SHARC4.0: Surface Hopping Including Arbitrary Couplings

Tutorial

González group
Institute of Theoretical Chemistry
University of Vienna, Austria

Vienna, July 20, 2026

Contents

1 Before you Start

In this tutorial, the steps necessary to perform non-adiabatic dynamics with the SHARC dynamics suite are explained. The tutorial consists of four tutorial sections.

The first part contains the full tutorial presenting a complete dynamics study including initial condition generation, ensemble management, and trajectory analysis (with plotting of the results and a brief discussion). **Keep in mind that the results shown in this tutorial can differ numerically from the results you see on your screen. This is probably not an error but simply due to differences in numerical libraries and random number generators. The tutorial should still be easy to follow.** This tutorial will use the interface to OPENMOLCAS. **Note that OPENMOLCAS is freely available.**

The second part is a collection of more specialized tutorials showing some advanced usage aspects in detail.

The third part contains general information for what is different if using the other quantum chemistry interfaces besides the one for MOLCAS.

The fourth part is a quick tutorial, just presenting the files necessary to run a single SHARC trajectory, without initial condition generation, ensemble management, or analysis.

Additionally, we suggest that you can watch the following instructional videos from the Cyber Training Workshop 2022 by Alexey Akimov from the University af Buffalo, NY. These material there consists of:

- Slides about surface hopping theory and how to set up an example surface hopping project
- Slides about nonadiabatic dynamics with TDDFT and with LVC models
- Slides about analyzing the example surface hopping project
- Three task sheets (setup with OpenMolcas, setup with TDDFT, analysis of trajectories). Note that these were adapted from the SHARC tutorial and we recommend that you follow the new tutorial rather than these older task sheets.
- Three Zoom recordings (here) of presenting the slides and task sheets live.

Finally, we recommend these general tutorial papers on how to run nonadiabatic dynamics:

- Best practices for nonadiabatic molecular dynamics simulations

1.1 Description of the model system

The task of the tutorial is to simulate the excited-state dynamics of the methaniminium cation ($CH_2NH_2^+$) after excitation to the bright excited state.

The employed quantum chemistry method will be CASSCF(6,4)/cc-pVDZ, using the OPENMOLCAS program package. This level of theory is not sufficient for a serious scientific investigation, and was chosen on purpose for this tutorial because it is fast and (relatively) easy to use. For this molecule, the method has a certain chance to lead to problematic trajectories; this is also on purpose so that such trajectories can be discussed.

The main goal of the tutorial is to simulate the excitation of the methaniminium cation to the first bright excited state, which is of $\pi\pi^*$ character. Other states of $\pi\pi^*$ and $\sigma\pi^*$ character are close in energy to the bright state. By vibrational motion, population in the $\pi\pi^*$ state can be transferred to the $\pi\sigma^*$ states (internal conversion), after which a bond can dissociate. Alternatively, $CH_2NH_2^+$ might relax to the ground state through a conical intersection involving the torsion around the double bond Although intersystem crossing is one focus area of SHARC in the methaniminium cation no intersystem crossing is expected to occur. Still, we will include a number of triplet states in the trajectories to show how this can be done.

In the following, an overview over the level of theory and the initial geometry are given:

Ab initio level of theory for pyrrole.

Charge	+1
Program	MOLCAS
Method	SA-CASSCF(6,4)
Basis set	cc-pVDZ
Number of states	4 Singlets, 3 Triplets

```
6
Starting  geometry  for  CH2NH2+
C +0.000000 +0.000000 +0.000000
N +0.000000 +0.000000 +1.350000
H +0.943102 +0.000000 -0.544500
H +0.943102 +0.000000 +1.879500
H -0.943102 +0.000000 +1.879500
H -0.943102 +0.000000 -0.544500
```

2 Full Tutorial

This tutorial presents all steps of an excited-state dynamics study. These steps include preparation tasks like optimization and frequency calculation, generation of the Wigner distribution of initial conditions, calculation of the excited states of these initial conditions, and initial state selection. Subsequently, it will be shown how to setup the input files for an ensemble of trajectories and how they are executed. Furthermore, the tutorial presents some trajectory analysis steps, including plotting of energies, populations, etc. of a single trajectory, calculation of internal coordinates, or of ensemble populations.

2.1 Important

Beyond the core dynamics program, the SHARC suite contains a number of Python scripts allowing to perform various types of setup and analysis tasks. There are two types of these scripts.

Non-interactive scripts can be controlled by command-line arguments and options. Every non-interactive script can be called with the command line option **`-h`** in order to get a description of the functionality and possible options.

Interactive scripts ask the user for information about the task to be conducted (using features like auto-complete and default values), and only perform the task after the input has been completed.

In the tutorial, the input dialogue of the interactive scripts is shown as in this example. **`Red bold text`** gives the input which the user has to type.

```
Type of calculation: 2
Frequency calculation? [True] <ENTER>

Geometry filename: [geom.xyz] (autocomplete enabled) g<TAB>
Geometry filename: [geom.xyz] (autocomplete enabled) geom.xyz

Enter atom indices: (range comprehension enabled) 1~4
```

During the interactive sessions, square brackets indicate that the question has a default answer, which can be used by just pressing ENTER. If filenames or directory paths need to be entered, auto-complete is active, which can be used by pressing TAB. If a list of integers (e.g., atom indices) needs to be entered, range comprehension is active, and ranges can be entered with the tilde symbol (e.g., **`1~4`** is equivalent to **`1 2 3 4`**). Upon completion, every interactive script produces a file **`KEYSTROKES.<name>`**, which contains all user input for the last run.

Please make sure before starting that **`$SHARC`** is set to the directory containing the SHARC scripts and executables.

It is also advisable to set **`$MOLCAS`** to the MOLCAS main directory, and (if installed) to set **`$ORCADIR`** to the ORCA main directory.

Note that in principle one should be able to reproduce all results in this tutorial, i.e., if you closely follow the given steps then you should see exactly the same output (and the same figures). The only exception to this is that the results might be different if you employed a different compiler to compile **`sharc.x`** (this tutorial uses **`gfortran 8.5.0`**) or if you use a different MOLCAS version (this tutorial was generated with OPENMOLCAS 22.10).

Note that it is recommended that for sections in which the user is especially interested, they should refer to the corresponding sections in the SHARC Manual. The corresponding sections are indicated in margin notes on the right (see example on the right).

See Section 7 (p. 177) in the manual.

2.2 Optimization and Frequency calculation

See Section 6.15.4 (p. 109) in the manual.

The first general step of a dynamics simulation is the setup of the initial conditions. Here, we will sample initial conditions randomly from the ground state Wigner distribution.

In order to carry out this step, we need to prepare a MOLDEN file containing the results of a frequency calculation for the ground state. In general, the user is free to calculate the frequencies and normal modes with any quantum chemistry software and any method he sees fit, as long as a MOLDEN file can be produced. However, usually it is advisable to calculate the frequencies at the same level of theory as the dynamics calculation. For the methaniminium cation in our example, we will do the frequency calculation with the quantum chemistry method specified above: (SA(4|3)-CASSCF(6,4)/cc-pVDZ).

Create an empty directory. Prepare a geometry file called **geom.xyz** containing the geometry given above.

```
user@host> mkdir Opt_Freq
user@host> cd Opt_Freq
user@host> vi geom.xyz
```

The SHARC suite comes with an input generator for MOLCAS, which produces input for single-point calculations, optimizations and frequency calculations. It can be invoked with

```
user@host> $SHARC/molcas_input.py
```

