaxed transitions plus
   ! total molecular, or (2) Unrelaxed/relaxed transitions plus
   ! total molecular [1 for dynamics, 2 for optimization and single-point]
   printbondorders=0, ! (0) No or (1) Yes [0]
   ! *** UNDER DEVELOPMENT, DO NOT USE ***
   density_predict=0, ! (0) None, (1) Reversible MD,
   ! or (2) XL-BOMD [0] *** ALL ARE UNDER DEVELOPMENT, DO NOT USE ***
   itrmax=3000, ! Max SCF iterations for ground state
   ! (negative to ignore convergence) [300]

   !***** Excited-State Parameters
   exst_method=1, ! CIS (1) or RPA (2) [1]
   dav_guess=1, ! Restart Davidson from (0) Scratch, (1) Previous,
   ! or (2) XL-BOMD [1] *** (2) IS UNDER DEVELOPMENT, DO NOT USE ***
   ftol0=1.0d-6, ! Acceptance tolerance (|emin-eold|) [1.0d-5]
   ftol1=1.0d-6, ! Acceptance tolerance for residual norm [1.0d-5]
   ! *** UNDER DEVELOPMENT, DO NOT USE ***
   dav_maxcyc=200, ! Max cycles for Davidson diagonalization
   ! (negative to ignore convergence) [100]
   printcharges=0, ! Print (1) or do not print (0) Mulliken charges of QM atoms [0]
   calcxdens=.false., ! Print (.true.) or do not print (.false.)
   ! excited-to-excited transition dipole moments [.false.]

   !***** Solvent Models and External Electric Fields
```

```
   solvent_model=0, ! (0) None, (1) Linear response, (2) Vertical excitation,
   ! or (3) State-specific [0]
   potential_type=1, ! (1) COSMO or (2) Onsager [1]
   onsager_radius=2, ! Onsager radius, A (system dependent) [2]
   ceps=10, ! Dielectric constant, unitless [10]
   linmixparam=1, ! Linear mixing parameter for vertical excitation
   ! or state-specific SCF calculation [1]
   cosmo_scf_ftol=1.0d-5, ! Vertical excitation or state-specific
   ! SCF tolerance, eV [1.0d-5]
   doZ=.false., ! Use relaxed (.true.) or unrelaxed (.false) density for
   ! vertical excitation or state-specific COSMO or Onsager [.false.]
   index_of_refraction=100, ! Dielectric constant for linear response
   ! solvent in excited-state, unitless [100] *** UNDER DEVELOPMENT, DO NOT USE ***
   EF=0, ! (0) None or (1) Electric field in ground- and excited-state [0]
   Ex=0, ! Electric field vector X, eV/A [0]
   Ey=0, ! Electric field vector Y, eV/A [0]
   Ez=0, ! Electric field vector Z, eV/A [0]
&endqmmm

&moldyn
   !***** General Parameters
   natoms=12, ! Number of atoms
   ! (must be equal to the number of atoms in system)
   rnd_seed=272184, ! seed for the random number generator
   bo_dynamics_flag=0, ! (0) Non-BO or (1) BO [1]
   exc_state_init=3, ! initial excited state (0 - ground state) [0]
   n_exc_states_propagate=4, ! Number of excited states [0]

   !***** Dynamics Parameters
   time_init=0.0, ! Initial time, fs [0.0]
   time_step=0.1, ! Time step, fs [0.1]
   n_class_steps=200, ! Number of classical steps [1]
   n_quant_steps=4, ! Number of quantum steps for each classical step [4]
   moldyn_deriv_flag=1, ! (0) None, (1) Analytical, or (2) Numerical [1]
   num_deriv_step=1.0d-3, ! Displacement for numerical derivatives, A [1.0d-3]

   !***** Non-Adiabatic Parameters
   decoher_type=2, ! Type of decoherence: Reinitialize (0) Never,
   ! (1) At successful hops, (2) At successful plus frustrated hops...
   ! (3) Persico/Granucci, or (4) Truhlar [2]
   ! *** (3) AND (4) ARE UNDER DEVELOPMENT, DO NOT USE ***
   decoher_e0=0.0, ! Decoherence parameter E0, Hartrees [0.1]
   ! (only for decoher_type = 3 or 4) *** UNDER DEVELOPMENT, DO NOT USE ***
   decoher_c=0.0, ! Decoherence parameter C, unitless [0.1]
   ! (only for decoher_type = 3 or 4) *** UNDER DEVELOPMENT, DO NOT USE ***
   iredpot=1, !For state reduction
   nstates=3, !How many states to reduce
```

```
   dotrivial=1, ! Do unavoided (trivial) crossing routine (1) or not (0) [1]
   quant_step_reduction_factor=2.5d-2, ! Quantum step reduction factor [2.5d-2]
   NAMD_type='tsh', ! Type of molecular dynamics ('tsh','mf' or 'aimc') ['tsh']
   AIMC_dclone_2=0.01,
   nclones0=0, ! Clones count for 'aimc' (must be declared here for restarting 'aimc') [0]

   !***** Thermostat Parameters
   therm_type=0, ! Thermostat type: (0) Newtonian, (1) Langevin,
   ! or (2) Berendsen [1] *** (2) IS UNDER DEVELOPMENT, DO NOT USE ***
   therm_temperature=300, ! Thermostat temperature, K [300]
   therm_friction=20, ! Thermostat friction coefficient, 1/ps [20]
   berendsen_relax_const=0.4, ! Bath relaxation constant for Berendsen
   ! thermostat, ps [0.4] *** UNDER DEVELOPMENT, DO NOT USE ***
   heating=0, ! Equilibrated (0) or heating (1) [0]
   ! *** UNDER DEVELOPMENT, DO NOT USE ***
   heating_steps_per_degree=100, ! Number of steps per degree
   ! during heating [100] *** UNDER DEVELOPMENT, DO NOT USE ***