The script is interactive. Start the script and prepare an optimization plus frequency calculation on SA-CASSCF level. The script can also generate a Bash-script to instantly launch the MOLCAS calculation.

```
  ================================================================================
||                                                                              ||
||                          MOLCAS Input file generator                         ||
||                                                                              ||
||                             Author: Sebastian Mai                            ||
||                                                                              ||
||                                  Version:4.0                                 ||
||                                   01.04.2024                                 ||
||                                                                              ||
  ================================================================================


This script allows to quickly create MOLCAS input files for single-points calculations
on the SA-CASSCF and (MS-)CASPT2 levels of theory.
It also generates MOLCAS.template files to be used with the SHARC-MOLCAS Interface.

--------------------Type of calculation--------------------

This script generates input for the following types of calculations:
  1       Single point calculations (RASSCF, CASPT2)
  2       Optimizations & Frequency calculations (RASSCF, CASPT2)
  3       MOLCAS.template file for SHARC dynamics (SA-CASSCF)
Please enter the number corresponding to the type of calculation.

Type of calculation: 2                      # Anything after # is a comment
Frequency calculation? [True] <ENTER>       # Opt + Freq

-------------------------Geometry-------------------------

Please specify the geometry file (xyz format, Angstroms):
Geometry filename: [geom.xyz] (autocomplete enabled) <ENTER>    # use default "geom.xyz"
Number of atoms: 6
Nuclear charge: 17
```

```
Enter the total (net) molecular charge:
Charge: [0] +1        # For cation
Number of electrons: 16


----------------------Level of theory----------------------

Supported by this script are:
  1       RASSCF
  2       CASPT2 (Only numerical gradients)

Level of theory: 1

Please enter the basis set.
Common available basis sets:
  Pople:     6-31G**, 6-311G, 6-31+G, 6-31G(d,p), ...
  Dunning:   cc-pVXZ, aug-cc-pVXZ, cc-pVXZ-DK, ...
  ANO:       ANO-S-vdzp, ANO-L, ANO-RCC
Basis set: cc-pVDZ
Use Cholesky decomposition? [True] <ENTER>

----------------------CASSCF Settings----------------------

Number of active electrons: 6
Number of active orbitals: 4
Please enter the number of states for state-averaging as a list of integers
e.g. 3 0 2 for three singlets, zero doublets and two triplets.
Number of states: [0 0 0 0 0 0 0 0] 4  0  3    # same as for dynamics
Accepted number of states: 4 0 3

Please specify the state to optimize
e.g. 3 2 for the second triplet state.
Root: [1 1] <ENTER>    # singlet ground state
Optimization: Only performing one RASSCF for Singlets.  # triplets ignored in optimization of S0
Accepted number of states: 4 0 0

----------------------Further Settings----------------------

########################Full input##########################    # this output is for debugging

ctype            2
freq             True
geom             [['C', 0.0, 0.0, 0.0], ['N', 0.0, 0.0, 1.35], ['H', 0.943102, 0.0, -0.5445],
                 ['H', 0.9431, 0.0, 1.8795], ['H', -0.943102, 0.0, 1.8795], ['H', -0.943102, 0.0, -0.5445]]
ncharge          17
natom            6
charge           1
nelec            16
masslist         [['C', 12.0], ['N', 14.003074], ['H', 1.007825],
                 ['H', 1.007825], ['H', 1.007825], ['H', 1.007825]]
ltype            1
basis            cc-pVDZ
cholesky         False
DK               False
cas.nact         6
cas.norb         4
maxmult          3
cas.nstates      [4, 0, 0]
opt.root         [1, 1]
soc              False
```

```
Writing input to MOLCAS.input

Runscript? [True] <ENTER>

----------------------Path to MOLCAS----------------------

Environment variable $MOLCAS detected:
$MOLCAS=/usr/license/openmolcas/

Do you want to use this MOLCAS installation? [True] <ENTER>

---------------------Scratch directory--------------------

Please specify an appropriate scratch directory. This will be used to run the calculation.
Remember that this script cannot check whether the path is valid, since you may run the
calculation on a different machine. The path will not be expanded by this script.
Path to scratch directory: (autocomplete enabled) $TMPDIR/Tutorial/Opt/

Delete scratch directory after calculation? [False] yes

--------------------------Memory--------------------------

Recommendation: for small systems: 100-300 MB, for medium-sized systems: 1000-2000 MB

Memory in MB:  [500] <ENTER>
Writing run script run_MOLCAS.sh

Finished

************************* WARNINGS *************************
*                                                          *
*                                                          *
************************************************************
```

Execute the run script to start the MOLCAS optimization and frequency calculation.

```
user@host> sh run_MOLCAS.sh
```

This will produce the files **MOLCAS.log**, **MOLCAS.freq.molden** and **MOLCAS.RasOrb**. The first file contains (among other things) the ground state minimum energy (should be −94.412945 Hartree) and the vibrational wavenumbers. Check for any imaginary frequencies. There should be an output block like this in the output file (search for **Numerical differentiation is finished!** to find it quickly, and note that MOLCAS also prints isotope-shifted frequencies which can be easily confused with the non-shifted results):

```
 Numerical differentiation is finished!

Observe that the harmonic oscillator analysis is only valid at stationary points!

Note that rotational and translational degrees have been automatically removed,
if the energy is invariant to these degrees of freedom.

Harmonic frequencies in cm-1

IR Intensities in km/mol

                      1         2         3         4         5         6
```

```
Frequency:       972.20   1022.51   1213.41   1312.37   1418.40   1537.84

Intensity:    2.040E+02 2.281E+00 4.478E-03 3.764E+01 5.286E+01 5.600E-02
Red. mass:      1.29092   1.03756   1.00784   1.29586   1.48038   1.23944

C1         x    0.00042  -0.03544  -0.00005  -0.00039   0.12240  -0.00006
C1         y    0.03987   0.00103  -0.00118  -0.14016  -0.00030  -0.00008

 ⋮          ⋮         ⋮         ⋮         ⋮         ⋮         ⋮         ⋮
```

The **MOLCAS.RasOrb** file contains the CASSCF orbital coefficients and can be used to provide starting orbitals for subsequent calculations. The **MOLCAS.freq.molden** is needed in the next step.

As the MOLCAS calculation produces also several other output files, it is recommended that for the following steps you switch to a different directory:

```
user@host> mkdir ../Tutorial/
user@host> cp MOLCAS.freq.molden MOLCAS.RasOrb MOLCAS.input ../Tutorial/
user@host> cd ../Tutorial/
```

2.3 Sampling initial Conditions from a Wigner distribution

In the next step, the initial coordinates and velocities for the trajectories have to be generated. Here, this task is acomplished by sampling randomly from the Wigner distribution of the ground state nuclear wavefunction (in the harmonic approximation), which can be calculated from the vibrational frequencies and normal modes. This task is performed by the non-interactive script **wigner.py**, which can be executed by typing

See Section 7.1 (p. 177) in the manual.

```
user@host> $SHARC/wigner.py -n 20 MOLCAS.freq.molden
```

The **-n** option is necessary to specify the number of initial conditions to be generated. Here, we generate 20 initial conditions. The output should look like this:

See Section 8.24 (p. 262) in the manual.

```
Initial condition generation started...
INPUT  file                  = "MOLCAS.freq.molden"
OUTPUT file                  = "initconds"
Number of geometries         = 20
Random number generator seed = 16661
Temperature                  = 0.000000

************************************************    # MOLCAS does not write translations and
WARNING: Less than 3*N_atom normal modes extracted!    # rotations to MOLCAS.freq.molden.
************************************************    # You can ignore this warning.


Starting normal mode format determination...
Final format specifier: 2 [cartesian (Molpro, Molcas)]
Multiple possible flags have been identified:
  gaussian-type (Gaussian, Turbomole, Q-Chem, ADF, Orca)
  cartesian (Molpro, Molcas)
The most likely assumption is cartesian (Molpro, Molcas) coordinates.    # Correctly identified Molcas!
These have been used in the creation of inital conditions.

You can override this behavior by setting the -f [int] flag in the command line:
  1     gaussian-type (Gaussian, Turbomole, Q-Chem, AMS, Orca)
  2     cartesian (Molpro, Molcas)
  3     columbus-type (Columbus)
  4     mass-weighted

Geometry:
 C   6.0  -0.00000000   0.00000000   0.06656040  12.00000000
 N   7.0  -0.00000001  -0.00000000   2.47206168  14.00307400
 H   1.0   1.77786426   0.00000005  -0.94410407   1.00782500
 H   1.0   1.62625677  -0.00000006   3.47314251   1.00782500
 H   1.0  -1.62625679   0.00000006   3.47314255   1.00782500
 H   1.0  -1.77786423  -0.00000005  -0.94410405   1.00782500
Assumed Isotopes: N-14 C-12 H-1
Isotopes with * are pure isotopes.

Frequencies (cm^-1) used in the calculation:
   1     972.1988
   2    1022.5063
   3    1213.4146
   4    1312.3702
   5    1418.3997
   6    1537.8447
   7    1679.1760
   8    1882.3179
   9    3330.3424
  10    3464.1861
```

```
  11    3679.9963
  12    3764.2148


Sampling initial conditions
Progress: [==================================================] 100%
```

The results of the sampling are written to the file **`initconds`**. This file contains all necessary information (equilibrium geometry, plus 20 sets of randomly sampled geometries with corresponding velocities) for subsequent steps.

2.4 Setting up the initial energy calculations

Besides the initial geometries and velocities, it is necessary to determine (for each initial geometry) the initial excited state from where the dynamics commences. In order to find the initial states after instantaneous vertical excitation, it is necessary to obtain the excitation energies and oscillator strengths for all initial geometries. These calculations can be setup using the script **setup_init.py**.

See Section 3.2 (p. 38) in the manual.

Note that it is also possible to simply specify the initial state for each initial condition manually; in this case, it is not necessary to use **setup_init.py** to prepare vertical excitation calculations.

2.4.1 MOLCAS input template

The computations which are setup by **setup_init.py** utilize the SHARC interfaces to call the respective quantum chemistry program (MOLCAS here). The SHARC-MOLCAS interface requires a template file which specifies the level of theory. Thus, before proceeding to setup the initial calculations, we need to prepare the MOLCAS template file.

See Section 6.15.4 (p. 109) in the manual.

Again, launch the MOLCAS input generator:

```
user@host> $SHARC/molcas_input.py
```

Prepare a template file for the SHARC-MOLCAS interface.

```
  ================================================================================
||                                                                                ||
||                          MOLCAS Input file generator                           ||
||                                                                                ||
||                             Author: Sebastian Mai                              ||
||                                                                                ||
||                                  Version:4.0                                   ||
||                                   01.04.2025                                   ||
||                                                                                ||
  ================================================================================


This script allows to quickly create MOLCAS input files for single-points calculations
on the SA-CASSCF and (MS-)CASPT2 levels of theory.
It also generates MOLCAS.template files to be used with the SHARC-MOLCAS Interface.

--------------------Type of calculation--------------------

This script generates input for the following types of calculations:
  1       Single point calculations (RASSCF, CASPT2)
  2       Optimizations & Frequency calculations (RASSCF, CASPT2)
  3       MOLCAS.template file for SHARC dynamics (SA-CASSCF)
Please enter the number corresponding to the type of calculation.

Type of calculation: 3        # MOLCAS.template generation

-------------------------Geometry-------------------------

No geometry necessary for MOLCAS.template generation

Number of electrons:  [16] <ENTER>        # If MOLCAS.input is in the same directory,
                                          # some of the following input is auto-detected.
----------------------Level of theory----------------------

Supported by this script are:
```

```
  1       RASSCF
  2       CASPT2

Level of theory: 1

Please enter the basis set.
Common available basis sets:
  Pople:     6-31G**, 6-311G, 6-31+G, 6-31G(d,p), ...    (Not available)
  Dunning:   cc-pVXZ, aug-cc-pVXZ, cc-pVXZ-DK, ...
  ANO:       ANO-S-vdzp, ANO-L, ANO-RCC
Basis set: [cc-pVDZ] <ENTER>
Use Cholesky decomposition? [True] <ENTER>

----------------------CASSCF Settings---------------------

Number of active electrons: [6] <ENTER>
Number of active orbitals: [4] <ENTER>
Please enter the number of states for state-averaging as a list of integers
e.g. 3 0 2 for three singlets, zero doublets and two triplets.
Number of states: [4] 4  0  3        # Override suggestion to include triplet states.
Accepted number of states: 4 0 3

----------------------Further Settings----------------------


########################Full input#########################

ltype            1
soc              False
cholesky         False
DK               False
basis            cc-pVDZ
cas.norb         4
maxmult          3
cas.nact         6
ctype            3
cas.nstates      [4, 0, 3]
nelec            16
geom             None
freq             False

Writing input to MOLCAS.template

Finished

************************* WARNINGS *************************
*                                                          *
*                                                          *
************************************************************
```

This will create a file called **`MOLCAS.template`**, which is needed for the following setup steps. Note that in the template file, the number of active electrons is given as 7, because that number refers to the neutral molecule. The interface will adjust the number of active electrons for each multiplicity based on the charge requested for that multiplicity.

2.4.2 Setup of initial calculations

With the necessary files (**initconds**, **MOLCAS.template**, **MOLCAS.RasOrb**) available, the script **setup_init.py** can be launched. Note that this script creates a large number of subdirectories in the directory where the script is run, so it is advisable to run the setup in a dedicated directory:

See Section 7.9 (p. 188) in the manual.

```
user@host> mkdir init/
user@host> cd init/
user@host> $SHARC/setup_init.py
```

This script is also interactive.

```
  ================================================================================
||                                                                              ||
||                  Setup initial conditions for SHARC dynamics                 ||
||                                                                              ||
||                   Authors: Sebastian Mai, Severin Polonius                   ||
||                                                                              ||
||                                Version:4.0                                   ||
||                                 01.05.25                                     ||
||                                                                              ||
  ================================================================================


This script automatizes the setup of excited-state calculations for initial conditions
for SHARC dynamics.

------------------Initial conditions file-----------------


If you do not have an initial conditions file, prepare one with wigner.py!

Please enter the filename of the initial conditions file.
Initial conditions filename: [initconds] (autocomplete enabled) ../initconds

File "../initconds" contains 20 initial conditions.
Number of atoms is 6

-----------------Range of initial conditions----------------

Please enter the range of initial conditions for which an excited-state calculation should be
performed as two integers separated by space.
Initial condition range: [1 20] 1  10    # not all initial conditions need to be calculated

Script will use initial conditions 1 to 10 (10 in total).

---------------Number of states and charge----------------

Please enter the number of states as a list of integers
e.g. 3 0 3 for three singlets, zero doublets and three triplets.
Number of states: 4  0  3     # this could differ from the number of SA roots in MOLCAS.template

Please enter the molecular charge for each chosen multiplicity
e.g. 0 +1 0 for neutral singlets and triplets and cationic doublets.
Molecular charges per multiplicity: [0 1 0] 1 0 1
Number of states: [4, 0, 3]
Total number of states: 13
```

```
Loading interface collection from $SHARC ...

-----------Choose the quantum chemistry interface-----------

Please specify the quantum chemistry interface (enter any of the following numbers):
  1 SHARC_ADAPTIVE          HYBRID interface for adaptive sampling
  2 SHARC_AMS_ADF                 (Not Available! Use SHARC_LEGACY to work with this interface)
  3 SHARC_ANALYTICAL          FAST interface for analytical model Hamiltonians with sympy
  4 SHARC_ASE_DB            HYBRID interface for saveing data to ASE_DB db
  5 SHARC_BAGEL                   (Not Available! Use SHARC_LEGACY to work with this interface)
  6 SHARC_COLUMBUS                (Not Available! Use SHARC_LEGACY to work with this interface)
  7 SHARC_DO_NOTHING          FAST interface for testing (zero E/grad/NAC/DM, unity overlaps/phases)
  8 SHARC_ECI               HYBRID interface for excitonic HF/CI with multiple fragments
  9 SHARC_FALLBACK          HYBRID interface for calling a fallback interface if primary interface fails
 10 SHARC_GAUSSIAN       AB INITIO interface for GAUSSIAN16 for single-reference methods (CIS, TDDFT)
 11 SHARC_LEGACY             BASIC interface for running legacy interfaces via file I/O (AMS-ADF, BAGEL, ...
 12 SHARC_LVC                FAST interface for linear/quadratic vibronic coupling models (LVC, QVC, ...
 13 SHARC_MNDO           AB INITIO interface for the MNDO program (OM2-MRCI)
 14 SHARC_MOLCAS         AB INITIO interface for OpenMolcas (>v23) for multireference calculations ...
 15 SHARC_MOLPRO                 (Not Available! Use SHARC_LEGACY to work with this interface)
 16 SHARC_MOPACPI        AB INITIO interface for the MOPAC-PI program
 17 SHARC_NUMDIFF           HYBRID interface for numerical derivatives (grad, NACdr, SOCdr, DMdr)
 18 SHARC_NWCHEM         AB INITIO interface for NWChem (TDDFT)
 19 SHARC_OPENMM                 (Not Available!)
 20 SHARC_ORCA           AB INITIO interface for ORCA v5-6 (CIS/TDDFT)
 21 SHARC_PYSCF                  (Not Available! Use SHARC_LEGACY to work with this interface)
 22 SHARC_QMMM              HYBRID interface for QM/MM (electrostatic embedding, link atom scheme)
 23 SHARC_QMOUT              FAST interface for frozen-nuclei dynamics (constant E/SOC/DM, unit ...
 24 SHARC_SPAINN             FAST interface for SPaiNN
 25 SHARC_TURBOMOLE      AB INITIO interface for TURBOMOLE (RICC2/ADC2)
 26 SHARC_UMBRELLA          HYBRID interface for adding umbrella-sampling-style restraints (harmonic ...

Interface number: 14

The following interface was selected:
 14 SHARC_MOLCAS         AB INITIO interface for OpenMolcas (>v23) for multireference calculations ...

The following features are available from this interface:
{'overlap', 'soc', 'nacdr', 'h', 'dm', 'nacdr_pc', 'grad_pc', 'ion', 'grad', 'molden', 'density_matrices',
'theodore', 'phases', 'multipolar_fit', 'wave_functions', 'point_charges', 'mol'}

-----------------Spin-orbit couplings (SOCs)---------------

Do you want to compute spin-orbit couplings?

Spin-Orbit calculation? [True] <ENTER>     # not required, but included anyways.
Will calculate spin-orbit matrix.


----------------Overlaps to reference states---------------

Do you want to compute the overlaps between the states at the equilibrium geometry
and the states at the initial condition geometries?
Reference overlaps? [False] yes    # not required, but included anyways.

--------------TheoDORE wave function analysis---------------

Do you want to run TheoDORE to obtain one-electron descriptors for the electronic wave functions?
TheoDORE? [False] <ENTER>
  ================================================================================
```

```
||                              MOLCAS Interface setup                            ||
  ================================================================================


Specify path to MOLCAS.
Path to MOLCAS: [$MOLCAS] (autocomplete enabled) <ENTER>


Specify a scratch directory. The scratch directory will be used to run the calculations.
Path to scratch directory: (autocomplete enabled) $TMPDIR/Tutorial/Init
Specify a path to a MOLCAS template file.
Template path: (autocomplete enabled) ../MOLCAS.template
Specify the number of CPUs to be used.
Number of CPUs: [1] <ENTER>
Specify the amount of RAM to be used.
Memory (MB): [1000] 500
Initial wavefunction: MO Guess

Please specify the path to a MOLCAS JobIph file containing suitable starting MOs for the CASSCF calculation.
Please note that this script cannot check whether the wavefunction file and the Input template are consistent!
Do you have initial wavefunction files for multiplicities 1 3? [True] <ENTER>
JobIph files (1) or RasOrb files (2)? 2
Initial wavefunction file for multiplicity 1: [MOLCAS.1.RasOrb.init] (autocomplete enabled) ../MOLCAS.RasOrb
Initial wavefunction file for multiplicity 3: [MOLCAS.3.RasOrb.init] (autocomplete enabled) ../MOLCAS.RasOrb

  ================================================================================
||                                 Run mode setup                                 ||
  ================================================================================


-------------------------Run script-------------------------

This script can generate the run scripts for each initial condition in two modes:

  - In the first mode, the calculation is run in subdirectories of the current directory.

  - In the second mode, the input files are transferred to another directory (e.g. a
  local scratch directory), the calculation is run there, results are copied back and
  the temporary directory is deleted. Note that this temporary directory is not the
  same as the scratchdir employed by the interfaces.

Note that in any case this script will setup the input subdirectories in the current working directory.

In case of mode 1, the calculations will be run in:
/user/mai/Documents/NewSHARC/SHARC_4.0/TUTORIAL/2_full/Tutorial/init

Use mode 1 (i.e., calculate here)? [True] <ENTER>

----------------------Submission script----------------------

During the setup, a script for running all initial conditions sequentially in batch mode is generated.
Additionally, a queue submission script can be generated for all initial conditions.

Generate submission script? [False] <ENTER>


#########################Full input#########################

cwd                       /user/mai/Documents/NewSHARC/SHARC_4.0/TUTORIAL/2_full/Tutorial/init
ninit                     20
```

```
natom                   6
initf                   <_io.TextIOWrapper name='../initconds' mode='r' encoding='UTF-8'>
irange                  [1, 10]
states                  [4, 0, 3]
nstates                 13
charge                  [0, 1, 0]
needed_requests         'dm', 'soc', 'overlap', 'h'
soc                     True
refov                   True
theodore                False
here                    True
qsub                    False


Do you want to setup the specified calculations? [True] <ENTER>

Do you want to link the interface files? [False] <ENTER>

  ================================================================================
||                           Setting up directories...                          ||
  ================================================================================


Progress: [================================================] 100%
```