   !***** Output & Log Parameters
   verbosity=3, ! NEXMD output verbosity (0-minimum, 3-maximum)
   ! [2 for dynamics, 3 for optimization and single-point]
   out_data_steps=1, ! Number of steps to write data [1]
   out_coords_steps=20, ! Number of steps to write the restart file [10]
   out_data_cube=0, ! Write (1) or do not write (0) view files to generate cubes [0]
   out_count_init=0, ! Initial count for view files [0]
   printTdipole=1, ! Flag for printing transition dipole moments [0]
   nmc=0, ! Number of normal modes to freeze
&endmoldyn

&coord
     6  -35.8948664799 9.2886259308 10.0544203477
     6  -36.3696092323 8.1368720916 9.4874253256
     6  -34.5363309682 9.5695703477 9.9555021126
     6  -35.4923084434 7.1597227488 8.9838424098
     6  -33.7126430908 8.6974582688 9.3044121015
     6  -34.1833404232 7.4945484254 8.7620358687
     1  -36.5854647341 9.9399336196 10.5379614226
     1  -37.4508240693 8.0646070979 9.4360164204
     1  -34.0700572906 10.5024940235 10.5000652405
     1  -35.9507481749 6.2012351180 8.7652116130
     1  -33.4383733642 6.8263742785 8.2469555967
     1  -32.6549106837 8.8726068085 9.1230185816
&endcoord

&veloc
    -0.8053143393  -4.1878833732  -3.6313775837
     1.0294675393   2.7627100918   4.4006100054
```

```
    -4.4870427177  -1.1534294697  -3.2087971501
    -2.8524435043   0.6142857400  -1.4239264979
     1.2815455099   4.6430462174   3.4289638426
     7.5572197841  -6.9819959784  -3.4951206895
     0.6887252580  17.5438701930  37.7476956202
   -13.3392762907  21.5658675938   4.2738139260
   -10.8410626110  -8.7990139589  -0.2271823192
   -27.6435417594  20.1653179965   7.1606228597
    28.5550727505  -2.3586673777   2.9304364808
     1.8799376326   3.5691627518  -4.6863835611
&endveloc

&coeff
0.0 0.0
0.0 0.0
1.0 0.0
0.0 0.0
&endcoeff
```

8.4 Mean field Ehrenfest dynamics

The following input can be used as a template for mean field Ehrenfest dynamics. The number of excited states considered `n_exc_states_propagate` has to be greater than zero.

```
&qmmm
   !***** Geometry Optimization
   maxcyc=0, ! Number of cycles for geometry optimization [0]
   ntpr=1, ! Print results every ntpr cycles [1]
   grms_tol=1.0d-2, ! Tolerance in eV/A (derivatives) [1.0d-2]

   !***** Normal Modes Analysis
   do_nm=0, ! Flag for doing Normal Modes Analysis [0]
   deltaX=1.0d-4, ! Displacement for the Hessian calculation, A [1.0d-4]

   !***** Ground-State and Output Parameters
   qm_theory='AM1', ! Integral type, check Amber's SQM for more options [AM1]
   scfconv=1.0d-6, ! Ground-state SCF convergence criteria, eV [1.0d-6]
   verbosity=2, ! QM/MM output verbosity (0-minimum, 5-maximum)
   ! [1 for dynamics and optimization, 5 for others]
   printdipole=1, ! (0) Unrelaxed transitions, (1) Unrelaxed transitions plus
   ! total molecular, or (2) Unrelaxed/relaxed transitions plus
   ! total molecular [1 for dynamics, 2 for optimization and single-point]
   printbondorders=0, ! (0) No or (1) Yes [0]
   ! *** UNDER DEVELOPMENT, DO NOT USE ***
   density_predict=0, ! (0) None, (1) Reversible MD,
   ! or (2) XL-BOMD [0] *** ALL ARE UNDER DEVELOPMENT, DO NOT USE ***
   itrmax=3000, ! Max SCF iterations for ground state
   ! (negative to ignore convergence) [300]
```

```
   !***** Excited-State Parameters
   exst_method=1, ! CIS (1) or RPA (2) [1]
   dav_guess=1, ! Restart Davidson from (0) Scratch, (1) Previous,
   ! or (2) XL-BOMD [1] *** (2) IS UNDER DEVELOPMENT, DO NOT USE ***
   ftol0=1.0d-6, ! Acceptance tolerance (|emin-eold|) [1.0d-5]
   ftol1=1.0d-6, ! Acceptance tolerance for residual norm [1.0d-5]
   ! *** UNDER DEVELOPMENT, DO NOT USE ***
   dav_maxcyc=200, ! Max cycles for Davidson diagonalization
   ! (negative to ignore convergence) [100]
   printcharges=0, ! Print (1) or do not print (0) Mulliken charges of QM atoms [0]
   calcxdens=.false., ! Print (.true.) or do not print (.false.)
   ! excited-to-excited transition dipole moments [.false.]

   !***** Solvent Models and External Electric Fields
   solvent_model=0, ! (0) None, (1) Linear response, (2) Vertical excitation,
   ! or (3) State-specific [0]
   potential_type=1, ! (1) COSMO or (2) Onsager [1]
   onsager_radius=2, ! Onsager radius, A (system dependent) [2]
   ceps=10, ! Dielectric constant, unitless [10]
   linmixparam=1, ! Linear mixing parameter for vertical excitation
   ! or state-specific SCF calculation [1]
   cosmo_scf_ftol=1.0d-5, ! Vertical excitation or state-specific
   ! SCF tolerance, eV [1.0d-5]
   doZ=.false., ! Use relaxed (.true.) or unrelaxed (.false) density for
   ! vertical excitation or state-specific COSMO or Onsager [.false.]
   index_of_refraction=100, ! Dielectric constant for linear response
   ! solvent in excited-state, unitless [100] *** UNDER DEVELOPMENT, DO NOT USE ***
   EF=0, ! (0) None or (1) Electric field in ground- and excited-state [0]
   Ex=0, ! Electric field vector X, eV/A [0]
   Ey=0, ! Electric field vector Y, eV/A [0]
   Ez=0, ! Electric field vector Z, eV/A [0]
&endqmmm