The script will create directories **ICOND_00001/**, **ICOND_00002/**, ... for each initial condition (and **ICOND_00000/** for the equilibrium geometry), with the corresponding inputs for the interface and a Bash runscript. Additionally, the script **all_run_init.sh** is generated, which allows to run all excited-state calculations subsequently.

Run all initial conditions calculations sequentially:

```
user@host> sh all_run_init.sh
```

For larger calculations, it is often advantageous to send the scripts **ICOND_*/run.sh** to a queuing system to distribute the calculations over a computing cluster. However, note that with the **reference overlap** calculations, **ICOND_00000** must be completed before you can send the other calculations.

After the calculations are finished, each subdirectory should contain a file called **QM.out** holding the Hamiltonian and transition dipole moment matrices.

2.5 Selection of initial excited states

In the next step, the results of the excited-state calculations have to be read, converted to excitation energies and oscillator strengths, and the brightest initial conditions selected for the dynamics simulation. These tasks can be accomplished using

See Section 7.10 (p. 190) in the manual.

```
user@host> $SHARC/excite.py
```

This script is interactive. Per default, during the run the script reads the ground state equilibrium energy from **ICOND_00000/QM.out**, if this file exists. Otherwise, the script asks the user to enter the ground states equilibrium energy.

See Section 8.9 (p. 247) in the manual.

```
  ================================================================================
||                                                                              ||
||                      Excite initial conditions for SHARC                     ||
||                                                                              ||
||                              Author: Sebastian Mai                           ||
||                                                                              ||
||                                   Version:4.0                                ||
||                                    17.03.24                                  ||
||                                                                              ||
  ================================================================================


This script automatizes to read-out the results of initial excited-state calculations for SHARC.
It calculates oscillator strength (in MCH and diagonal basis) and stochastically determines whether
a trajectory is bright or not.

------------------Initial conditions file-----------------


If you do not have an initial conditions file, prepare one with wigner.py!

Please enter the filename of the initial conditions file.
Initial conditions filename: [initconds] (autocomplete enabled) ../initconds

File "../initconds" contains 20 initial conditions.
Number of atoms is 6

---------------Generate excited state lists---------------

Using the following options, excited state lists can be added to the initial conditions:

1       Generate a list of dummy states
2       Read excited-state information from ab initio calculations (from setup_init.py)

How should the excited-state lists be generated? [2] <ENTER>     # Read from ICOND_*/
Please enter the path to the directory containing the ICOND subdirectories.
Path to ICOND directories: (autocomplete enabled) .            # "." is the current directory

/user/mai/Documents/NewSHARC/SHARC_2.0/TUTORIAL/2_full/Tutorial/init
Directory contains 11 subdirectories.
There are more initial conditions in ../initconds.

---------------Excited-state representation---------------

This script can calculate the excited-state energies and oscillator strengths in two representations.
These representations are:
- MCH representation: Only the diagonal elements of the Hamiltonian are taken into account.
```

```
  The states are the spin-free states as calculated in the quantum chemistry code.
  This option should be used if the ground state is spin-pure.
- diagonal representation: The Hamiltonian including spin-orbit coupling is diagonalized.
  The states are spin-corrected, fully adiabatic. Note that for this the excited-state calculations ...
  This option should be used if the ground state is spin-mixed.

Do you want to use the diagonal representation (True=diag, False=MCH)? [False] <ENTER>

----------------------Reference energy----------------------

Reference energy read from file
/user/mai/Documents/NewSHARC/SHARC_4.0/TUTORIAL/2_full/Tutorial/init/ICOND_00000/QM.out
E_ref= -94.412945540000      # automatically read from ICOND_00000/QM.out

------------------Excited-state selection-----------------

Using the following options, the excited states can be flagged as valid initial states for dynamics:

1       Unselect all initial states
2       Provide a list of desired initial states
3       Simulate delta-pulse excitation based on excitation energies and oscillator strengths

How should the excited states be flagged? [3] <ENTER>


----------------------Excitation window---------------------

Enter the energy window for exciting the trajectories.
Range (eV): [0.0 10.0] 9  12

Script will allow excitations only between 9.000000 eV and 12.000000 eV.

----------------------Considered states---------------------

From which state should the excitation originate (for computation of excitation energies
and oscillator strength)?
Lower state for excitation? [1] <ENTER>
#State  Mult    M_s     Quant    # Here the states are listed.
1       1       +0.0    1
2       1       +0.0    2
3       1       +0.0    3
4       1       +0.0    4
5       3       -1.0    1
6       3       -1.0    2
7       3       -1.0    3
8       3       +0.0    1
9       3       +0.0    2
10      3       +0.0    3
11      3       +1.0    1
12      3       +1.0    2
13      3       +1.0    3

Do you want to include all states in the selection? [True] <ENTER>

---------------------Random number seed--------------------

Please enter a random number generator seed (type "!" to initialize the RNG from the system time).
RNG Seed:  [!] 1234
```

```
#########################Full input#########################

ninit                       20
natom                       6
repr                        MCH
eref                        -457.9790486157
eharm                       0.0
states                      [4, 0, 3]
initf                       <_io.TextIOWrapper name='../initconds' mode='r' encoding='UTF-8'>
gen_list                    2
read_QMout                  True
make_list                   False
iconddir                    /user/mai/Documents/NewSHARC/SHARC_4.0/TUTORIAL/2_full/Tutorial/init
ncond                       11
diag                        False
ion                         False
excite                      3
erange                      [0.3307438995808949, 0.4409918661078599]
diabatize                   False
initstate                   0
allowed                     set()


Do you want to continue? [True] <ENTER>

Reading initial condition file ....
  Progress: [==================================================] 100%
Number of initial conditions in file:          20

Reading QM.out data ...
  Progress: [==================================================] 100%
Number of initial conditions with QM.out:      10

Selecting initial states ...
  Progress: [==================================================] 100%
Number of initial states:                       9

Number of initial conditions excited:
State   Selected   InRange   Total
    1          0         0      10
    2          1         5      10
    3          9        10      10     # we can setup 9 trajectories from state 3 (S2)
    4          2         8      10
    5          0         0      10
    6          0         1      10
    7          0        10      10
    8          0         0      10
    9          0         1      10
   10          0        10      10
   11          0         0      10
   12          0         1      10
   13          0        10      10

Writing output to ../initconds.excited ...
```

excite.py will generate a new file called **initconds.excited**, which contains all information from the **initconds** file, as well as information about the ground state equilibrium energy, the state representation and the excited states for each initial condition. This file is necessary in order to calculate absorption spectra

and to setup trajectories.

If you later want to do another selection (with a different excitation window or with the exclusion of some states), you can tell **`excite.py`** to read from **`initconds.excited`**, instead of reading all **`QM.out`** files again.

From the **`initconds.excited`** file, also absorption spectra can be generated, see section 2.6. If you do not want to compute a spectrum and directly go to the trajectory setup, go to section 2.7.

2.6 Absorption spectra from Initial conditions files

The content of the file **initconds.excited** can be used to generate absorption spectra which go beyond the Condon approximation. The spectrum is the sum of the spectra of each initial condition, which is a line spectrum of the excitation energies versus the oscillator strengths. A Gaussian (or Lorentzian, or Log-normal) convolution of the line spectra can be done as well.

The calculation of convoluted or line spectra is carried out by **spectrum.py**.

See Section 7.11 (p. 194) in the manual.

2.6.1 Example

See Section 8.1 (p. 242) in the manual.

Call the script by

```
user@host> cd ..
user@host> $SHARC/spectrum.py -o spectrum.out -e 9 12 initconds.excited
```

Using command-line options, it is possible to calculate only spectra for part of the initial condition set, to change the size and limits of the energy grid (here we plot from 9 eV to 12 eV) and to influence the line shape (Gaussian vs. Lorentzian vs. Log-normal, as well as FWHM). With the **-l** option a line spectrum is produced, and with the **-D** option a density-of-states spectrum is produced.

The program also writes some information about the calculation to the screen:

```
Number of grid points: 500
Energy range: 9.000 to 12.000 eV
Lineshape: Gaussian (FWHM=0.100 eV)
Number of initial conditions: 20
Reference energy   -94.4129639182
Representation: MCH
Reading initial conditions 1 to 20

Progress: [================================================] 100%

Number of states: 13
Number of initial conditions with excited-state information (per state):
10 10 10 10 10 10 10 10 10 10 10 10 10

Progress: [================================================] 100%

Maximum of the absorption spectrum: 0.869223

Output spectrum written to "spectrum.out".
```

The results can be easily plotted using GNUPLOT. Just give the corresponding command-line flag and then call GNUPLOT:

```
user@host> $SHARC/spectrum.py -o spectrum.out -e 9 12
           --gnuplot spectrum.gp initconds.excited
user@host> gnuplot spectrum.gp
```

In figure 2.1 the result of this convolution is shown. Note that on CASSCF level of theory, the excitation energy of ethylene is reproduced quite badly and the number of initial conditions is too low to reliably sample the ground state Wigner distribution.

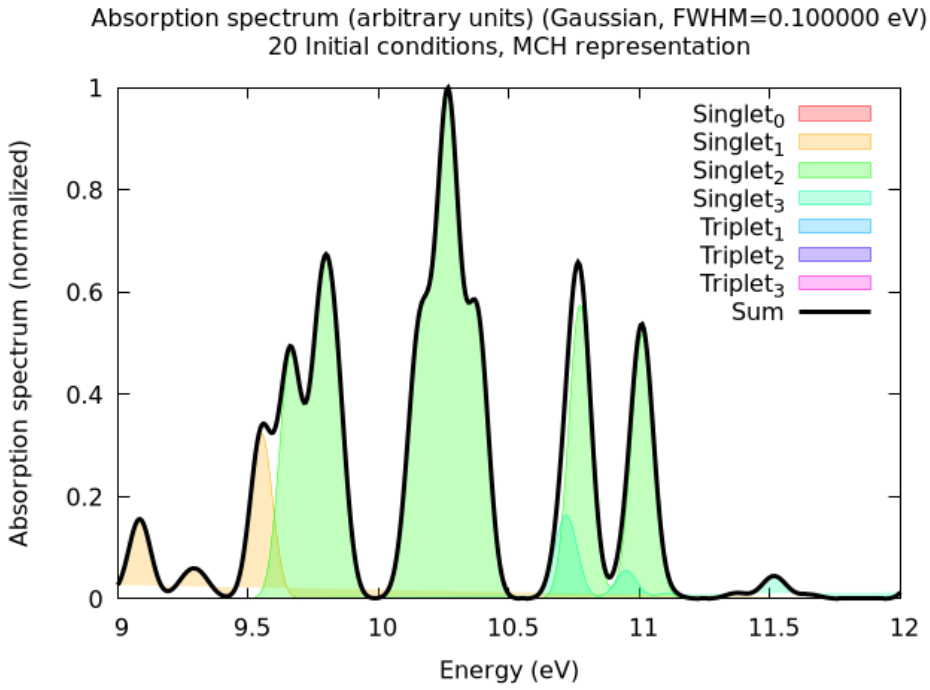

Figure 2.1: Absorption spectrum based on 10 initial conditions (Since there are 20 initial conditions in **`initconds.excited`**, the title lists 20, even though only 10 were actually evaluated). **Your result might numerically differ.**

2.7 Setting up dynamics simulations

The last preparatory step towards dynamics simulations consists naturally in setting up the SHARC input files and run scripts. The interactive script **setup_traj.py** takes care of this step. Run the script in the directory where the trajectories should be set up. For this, create a new directory:

See Section 7.18 (p. 201) in the manual.

```
user@host> mkdir traj
user@host> cd traj
```