&moldyn
   !***** General Parameters
   natoms=12, ! Number of atoms
   ! (must be equal to the number of atoms in system)
   rnd_seed=272184, ! seed for the random number generator
   bo_dynamics_flag=0, ! (0) Non-BO or (1) BO [1]
   exc_state_init=3, ! initial excited state (0 - ground state) [0]
   n_exc_states_propagate=4, ! Number of excited states [0]

   !***** Dynamics Parameters
   time_init=0.0, ! Initial time, fs [0.0]
   time_step=0.05, ! Time step, fs [0.1]
   n_class_steps=400, ! Number of classical steps [1]
```

```
   n_quant_steps=4, ! Number of quantum steps for each classical step [4]
   moldyn_deriv_flag=1, ! (0) None, (1) Analytical, or (2) Numerical [1]
   num_deriv_step=1.0d-3, ! Displacement for numerical derivatives, A [1.0d-3]

   !***** Non-Adiabatic Parameters
   decoher_type=2, ! Type of decoherence: Reinitialize (0) Never,
   ! (1) At successful hops, (2) At successful plus frustrated hops...
   ! (3) Persico/Granucci, or (4) Truhlar [2]
   ! *** (3) AND (4) ARE UNDER DEVELOPMENT, DO NOT USE ***
   decoher_e0=0.0, ! Decoherence parameter E0, Hartrees [0.1]
   ! (only for decoher_type = 3 or 4) *** UNDER DEVELOPMENT, DO NOT USE ***
   decoher_c=0.0, ! Decoherence parameter C, unitless [0.1]
   ! (only for decoher_type = 3 or 4) *** UNDER DEVELOPMENT, DO NOT USE ***
   iredpot=1, !For state reduction
   nstates=3, !How many states to reduce
   dotrivial=1, ! Do unavoided (trivial) crossing routine (1) or not (0) [1]
   quant_step_reduction_factor=2.5d-2, ! Quantum step reduction factor [2.5d-2]
   NAMD_type='mf', ! Type of molecular dynamics ('tsh','mf' or 'aimc') ['tsh']
   AIMC_dclone_2=0.01,
   nclones0=0, ! Clones count for 'aimc' (must be declared here for restarting 'aimc') [0]

   !***** Thermostat Parameters
   therm_type=0, ! Thermostat type: (0) Newtonian, (1) Langevin,
   ! or (2) Berendsen [1] *** (2) IS UNDER DEVELOPMENT, DO NOT USE ***
   therm_temperature=300, ! Thermostat temperature, K [300]
   therm_friction=20, ! Thermostat friction coefficient, 1/ps [20]
   berendsen_relax_const=0.4, ! Bath relaxation constant for Berendsen
   ! thermostat, ps [0.4] *** UNDER DEVELOPMENT, DO NOT USE ***
   heating=0, ! Equilibrated (0) or heating (1) [0]
   ! *** UNDER DEVELOPMENT, DO NOT USE ***
   heating_steps_per_degree=100, ! Number of steps per degree
   ! during heating [100] *** UNDER DEVELOPMENT, DO NOT USE ***

   !***** Output & Log Parameters
   verbosity=3, ! NEXMD output verbosity (0-minimum, 3-maximum)
   ! [2 for dynamics, 3 for optimization and single-point]
   out_data_steps=1, ! Number of steps to write data [1]
   out_coords_steps=20, ! Number of steps to write the restart file [10]
   out_data_cube=0, ! Write (1) or do not write (0) view files to generate cubes [0]
   out_count_init=0, ! Initial count for view files [0]
   printTdipole=1, ! Flag for printing transition dipole moments [0]
   nmc=0, ! Number of normal modes to freeze
&endmoldyn

&coord
     6  -35.8948664799 9.2886259308 10.0544203477
     6  -36.3696092323 8.1368720916 9.4874253256
```

```
     6  -34.5363309682 9.5695703477 9.9555021126
     6  -35.4923084434 7.1597227488 8.9838424098
     6  -33.7126430908 8.6974582688 9.3044121015
     6  -34.1833404232 7.4945484254 8.7620358687
     1  -36.5854647341 9.9399336196 10.5379614226
     1  -37.4508240693 8.0646070979 9.4360164204
     1  -34.0700572906 10.5024940235 10.5000652405
     1  -35.9507481749 6.2012351180 8.7652116130
     1  -33.4383733642 6.8263742785 8.2469555967
     1  -32.6549106837 8.8726068085 9.1230185816
&endcoord

&veloc
    -0.8053143393  -4.1878833732  -3.6313775837
     1.0294675393   2.7627100918   4.4006100054
    -4.4870427177  -1.1534294697  -3.2087971501
    -2.8524435043   0.6142857400  -1.4239264979
     1.2815455099   4.6430462174   3.4289638426
     7.5572197841  -6.9819959784  -3.4951206895
     0.6887252580  17.5438701930  37.7476956202
   -13.3392762907  21.5658675938   4.2738139260
   -10.8410626110  -8.7990139589  -0.2271823192
   -27.6435417594  20.1653179965   7.1606228597
    28.5550727505  -2.3586673777   2.9304364808
     1.8799376326   3.5691627518  -4.6863835611
&endveloc

&coeff
0.0 0.0
0.0 0.0
1.0 0.0
0.0 0.0
&endcoeff
```

8.5 AIMC dynamics

The following input can be used as a template for AIMC dynamics. The number of excited states considered `n_exc_states_propagate` has to be greater than zero.