Make sure that you have all required files (**initconds.excited**, **MOLCAS.template**, **MOLCAS.RasOrb**) in this directory. Then start the setup script:

```
user@host> $SHARC/setup_traj.py
```

```
Script for setup of SHARC trajectories started...


  ============================================================================
||                                                                            ||
||                    Setup trajectories for SHARC dynamics                   ||
||                                                                            ||
||          Authors: Sebastian Mai, Philipp Marquetand, Severin Polonius      ||
||                                                                            ||
||                                  Version: 4.0                              ||
||                                 Date: 01.09.24                             ||
||                                                                            ||
  ============================================================================


This script automatizes the setup of the input files for SHARC dynamics.


  ============================================================================
||                             Initial conditions                             ||
  ============================================================================



This script reads the initial conditions (geometries, velocities, initial excited state)
from the initconds.excited files as provided by excite.py.

Please enter the filename of the initial conditions file.
Initial conditions filename: [initconds.excited] (autocomplete enabled) ../initconds.excited

File ../initconds.excited contains 20 initial conditions.
Number of atoms is 6
Reference energy -94.412963918200 a.u.
Excited states are in MCH representation.


Please enter the number of states as a list of integers
e.g. 3 0 3 for three singlets, zero doublets and three triplets.
Number of states: [4 0 3] <ENTER>       # a different number of states than in the
                                        # initial calculations could be used in the dynamics

Number of states: [4, 0, 3]


Please enter the molecular charge for each chosen multiplicity
```

```
e.g. 0 +1 0 for neutral singlets and triplets and cationic doublets.
Molecular charges per multiplicity: [0 1 0] 1 0 1
Number of states: [4, 0, 3]
Total number of states: 13

Do you want all states to be active? [True] <ENTER>

Do you want to see the content of the initconds file? [False] yes
Reading initconds file
  Progress: [================================================================================] 100%
Number of initial conditions in file:          20
Contents of the initconds file:

Legend:
?       Geometry and Velocity
.       not selected
#       selected

State 1:                      # we only performed 10 calculations, hence the "?"
             10         20
              |          |
 0 | .......... ??????????   # no initial conditions selected in S0
State 2:
             10         20
              |          |
 0 | ....#..... ??????????   # 1 initial condition selected in S1
State 3:
             10         20
              |          |
 0 | ####.##### ??????????   # initial conditions 1-4 and 6-10 are selected
State 4:
             10         20
              |          |
 0 | #..#...... ??????????   # 2 initial conditions selected in S3
State 5:
             10         20
              |          |
 0 | .......... ??????????   # no initial conditions selected in T1
State 6:
             10         20
              |          |
 0 | .......... ??????????   # no initial conditions selected in T2
State 7:
             10         20
              |          |
 0 | .......... ??????????   # no initial conditions selected in T3
State 8:
             10         20
              |          |
 0 | .......... ??????????
State 9:
             10         20
              |          |
 0 | .......... ??????????
State 10:
             10         20
              |          |
 0 | .......... ??????????
State 11:
             10         20
```

```
             |          |
 0 | .......... ??????????
State 12:
             10         20
             |          |
 0 | .......... ??????????
State 13:
             10         20
             |          |
 0 | .......... ??????????
Number of excited states and selections:
State    #InitCalc       #Selected
    1           10               0
    2           10               1
    3           10               9  # we can setup 9 trajectories starting in state 3 (S2)
    4           10               2
    5           10               0
    6           10               0
    7           10               0
    8           10               0
    9           10               0
   10           10               0
   11           10               0
   12           10               0
   13           10               0

Please enter a list specifying for which excited states trajectories should be set-up
e.g. 8 10 12 to select states 8, 10, and 12.
States to setup the dynamics: [2 3 4] (range comprehension enabled) <ENTER>

There can be 12 trajectories set up.

Please enter the index of the first initial condition in the initconds file to be setup.
Starting index: [1] <ENTER>

There can be 12 trajectories set up, starting in 1 states.

Please enter the total number of trajectories to setup.
Number of trajectories: [12] <ENTER>

Please enter a random number generator seed (type "!" to initialize the RNG from the system time).
RNG Seed:  [!] 1234


  ================================================================================
||                          Quantum chemistry interface                           ||
  ================================================================================


Loading interface collection from $SHARC ...

-----------Choose the quantum chemistry interface-----------

Please specify the quantum chemistry interface (enter any of the following numbers):
  1 SHARC_ADAPTIVE          HYBRID interface for adaptive sampling
  2 SHARC_AMS_ADF                 (Not Available! Use SHARC_LEGACY to work with this interface)
  3 SHARC_ANALYTICAL          FAST interface for analytical model Hamiltonians with sympy
  4 SHARC_ASE_DB            HYBRID interface for saveing data to ASE db
  5 SHARC_BAGEL                   (Not Available! Use SHARC_LEGACY to work with this interface)
  6 SHARC_COLUMBUS                (Not Available! Use SHARC_LEGACY to work with this interface)
```

```
  7 SHARC_DO_NOTHING          FAST interface for testing (zero E/grad/NAC/DM, unity overlaps/phases)
  8 SHARC_ECI               HYBRID interface for excitonic HF/CI with multiple fragments
  9 SHARC_FALLBACK          HYBRID interface for calling a fallback interface if primary interface fails
 10 SHARC_GAUSSIAN       AB INITIO interface for GAUSSIAN16 for single-reference methods (CIS, TDDFT)
 11 SHARC_LEGACY             BASIC interface for running legacy interfaces via file I/O (AMS-ADF, BAGEL, ...
 12 SHARC_LVC                FAST interface for linear/quadratic vibronic coupling models (LVC, QVC, ...
 13 SHARC_MNDO           AB INITIO interface for the MNDO program (OM2-MRCI)
 14 SHARC_MOLCAS         AB INITIO interface for OpenMolcas (>v23) for multireference calculations ...
 15 SHARC_MOLPRO                   (Not Available! Use SHARC_LEGACY to work with this interface)
 16 SHARC_MOPACPI        AB INITIO interface for the MOPAC-PI program
 17 SHARC_NUMDIFF          HYBRID interface for numerical derivatives (grad, NACdr, SOCdr, DMdr)
 18 SHARC_NWCHEM         AB INITIO interface for NWChem (TDDFT)
 19 SHARC_OPENMM                   (Not Available!)
 20 SHARC_ORCA           AB INITIO interface for ORCA v5-6 (CIS/TDDFT)
 21 SHARC_PYSCF                    (Not Available! Use SHARC_LEGACY to work with this interface)
 22 SHARC_QMMM             HYBRID interface for QM/MM (electrostatic embedding, link atom scheme)
 23 SHARC_QMOUT              FAST interface for frozen-nuclei dynamics (constant E/SOC/DM, unit ...
 24 SHARC_SPAINN             FAST interface for SPaiNN
 25 SHARC_TINKER                   (Not Available!)
 26 SHARC_TURBOMOLE      AB INITIO interface for TURBOMOLE (RICC2/ADC2)
 27 SHARC_UMBRELLA         HYBRID interface for adding umbrella-sampling-style restraints (harmonic ...

Interface number: 14

The following interface was selected:
 14 SHARC_MOLCAS         AB INITIO interface for OpenMolcas (>v23) for multireference calculations ...


The following features are available from this interface:
{'point_charges', 'nacdr_pc', 'density_matrices', 'nacdr', 'overlap', 'multipolar_fit', 'mol',
'wave_functions', 'soc', 'ion', 'phases', 'molden', 'h', 'dm', 'grad', 'theodore', 'grad_pc'}

  ============================================================================
||                       Surface Hopping dynamics settings                    ||
  ============================================================================


Please choose the dynamics method you want to employ.
1       Trajectory surface hopping dynamics using single surface potential
2       Semi-classical Ehrenfest dynamics using self-consistent potential
Method: [1] <ENTER>
----------------------Simulation time---------------------

Please enter the total simulation time.
Simulation time (fs): [1000.0] 100

Please enter the simulation timestep (0.5 fs recommended).
Simulation timestep (fs): [0.5] <ENTER>

Simulation will have 21 timesteps.

Please choose the integrator you want to use
1       adaptive timestep Velocity-Verlet integrator
2       fixed timestep Velocity-Verlet integrator
Integrator: [2] <ENTER>

Please enter the number of substeps for propagation (25 recommended).
Nsubsteps: [25] <ENTER>

The trajectories can be prematurely terminated after they run for a certain time in the lowest state.
```

```
Do you want to prematurely terminate trajectories? [False] <ENTER>


----------------------Dynamics settings--------------------

Do you want to perform the dynamics in the diagonal representation (SHARC dynamics)
or in the MCH representation (regular TSH/SCP)?
SHARC dynamics? [True] <ENTER>
Do you want to include spin-orbit couplings in the dynamics?

Spin-Orbit calculation? [True] <ENTER>
Will calculate spin-orbit matrix.


Please choose the quantities to describe non-adiabatic effects between the states:
1       DDT     =  < a|d/dt|b >            Hammes-Schiffer-Tully scheme (not available)
2       DDR     =  < a|d/dR|b >            Original Tully scheme
3       ktdc    = sqrt(D2(dV)/dt2/(dV))/2  Curvature Driven TDC scheme
4       overlap = < a(t0)|b(t) >           Local Diabatization scheme
Coupling number: [4] <ENTER>

Please choose the gradient mixing scheme for the gradients:
1       mixed gradients are calculated as linear combination of MCH gradients only
2       mixed gradients are calculated by correction of MCH gradients with non-adiabatic coupling vector
3       mixed gradients are calculated by rescaling of the MCH gradients according
        to time derivatives in diagonal and MCH representations
Gradient mixing scheme: [1] <ENTER>

During a surface hop, the kinetic energy has to be modified in order to conserve total energy.
There are several options to that:
1       Do not conserve total energy. Hops are never frustrated.