```
&qmmm
   !***** Geometry Optimization
   maxcyc=0, ! Number of cycles for geometry optimization [0]
   ntpr=1, ! Print results every ntpr cycles [1]
   grms_tol=1.0d-2, ! Tolerance in eV/A (derivatives) [1.0d-2]

   !***** Normal Modes Analysis
   do_nm=0, ! Flag for doing Normal Modes Analysis [0]
   deltaX=1.0d-4, ! Displacement for the Hessian calculation, A [1.0d-4]
```

```
!***** Ground-State and Output Parameters
qm_theory='AM1', ! Integral type, check Amber's SQM for more options [AM1]
scfconv=1.0d-6, ! Ground-state SCF convergence criteria, eV [1.0d-6]
verbosity=2, ! QM/MM output verbosity (0-minimum, 5-maximum)
! [1 for dynamics and optimization, 5 for others]
printdipole=1, ! (0) Unrelaxed transitions, (1) Unrelaxed transitions plus
! total molecular, or (2) Unrelaxed/relaxed transitions plus
! total molecular [1 for dynamics, 2 for optimization and single-point]
printbondorders=0, ! (0) No or (1) Yes [0]
! *** UNDER DEVELOPMENT, DO NOT USE ***
density_predict=0, ! (0) None, (1) Reversible MD,
! or (2) XL-BOMD [0] *** ALL ARE UNDER DEVELOPMENT, DO NOT USE ***
itrmax=3000, ! Max SCF iterations for ground state
! (negative to ignore convergence) [300]

!***** Excited-State Parameters
exst_method=1, ! CIS (1) or RPA (2) [1]
dav_guess=1, ! Restart Davidson from (0) Scratch, (1) Previous,
! or (2) XL-BOMD [1] *** (2) IS UNDER DEVELOPMENT, DO NOT USE ***
ftol0=1.0d-6, ! Acceptance tolerance (|emin-eold|) [1.0d-5]
ftol1=1.0d-6, ! Acceptance tolerance for residual norm [1.0d-5]
! *** UNDER DEVELOPMENT, DO NOT USE ***
dav_maxcyc=200, ! Max cycles for Davidson diagonalization
! (negative to ignore convergence) [100]
printcharges=0, ! Print (1) or do not print (0) Mulliken charges of QM atoms [0]
calcxdens=.false., ! Print (.true.) or do not print (.false.)
! excited-to-excited transition dipole moments [.false.]

!***** Solvent Models and External Electric Fields
solvent_model=0, ! (0) None, (1) Linear response, (2) Vertical excitation,
! or (3) State-specific [0]
potential_type=1, ! (1) COSMO or (2) Onsager [1]
onsager_radius=2, ! Onsager radius, A (system dependent) [2]
ceps=10, ! Dielectric constant, unitless [10]
linmixparam=1, ! Linear mixing parameter for vertical excitation
! or state-specific SCF calculation [1]
cosmo_scf_ftol=1.0d-5, ! Vertical excitation or state-specific
! SCF tolerance, eV [1.0d-5]
doZ=.false., ! Use relaxed (.true.) or unrelaxed (.false) density for
! vertical excitation or state-specific COSMO or Onsager [.false.]
index_of_refraction=100, ! Dielectric constant for linear response
! solvent in excited-state, unitless [100] *** UNDER DEVELOPMENT, DO NOT USE ***
EF=0, ! (0) None or (1) Electric field in ground- and excited-state [0]
Ex=0, ! Electric field vector X, eV/A [0]
Ey=0, ! Electric field vector Y, eV/A [0]
Ez=0, ! Electric field vector Z, eV/A [0]
```

```
&endqmmm

&moldyn
   !***** General Parameters
   natoms=12, ! Number of atoms
   ! (must be equal to the number of atoms in system)
   rnd_seed=272184, ! seed for the random number generator
   bo_dynamics_flag=0, ! (0) Non-BO or (1) BO [1]
   exc_state_init=3, ! initial excited state (0 - ground state) [0]
   n_exc_states_propagate=4, ! Number of excited states [0]

   !***** Dynamics Parameters
   time_init=0.0, ! Initial time, fs [0.0]
   time_step=0.05, ! Time step, fs [0.1]
   n_class_steps=400, ! Number of classical steps [1]
   n_quant_steps=4, ! Number of quantum steps for each classical step [4]
   moldyn_deriv_flag=1, ! (0) None, (1) Analytical, or (2) Numerical [1]
   num_deriv_step=1.0d-3, ! Displacement for numerical derivatives, A [1.0d-3]

   !***** Non-Adiabatic Parameters
   decoher_type=2, ! Type of decoherence: Reinitialize (0) Never,
   ! (1) At successful hops, (2) At successful plus frustrated hops...
   ! (3) Persico/Granucci, or (4) Truhlar [2]
   ! *** (3) AND (4) ARE UNDER DEVELOPMENT, DO NOT USE ***
   decoher_e0=0.0, ! Decoherence parameter E0, Hartrees [0.1]
   ! (only for decoher_type = 3 or 4) *** UNDER DEVELOPMENT, DO NOT USE ***
   decoher_c=0.0, ! Decoherence parameter C, unitless [0.1]
   ! (only for decoher_type = 3 or 4) *** UNDER DEVELOPMENT, DO NOT USE ***
   iredpot=1, !For state reduction
   nstates=3, !How many states to reduce
   dotrivial=1, ! Do unavoided (trivial) crossing routine (1) or not (0) [1]
   quant_step_reduction_factor=2.5d-2, ! Quantum step reduction factor [2.5d-2]
   NAMD_type='aimc', ! Type of molecular dynamics ('tsh','mf' or 'aimc') ['tsh']
   AIMC_dclone_2=0.01,
   nclones0=0, ! Clones count for 'aimc' (must be declared here for restarting 'aimc') [0]

   !***** Thermostat Parameters
   therm_type=0, ! Thermostat type: (0) Newtonian, (1) Langevin,
   ! or (2) Berendsen [1] *** (2) IS UNDER DEVELOPMENT, DO NOT USE ***
   therm_temperature=300, ! Thermostat temperature, K [300]
   therm_friction=20, ! Thermostat friction coefficient, 1/ps [20]
   berendsen_relax_const=0.4, ! Bath relaxation constant for Berendsen
   ! thermostat, ps [0.4] *** UNDER DEVELOPMENT, DO NOT USE ***
   heating=0, ! Equilibrated (0) or heating (1) [0]
   ! *** UNDER DEVELOPMENT, DO NOT USE ***
   heating_steps_per_degree=100, ! Number of steps per degree
   ! during heating [100] *** UNDER DEVELOPMENT, DO NOT USE ***
```

```
   !***** Output & Log Parameters
   verbosity=3, ! NEXMD output verbosity (0-minimum, 3-maximum)
   ! [2 for dynamics, 3 for optimization and single-point]
   out_data_steps=1, ! Number of steps to write data [1]
   out_coords_steps=20, ! Number of steps to write the restart file [10]
   out_data_cube=0, ! Write (1) or do not write (0) view files to generate cubes [0]
   out_count_init=0, ! Initial count for view files [0]
   printTdipole=1, ! Flag for printing transition dipole moments [0]
   nmc=0, ! Number of normal modes to freeze
&endmoldyn

&coord
     6  -35.8948664799 9.2886259308 10.0544203477
     6  -36.3696092323 8.1368720916 9.4874253256
     6  -34.5363309682 9.5695703477 9.9555021126
     6  -35.4923084434 7.1597227488 8.9838424098
     6  -33.7126430908 8.6974582688 9.3044121015
     6  -34.1833404232 7.4945484254 8.7620358687
     1  -36.5854647341 9.9399336196 10.5379614226
     1  -37.4508240693 8.0646070979 9.4360164204
     1  -34.0700572906 10.5024940235 10.5000652405
     1  -35.9507481749 6.2012351180 8.7652116130
     1  -33.4383733642 6.8263742785 8.2469555967
     1  -32.6549106837 8.8726068085 9.1230185816
&endcoord

&veloc
    -0.8053143393  -4.1878833732  -3.6313775837
     1.0294675393   2.7627100918   4.4006100054
    -4.4870427177  -1.1534294697  -3.2087971501
    -2.8524435043   0.6142857400  -1.4239264979
     1.2815455099   4.6430462174   3.4289638426
     7.5572197841  -6.9819959784  -3.4951206895
     0.6887252580  17.5438701930  37.7476956202
   -13.3392762907  21.5658675938   4.2738139260
   -10.8410626110  -8.7990139589  -0.2271823192
   -27.6435417594  20.1653179965   7.1606228597
    28.5550727505  -2.3586673777   2.9304364808
     1.8799376326   3.5691627518  -4.6863835611
&endveloc