2       Adjust kinetic energy by rescaling the velocity vectors. Often sufficient.
3       Adjust ... along the vibrational velocity vector.
4       Adjust ... along the non-adiabatic coupling vector.
5       Adjust ... along the gradient difference vector.
6       Adjust ... along the projected non-adiabatic coupling vector.
7       Adjust ... along the effective non-adiabatic coupling vector.
8       Adjust ... along the projected effective non-adiabatic coupling vector.
EkinCorrect: [2] <ENTER>

If a surface hop is refused (frustrated) due to insufficient energy, the velocity can either be
left unchanged or reflected:
1       Do not reflect at a frustrated hop.
2       Reflect the full velocity vector.
3       Reflect the vibrational velocity vector.
4       Reflect ... along the non-adiabatic coupling vector.
5       Reflect ... along the gradient difference vector.
6       Reflect ... along the projected non-adiabatic coupling vector.
7       Reflect ... along the effective non-adiabatic coupling vector.
8       Reflect ... along the projected effective non-adiabatic coupling vector.
Reflect frustrated: [1] <ENTER>

Please choose a decoherence correction for the diagonal states:
1       No decoherence correction.
2       Energy-based decoherence scheme (Granucci, Persico, Zoccante).
3       Augmented fewest-switching surface hopping (Jain, Alguire, Subotnik).
Decoherence scheme: [2] <ENTER>

Please choose a surface hopping scheme for the diagonal states:
1       Surface hops off.
```

```
2       Standard SHARC surface hopping probabilities (Mai, Marquetand, Gonzalez).
3       Global flux surface hopping probabilities (Wang, Trivedi, Prezhdo).
Hopping scheme: [2] <ENTER>

Do you want to perform forced hops to the lowest state based on a energy gap criterion?
(Note that this ignores spin multiplicity)
Forced hops to ground state? [False] <ENTER>

Do you want to scale the energies and gradients?
Scaling? [False] <ENTER>

Do you want to damp the dynamics (Kinetic energy is reduced at each timestep by a factor)?
Damping? [False] <ENTER>

Do you want to use an atom mask for velocity rescaling or decoherence?
Atom masking? [False] <ENTER>

--------------Selection of Gradients and NACs-------------

In order to speed up calculations, SHARC is able to select which gradients and NAC vectors it has to
calculate at a certain timestep. The selection is based on the energy difference between the state
under consideration and the classical occupied state.

Select gradients? [False] yes    # this strongly speeds up the calculations

Please enter the energy difference threshold for the selection of gradients and non-adiabatic
couplings (in eV). (0.5 eV recommended, or even larger if SOC is strong in this system.)
Selection threshold (eV): [0.5] 0.1

-----------------Settings for large systems-----------------

Do you want to constrain some bond lengths (via a RATTLE)? [False] <ENTER>
Do you want to use a thermostat? [False] <ENTER>


------------------------Laser file------------------------

Do you want to include a laser field in the simulation? [False] <ENTER>

--------------TheoDORE wave function analysis--------------

Do you want to run TheoDORE to obtain one-electron descriptors for the electronic wave functions?
TheoDORE? [False] <ENTER>

  ================================================================================
||                                Interface setup                                 ||
  ================================================================================
# Everything coming after here is done by the MOLCAS interfaces itself.
# If you use hybrid interfaces, you will get multiple such dialogues.

================================================================================
||                                                                            ||
||                          MOLCAS interface setup                            ||
||                                                                            ||
================================================================================

Specify path to MOLCAS.
Path to MOLCAS: [$MOLCAS] (autocomplete enabled) <ENTER>

Specify a scratch directory. The scratch directory will be used to run the calculations.
```

```
Path to scratch directory: (autocomplete enabled) $TMPDIR/Tutorial/Traj_WORK/
Specify a path to a MOLCAS template file.
Template path: (autocomplete enabled) ../MOLCAS.template
Specify the number of CPUs to be used.
Number of CPUs: [1] 1
Specify the amount of RAM to be used.
Memory (MB): [1000] 500
Initial wavefunction: MO Guess

Please specify the path to a MOLCAS JobIph file containing suitable starting MOs for the CASSCF calculation.
Please note that this script cannot check whether the wavefunction file and the Input template are consistent!
Do you have initial wavefunction files for multiplicities 1 3? [True] <ENTER>
JobIph files (1) or RasOrb files (2)? 2
Initial wavefunction file for multiplicity 1: [MOLCAS.1.RasOrb.init] (autocomplete enabled) ../MOLCAS.RasOrb
Initial wavefunction file for multiplicity 3: [MOLCAS.3.RasOrb.init] (autocomplete enabled) ../MOLCAS.RasOrb

  ================================================================================
||                                    PYSHARC                                   ||
  ================================================================================

The chosen interface can be run with PYSHARC.
PYSHARC runs the SHARC dynamics directly within Python (with C and Fortran extension)
with minimal file I/O for maximum performance.
Setup for PYSHARC? [True] <ENTER>

  ================================================================================
||                          Content of output.dat files                           ||
  ================================================================================


SHARC or PYSHARC can produce output in ASCII format (all features supported currently)
or in NetCDF format (more efficient file I/O, some features currently not supported).
Write output in NetCDF format? [True] no

Do you want to write the gradients to the output.dat file ?
Write gradients? [False] <ENTER>

Do you want to write the non-adiabatic couplings (NACs) to the output.dat file ?
Write NACs? [False] <ENTER>

Do you want to write property matrices to the output.dat file  (e.g., Dyson norms)?
Write property matrices? [False] <ENTER>

Do you want to write property vectors to the output.dat file  (e.g., TheoDORE results)?
Write property vectors? [False] <ENTER>

Do you want to write the overlap matrix to the output.dat file ?
Write overlap matrix? [True] <ENTER>

Do you want to modify the output.dat writing stride?
Modify stride? [False] <ENTER>


  ================================================================================
||                                 Run mode setup                                 ||
  ================================================================================


-------------------------Run script------------------------
```

```
This script can generate the run scripts for each trajectory in two modes:

  - In the first mode, the calculation is run in subdirectories of the current directory.

  - In the second mode, the input files are transferred to another directory (e.g. a local scratch
  directory), the calculation is run there, results are copied back and the temporary directory is
  deleted. Note that this temporary directory is not the same as the scratchdir employed by the interfaces.

Note that in any case this script will setup the input subdirectories in the current working directory.

In case of mode 1, the calculations will be run in:
/user/mai/Documents/NewSHARC/SHARC_4.0/TUTORIAL/traj

Use mode 1 (i.e., calculate here)? [True] <ENTER>

----------------------Submission script--------------------

During the setup, a script for running all initial conditions sequentially in batch mode is generated.
Additionally, a queue submission script can be generated for all initial conditions.

Generate submission script? [False] <ENTER>


########################Full input#########################

select_directly          True
ninit                    20
natom                    6
repr                     MCH
diag                     False
eref                     -94.41294554
eharm                    0.0
initf                    <_io.TextIOWrapper name='../initconds.excited' mode='r' encoding='UTF-8'>
states                   [4, 0, 3]
nstates                  13
charge                   [0, 1, 0]
statemap                 {1: [1, 1, 0.0], 2: [1, 2, 0.0], 3: [1, 3, 0.0], 4: [1, 4, 0.0], 5: [3, 1, -1.0],
                         6: [3, 2, -1.0], 7: [3, 3, -1.0], 8: [3, 1, 0.0], 9: [3, 2, 0.0],
                         10: [3, 3, 0.0], 11: [3, 1, 1.0], 12: [3, 2, 1.0], 13: [3, 3, 1.0]}
actstates                [4, 0, 3]
isactive                 [True, True, True, True, True, True, True, True, True, True, True, True, True]
show_content             True
n_issel                  [0, 0, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
setupstates              3
firstindex               1
ntraj                    9
needed_requests          'overlap', 'soc', 'h', 'dm', 'grad'
method                   tsh
tmax                     100.0
dtstep                   0.5
integrator               2
nsubstep                 25
kill                     False
surf                     diagonal
soc                      True
coupling                 4
phases_from_interface    False
gradcorrect              1
ekincorrect              2
```

```
reflect                 1
decoherence             ['edc', '0.1']
hopping                 sharc
force_hops              False
force_hops_dE           9999.0
scaling_for_sharc       False
damping                 False
atommaskarray           None
sel_g                   True
sel_t                   False
eselect                 0.1
rattle                  False
use_thermostat          False
laser                   False
dipolegrad              False
ion                     False
theodore                False
pysharc                 True
netcdf                  False
netcdf_separate         False
write_grad              False
write_NAC               False
write_property2d        False
write_property1d        False
write_overlap           True
stride                  [1]
log.infolevel           2
cwd                     /user/mai/Documents/NewSHARC/SHARC_4.0/TUTORIAL/2_full/Tutorial/traj
here                    True
copydir                 /user/mai/Documents/NewSHARC/SHARC_4.0/TUTORIAL/2_full/Tutorial/traj
qsub                    False

Do you want to setup the specified calculations? [True] <ENTER>

Do you want to link the interface files? [False] <ENTER>

  ================================================================================
||                           Setting up directories...                          ||
  ================================================================================

Do you want to see the input for the first trajectory? [False] <ENTER>
Do you want to add keywords to the input of all trajectories? [False] <ENTER>

Progress: [================================================] 100%

12 trajectories setup, last initial condition was 10 in state 3.
```