&coeff
0.0 0.0
0.0 0.0
1.0 0.0
0.0 0.0
```

```
&endcoeff
```

A Contributors

Contributors are listed in the alphabetical order of their last name.

A.1 Principal Investigators

- Sebastian Fernandez-Alberti (UNQui)
  sfalberti@gmail.com
- Adrian E. Roitberg (UF)
  roitberg@ufl.edu
- Sergei Tretiak (LANL)
  serg@lanl.gov

A.2 Developers

- Josiah A. Bjorgaard (LANL)
  jbjorgaard@lanl.gov
- Victor Manuel Freixas Lemus (UCI)
  vfreixas@uci.edu
- Walter Malone (TU)
  wmalone@tuskegee.edu
- Benjamin Nebgen (LANL)
  bnebgen@lanl.gov
- Lázaro Hassiel Negrín-Yuvero (UNQui)
  lhny1990@gmail.com
- Tammie R. Nelson (LANL)
  tammien@lanl.gov
- Andrew E. Sifain (LANL & USC)
  sifain@usc.edu
- Huajing (Wilson) Song (LANL)
  songhw@lanl.gov
- Dustin A. Tracy (UF)
  dtracy@ufl.edu
- Kirill Velizhanin (LANL)
  kirill@lanl.gov
- Alexander J. White (LANL)
  alwhite@lanl.gov
- Yu Zhang (LANL)
  zhy@lanl.gov

B Observable calculation for AIMC

In this section we will provide insights on observable calculations for AIMC. AIMC is a sampling technique developed for the MultiConfigurational Ehrenfest (MCE) approach and implemented in the NEXMD software package [36]. In the MCE approach, the system is described by means of a molecular wave function $|\Psi\rangle$ which is a linear superposition of configurations $|\varphi_n\rangle$:

$$|\Psi\rangle = \sum_n |\varphi_n\rangle \,. \tag{1}$$

Each configuration has its own nuclear and electronic parts. The nuclear part is given by Gaussian functions $|\chi_n\rangle$ centered on mean field Ehrenfest trajectories [36], while the electronic part is given by a linear superposition of the adiabatic solutions $\phi_I^{(n)}$ for the nuclear center. Therefore, the total wave function can be written as:

$$|\Psi\rangle = \sum_n c_n |\chi_n\rangle \sum_I a_I^{(n)} |\phi_I^{(n)}\rangle \,, \tag{2}$$

where the magnitudes c_n and $a_I^{(n)}$ are complex numbers propagated on the fly according to the time dependent Schrödinger equation and stored in the files `nuclear-coeff.dat` and `coefficient_????.out` respectively.

B.1 Observable over the electronic subspace

There are two mean observable types according to the subspace where they are defined. Let us start with those defined over the electronic subspace. One of the most important magnitudes for analyzing non-adiabatic dynamics simulations is the population of states. For trajectory surface hopping dynamics, these are the fraction of the ensemble driven by a given surface at a given time and are referred as classical populations. For AIMC we need to introduce the population operator:

$$\hat{P}_K = |\phi_K^{(n)}\rangle\langle\phi_K^{(n)}| \,. \tag{3}$$

The corresponding expectation value is given by:

$$P_K = \langle \hat{P}_K \rangle = \sum_{n,m} c_m^* c_n \langle \chi_m | \chi_n \rangle \sum_{I,J} \left(a_J^{(m)}\right)^* a_I^{(n)} \langle \phi_J^{(m)} | \hat{P}_K | \phi_I^{(n)} \rangle \,. \tag{4}$$

Taking into account the orthonormality of the adiabatic basis for a given configuration n, we can get to [36]:

$$\hat{P}_K = \sum_{n,m} c_m^* c_n \langle \chi_m | \chi_n \rangle \left(a_K^{(m)}\right)^* \sum_I a_I^{(n)} \langle \phi_K^{(m)} | \phi_I^{(n)} \rangle \,. \tag{5}$$

The nuclear overlaps $\langle \chi_m | \chi_n \rangle$ can be calculated analytically [75] as a function of nuclear coordinates and velocities stored respectively in `coords.xyz` and `velocity.out`. The electronic overlaps $\langle \phi_K^{(m)} | \phi_I^{(n)} \rangle$ are propagated on the fly [36] and are stored in the file `electronic_overlaps.dat`.

Is worth noting that equation (3) is defined over n while it could have been defined over m leading to a different analytical result. When using a complete electronic basis these two options are exactly the same. Since we are using a truncated basis there might be some small numerical differences and sometime a half summation over n and m is used for (3), leading to:

$$\langle \hat{P}_K \rangle = \frac{1}{2} \sum_{n,m} c_m^* c_n \langle \chi_m | \chi_n \rangle \sum_I \left[\left(a_K^{(m)} \right)^* a_I^{(n)} \langle \phi_K^{(m)} | \phi_I^{(n)} \rangle + \left(a_I^{(m)} \right)^* a_K^{(n)} \langle \phi_I^{(m)} | \phi_K^{(n)} \rangle \right] \tag{6}$$

The same reasoning applies for the closure relation used in the derivation of other observable.

One important property of $\hat{P}_K$ is that summation over K is the norm of the molecular wave function, which should always be 1. Therefore, it is strongly advised that the implementation of any other observable is preceded by the calculation of the norm to verify that all complex phases fit in correctly.

Some other examples of observable over the electronic subspace are:

- *Transition densities.* Diagonal terms of the transition density matrix are commonly used to track the excitation localization in real space. The fraction of TD corresponding to state I localized over a given fragment X of the molecule for a given configuration can be calculated as [36]:

$$\rho_{I,X}^{(n)} = \frac{\sum_{i \in X} \left(\rho_I^{(n)} \right)_{i,i}^2}{\sum_i \left(\rho_I^{(n)} \right)_{i,i}^2} , \tag{7}$$

  where the diagonal terms $\left(\rho_I^{(n)} \right)_{i,i}$ are stored in the files `transition-densities.out`. The corresponding operator $\hat{\rho}_X$ is such that [29]:

$$\hat{\rho}_X |\phi_I^{(n)}\rangle = \rho_{I,X}^{(n)} |\phi_I^{(n)}\rangle , \tag{8}$$

  and the corresponding expectation value is:

$$\langle \rho_X \rangle = \sum_{n,m} c_m^* c_n \langle \chi_m | \chi_n \rangle \sum_{I,J} \left(a_J^{(m)} \right)^* a_I^{(n)} \rho_{I,X}^{(n)} \langle \phi_J^{(m)} | \phi_I^{(n)} \rangle . \tag{9}$$

- *Coherences.* Coherences can be of relevance for the calculation of spectroscopic signals when the polarization is assumed to be constant [28, 29] and can be calculated as the non-diagonal version of the population operator $\hat{P}_{KL}$:

$$\hat{P}_{KL} = |\phi_K^{(n)}\rangle \langle \phi_L^{(n)}| . \tag{10}$$

  The corresponding expectation value (symmetrised) is given by [28]:

$$\langle \hat{P}_{KL} \rangle = \frac{1}{2} \sum_{n,m} c_m^* c_n \langle \chi_m | \chi_n \rangle \sum_I \left[\left(a_K^{(m)} \right)^* a_I^{(n)} \langle \phi_L^{(m)} | \phi_I^{(n)} \rangle + \left(a_I^{(m)} \right)^* a_L^{(n)} \langle \phi_I^{(m)} | \phi_K^{(n)} \rangle \right] \tag{11}$$

- *Potential energy.* The operator corresponding to the potential energy is such that:

$$\hat{V}|\phi_I^{(n)}\rangle = V_I^{(n)}|\phi_I^{(n)}\rangle, \tag{12}$$

where $V_I^{(n)}$ is written in the files `pes_????.out`. The corresponding expectation value (symmetrised) is given by:

$$\langle\hat{V}\rangle = \frac{1}{2}\sum_{n,m} c_m^* c_n \langle\chi_m|\chi_n\rangle \sum_{I,J}\left(a_J^{(m)}\right)^* a_I^{(n)}\langle\phi_J^{(m)}|\phi_I^{(n)}\rangle\left(V_I^{(n)} + V_J^{(m)}\right) \tag{13}$$

B.2 Observable over the nuclear subspace

In a similar way to the case of observable over the electronic subspace, the one over the nuclear subspace are calculated by introducing fist the corresponding operator. Let us see some examples:

- *Distance between atoms.* The expectation value $\langle\hat{R}_{ij}\rangle$ of the distance between atoms i and j of the molecule is given by:

$$\langle\hat{R}_{ij}\rangle = \sum_{m,n} c_m^* c_n \langle\chi_n||R_i - R_j||\chi_n\rangle \sum_{IJ}\left(a_J^{(m)}\right)^* a_I^{(n)}\langle\phi_J^{(m)}|\phi_I^{(n)}\rangle\,. \tag{14}$$

In order to calculate the nuclear matrix elements needed, we can introduce the approximation [36]:

$$\langle\hat{R}_{ij}\rangle \approx \left|\sum_{n,m} c_m^* c_n \left(\langle\chi_m|R_i|\chi_n\rangle - \langle\chi_m|R_j|\chi_n\rangle\right)\right| \sum_{IJ}\left(a_J^{(m)}\right)^* a_I^{(n)}\langle\phi_J^{(m)}|\phi_I^{(n)}\rangle\,, \tag{15}$$

which, taking into account the analytical expressions reported in [75] for the nuclear matrix overlaps, can be reduced to [36]:

$$\langle\hat{R}_{ij}\rangle \approx \sum_{n,m} c_m^* c_n \langle\chi_m|\chi_n\rangle \left|\frac{R_i^{(n)} + R_i^{(m)}}{2} - \frac{R_j^{(n)} + R_j^{(m)}}{2}\right| \sum_{IJ}\left(a_J^{(m)}\right)^* a_I^{(n)}\langle\phi_J^{(m)}|\phi_I^{(n)}\rangle\,, \tag{16}$$

- *Potential energy of a given state.* The energy of a given state V_K might be useful during the analysis of the non-adiabatic molecular dynamics. It does not depend on the time solution of the Schrödinger equation for the electronic subsystem. Therefore, it can be calculated by introducing an operator $\hat{V}_K$ over the nuclear subspace such that:

$$\hat{V}_K|\chi_n\rangle = V_K^{(n)}|\chi_n\rangle\,, \tag{17}$$

where the energies $V_K^{(n)}$ of the state K for the configuration n are stored in the `pes_????.out` files. The corresponding expectation value is given by:

$$\langle\hat{V}_K\rangle = \frac{1}{2}\sum_{n,m} c_m^* c_n \langle\chi_m|\chi_n\rangle \left(V_K^{(m)} + V_K^{(n)}\right) \sum_{I,J}\left(a_J^{(m)}\right)^* a_I^{(n)}\langle\phi_J^{(m)}|\phi_I^{(n)}\rangle\,. \tag{18}$$

This idea can be extended for the calculation of any other electronic magnitude belonging to a given adiabatic state, such as the fraction of the transition density of state K localized over the fragment X of the molecule.

References

[1] W. Malone et al., Journal of Chemical Theory and Computation **16**, 5771 (2020).

[2] T. R. Nelson et al., Chemical reviews **120**, 2215 (2020).

[3] D. Polli et al., Nature **467**, 440 (2010).

[4] G. D. Scholes et al., Nature **543**, 647 (2017).

[5] A. Salehi, X. Fu, D.-H. Shin, and F. So, Advanced Functional Materials **29**, 1808803 (2019).

[6] N. Wang, X. Tong, Q. Burlingame, J. Yu, and S. R. Forrest, Solar energy materials and solar cells **125**, 170 (2014).

[7] P. Wolfer et al., Journal of Materials Chemistry C **2**, 71 (2014).

[8] A. Zhugayevych and S. Tretiak, Annual Review of Physical Chemistry **66**, 305 (2015).

[9] H. Sirringhaus et al., Science **290**, 2123 (2000).

[10] K. E. Lee, J. U. Lee, D. G. Seong, M.-K. Um, and W. Lee, The Journal of Physical Chemistry C **120**, 23172 (2016).

[11] B. Satishkumar et al., Nature Nanotechnology **2**, 560 (2007).

[12] E. G. Maksimov et al., Scientific reports **9**, 8937 (2019).

[13] H. Li, J. Wang, X. Wang, H. Lin, and F. Li, ACS applied materials & interfaces **11**, 16958 (2019).

[14] A. Wilson et al., Proceedings of the National Academy of Sciences **105**, 12075 (2008).

[15] N. K. Singer, P. A. Sánchez-Murcia, M. Ernst, and L. González, Angewandte Chemie **134**, e202205198 (2022).

[16] N. A. Romero and D. A. Nicewicz, Chemical reviews **116**, 10075 (2016).

[17] J.-L. Brédas, J. E. Norton, J. Cornil, and V. Coropceanu, Accounts of chemical research **42**, 1691 (2009).

[18] B. Schmidt-Hansberg et al., ACS nano **5**, 8579 (2011).

[19] S. Tretiak, A. Saxena, R. Martin, and A. Bishop, Physical review letters **89**, 097402 (2002).

[20] Y. Cao, I. D. Parker, G. Yu, C. Zhang, and A. J. Heeger, Nature **397**, 414 (1999).

[21] M. Kasha, Discussions of the Faraday society **9**, 14 (1950).

[22] M. A. Robb, M. Garavelli, M. Olivucci, and F. Bernardi, Reviews in computational chemistry , 87 (2000).

[23] Y. Jiang and J. McNeill, Chemical reviews **117**, 838 (2017).

[24] M. Huix-Rotllant, H. Tamura, and I. Burghardt, The journal of physical chemistry letters **6**, 1702 (2015).

[25] J.-L. Brédas, D. Beljonne, V. Coropceanu, and J. Cornil, Chemical reviews **104**, 4971 (2004).

[26] L. Adamska et al., Nano letters **14**, 6539 (2014).

[27] V. Freixas, S. Tretiak, D. Makhov, D. Shalashilin, and S. Fernandez-Alberti, The Journal of Physical Chemistry B **124**, 3992 (2020).

[28] D. Keefer et al., Chemical science **12**, 5286 (2021).

[29] V. M. Freixas, D. Keefer, S. Tretiak, S. Fernandez-Alberti, and S. Mukamel, Chemical Science **13**, 6373 (2022).

[30] S. Tretiak and S. Mukamel, Chemical reviews **102**, 3171 (2002).

[31] D. J. Thouless, *The quantum mechanics of many-body systems*, Courier Corporation, 2014.

[32] P. Jorgensen, Annual Review of Physical Chemistry **26**, 359 (1975).

[33] A. McLachlan and M. Ball, Reviews of Modern Physics **36**, 844 (1964).

[34] J. C. Tully, J. Chem. Phys. **93**, 1061 (1990).

[35] S. Fernandez-Alberti, D. V. Makhov, S. Tretiak, and D. V. Shalashilin, Physical Chemistry Chemical Physics **18**, 10028 (2016).

[36] V. M. Freixas, S. Fernandez-Alberti, D. V. Makhov, S. Tretiak, and D. Shalashilin, Physical Chemistry Chemical Physics **20**, 17762 (2018).

[37] J. Clark, T. Nelson, S. Tretiak, G. Cirmi, and G. Lanzani, Nature Physics **8**, 225 (2012).

[38] N. Oldani, S. Tretiak, G. Bazan, and S. Fernandez-Alberti, Energy Environ. Sci. **7**, 1175 (2014).

[39] D. Ondarse-Alvarez, N. Oldani, S. Tretiak, and S. Fernandez-Alberti, J. Phys. Chem. A **118**, 10742 (2014).

[40] L. Alfonso Hernandez et al., J. Phys. Chem. Lett. **7**, 4936 (2016).

[41] A. E. Sifain et al., Journal of chemical theory and computation **14**, 3955 (2018).

[42] D. Ondarse-Alvarez, T. Nelson, J. M. Lupton, S. Tretiak, and S. Fernandez-Alberti, The journal of physical chemistry letters **9**, 7123 (2018).

[43] A. Mukazhanova et al., Journal of Materials Chemistry C **11**, 5297 (2023).

[44] D. Ondarse-Alvarez et al., Phys. Chem. Chem. Phys. **18**, 25080 (2016).

[45] J. F. Galindo et al., J. Am. Chem. Soc. **137**, 11637 (2015).

[46] S. Fernandez-Alberti, A. E. Roitberg, V. D. Kleiman, T. Nelson, and S. Tretiak, J. Chem. Phys. **137**, 22A526 (2012).

[47] M. A. Soler, A. E. Roitberg, T. Nelson, S. Tretiak, and S. Fernandez-Alberti, J. Phys. Chem. A **116**, 9802 (2012).

[48] D. Ondarse-Alvarez et al., Physical Chemistry Chemical Physics **20**, 29648 (2018).

[49] V. M. Freixas et al., The Journal of Chemical Physics **150** (2019).

[50] V. Bonilla, V. M. Freixas, S. Fernandez-Alberti, and J. F. Galindo, Physical Chemistry Chemical Physics **25**, 12097 (2023).

[51] R. Franklin-Mergarejo, T. Nelson, S. Tretiak, and S. Fernandez-Alberti, Phys. Chem. Chem. Phys. **19**, 9478 (2017).

[52] N. Oldani, S. Doorn, S. Tretiak, and S. Fernandez-Alberti, Physical Chemistry Chemical Physics **19**, 30914 (2017).

[53] B. Rodriguez-Hernandez et al., The Journal of Physical Chemistry C **122**, 16639 (2018).

[54] R. Franklin-Mergarejo, D. O. Alvarez, S. Tretiak, and S. Fernandez-Alberti, Sci. Rep. **6** (2016).

[55] V. M. Freixas, S. Tretiak, and S. Fernandez-Alberti, The Journal of Physical Chemistry Letters **13**, 8495 (2022).

[56] H. Negrin-Yuvero et al., The Journal of Physical Chemistry C **127**, 5449 (2023).

[57] W. P. Bricker et al., Sci. Rep. **5** (2015).

[58] P. M. Shenai, S. Fernandez-Alberti, W. P. Bricker, S. Tretiak, and Y. Zhao, J. Phys. Chem. B **120**, 49 (2015).

[59] F. zheng, S. Fernandez-Alberti, S. Tretiak, and Y. Zhao, J. Phys. Chem. B (2017).

[60] M. T. Greenfield et al., J. Phys. Chem. A **119**, 4846 (2015).

[61] T. Nelson et al., J. Phys. Chem. A **120**, 519 (2016).

[62] L. Lystrom, Y. Zhang, S. Tretiak, and T. Nelson, The Journal of Physical Chemistry A **122**, 6055 (2018).

[63] M. A. Soler, T. Nelson, A. E. Roitberg, S. Tretiak, and S. Fernandez-Alberti, J. Phys. Chem. A **118**, 10372 (2014).

[64] Amber Reference Manual: http://ambermd.org/doc12/Amber17.pdf (2017).

[65] J. A. Bjorgaard, V. Kuzmenko, K. Velizhanin, and S. Tretiak, J. Chem. Phys. **142**, 044103 (2015).

[66] J. A. Bjorgaard, K. A. Velizhanin, and S. Tretiak, J. Chem. Phys. **143**, 054305 (2015).

[67] W. M. Haynes, *CRC Handbook of Chemistry and Physics*, CRC press, 2014.

[68] V. M. Freixas et al., The Journal of Physical Chemistry Letters **12**, 2970 (2021).

[69] H. C. Andersen, Journal of computational Physics **52**, 24 (1983).

[70] H. Negrin-Yuvero et al., Journal of Chemical Theory and Computation **16**, 7289 (2020).

[71] S. Fernandez-Alberti, A. E. Roitberg, T. Nelson, and S. Tretiak, J. Chem. Phys. **137**, 014512 (2012).

[72] T. Nelson, A. Naumov, S. Fernandez-Alberti, and S. Tretiak, Chem. Phys. **481**, 84 (2016).

[73] V. M. F. Lemus, D. Keefer, S. Tretiak, S. Fernandez-Alberti, and S. Mukamel, Chemical Science (2022).

[74] T. Nelson, S. Fernandez-Alberti, A. E. Roitberg, and S. Tretiak, Chem. Phys. Lett. **590**, 208 (2013).

[75] D. V. Makhov, W. J. Glover, T. J. Martinez, and D. V. Shalashilin, The Journal of chemical physics **141**, 054110 (2014).