The script creates for each of the initial states ("States to setup the dynamics") a directory called **<Mult>_<Num>**, e.g., **Singlet_2/**, which contains the input for all trajectories starting in that state. Each of these directories contains subdirectories named **TRAJ_00001/**, **TRAJ_00002/**, etc. Note that these numbers are not consecutive: if an initial condition has not been selected, the number will be missing. Each subdirectory contains the SHARC input (consisting of the files **input**, **geom**, and **veloc**), the directories **QM/** and **restart/**, and the run script for the trajectory, **run.sh**.

For the purposes of the tutorial it is sufficient to only calculate one trajectory. Change to the subdirectory of one of the trajectories and execute it.

```
user@host> cd Singlet_2/TRAJ_00002
user@host> sh run.sh&
```

```
user@host> tailf output.lis
```

While the trajectory is running, you can watch its progress in the file **output.lis** (short output listing). For each timestep, it contains the currently occupied diagonal state (and approximate MCH state), the kinetic, potential and total energy, the RMS gradient, the state dipole and spin expectation values of the currently occupied diagonal state and the time needed for this step. Surface hopping events are also mentioned in this file.

Besides the **output.lis** file, SHARC creates the files **output.log**, **output.xyz** and **output.dat**. The file **output.log** contains mainly parsing information of the input file parsing and a list of internal steps of the dynamics simulation. With sufficiently high **printlevel** in the SHARC input file, the log file may also contain debug information in various detail, but with the default setting, no relevant information is printed. The file **output.dat** contains for each timestep the most important matrices and vectors. This information can be used to calculate the excited-state energies, populations, hopping probabilities and a large number of expectation values. See below for the usage of **data_extractor.x** and **make_gnuplot.py**, which can be used for plotting the mentioned quantities. Finally, **output.xyz** contains the cartesian coordinates of all atoms for each timestep. This file can be opened with any program capable of processing xyz files, like MOLDEN, MOLEKEL and GABEDIT. Additionally, the geometries can be analyzed with the programs **geo.py**, which is a command line tool to extract internal coordinates from such an xyz file, **trajana_nma.py**, and **trajana_essdyn.py**.

See Section 3.2 (p. 38) in the manual.

2.8 Analyzing a single trajectory

We will first discuss the analysis of a single trajectory based on the output files. Later (section 2.9) we will also analyze ensemble properties.

If you are not in the directory for the trajectory **TRAJ_00002/**, change to this directory:

```
user@host> cd Singlet_2/TRAJ_00002
```

2.8.1 Data extraction and plotting

The file **output.dat** contains the Hamiltonian, transformation matrix, dipole matrices, coefficients, hopping probabilities, kinetic energy and random number from the surface hopping procedure in a compressed form. The program **data_extractor.x** can be used to generate data tables, which can then be plotted.

See Section 7.22 (p. 208) in the manual.

```
user@host> $SHARC/data_extractor.x output.dat
```

The program creates a subdirectory called **output_data/**. With the default settings, the following files will be created:

See Section 7.27 (p. 214) in the manual.

- **coeff_diab.out**, **coeff_class_diab.out**, **coeff_mixed_diab.out** contain the coefficients and populations in the diabatic representation (only approximate).
- **coeff_diag.out**, **coeff_class_diag.out**, **coeff_mixed_diag.out** contain the coefficients and populations in the diagonal representation.
- **coeff_MCH.out**, **coeff_class_MCH.out**, **coeff_mixed_MCH.out** contain the coefficients and populations in the MCH representation.
- **energy.out** contains kinetic, current potential, total and potential energy of all excited states.
- **fosc.out** contains the oscillator strengths of the current state and all excited states.
- **fosc_act.out** contains the oscillator strengths of all states relative to the active state.
- **spin.out** contains the total spin expectation values of the current state and all excited states.
- **prob.out** contains the surface hopping random number and the hopping probabilities in the diagonal representation.
- **expec.out** contains the content of **energy.out**, **fosc.out** and **spin.out** in one file (for plotting).
- **expec_MCH.out** is analogue to **expec.out**, except all data is given in the MCH representation (except the active state energy).

In order to plot the content of these files in an efficient manner, **gnuplot** can be used. Use

```
user@host> $SHARC/make_gnuscript.py  4  0  3 > plot.gp
```

to create a **gnuplot** script with the correct state numbering and labeling. Execute

```
user@host> gnuplot plot.gp
```

to plot energies, populations and hopping probabilities (Use **<ENTER>** to continue with the next plot). In figures 2.2, 2.4, 2.5 and 2.6 the output for trajectory **TRAJ_00002/** for the first 100 fs is given.

Discussion of Figure 2.2 In figure 2.2, the potential energies of all states included in the dynamics depending on time is given. The total energy is given by the thin black line (around 12 eV) and the currently occupied state is marked with black circles. Each state is represented by a line that is colored in two ways, with an inner core color and an outer colored contour. The inner color encodes the oscillator strength of the state at each instant of time. Dark states are light grey, while brighter states are given in grey, dark grey, orange, red, magenta or blue, in order of increasing oscillator strength. The outer color encodes the total spin expectation value. Singlets are blue, triplets red and states with mixed singlet-triplet character are green. Since in the methaniminium cation spin-orbit coupling is negligible, in the figure only blue and red contours are visible.

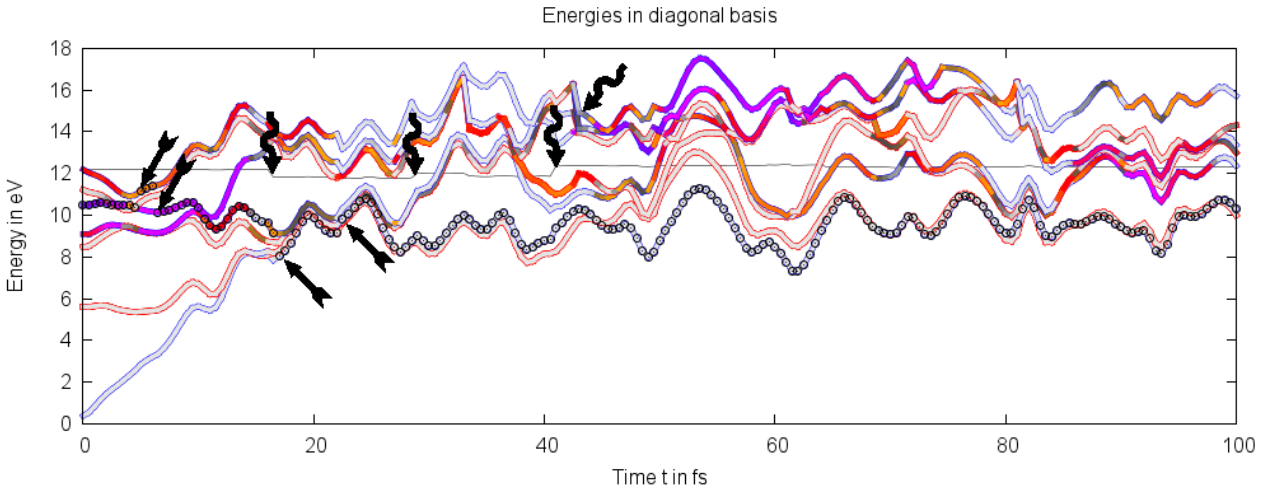

Figure 2.2: Plot of the potential energies for trajectory **`TRAJ_00002/`** with 4 singlet and 3 triplet states. Straight arrows indicate hopping events discussed in the text, wiggly arrows indicate problems with energy conservation/continuity. **Your result might numerically differ.**

In the figure, the trajectory starts in the very bright singlet state slightly above 10 eV (the S_3). The ground state (grey/blue line), the T_1 and T_2 (grey/red lines), and the S_1 (red/blue line) are at lower energies, whereas the T_3 (grey/red line) and the S_4 (red/blue line) are at higher energies. All important nonadiabatic events occur within the first 25 fs. In the figure, four hopping events are marked with straight arrows (at 5.0 fs, 6.5 fs, 17.0 fs, and 22.5 fs; note that these are not the only hopping events in the figure). It can be seen that the trajectory briefly switches to the S_4 state but quickly returns to S_3. Subsequently, it changes to S_1 and then to S_0 (at 17.0 fs). The hop at 22.5 fs is due to a crossing of the S_0 with the T_1, and since the T_1 is not populated, the trajectory hops to stay in the singlet state. For the remainder of the simulation time, the trajectory stays in S_0, occasionally performing a hop at S_0/T_1 crossings. The very high potential energy indicates that the trajectory did not simply return to the ground state equilibrium geometry.

In the figure, four wiggly arrows indicate possible problems in the trajectory. Three of these arrows (at 17.0 fs, 29.0 fs, and 41.0 fs) point to time steps where the total energy was not well conserved. In general, this can have different reasons (e.g., wrong gradients, too large time steps, convergence problems), but here all three cases are due to abrupt changes in the active space because the highest singlet state crossed with another state. Often, this problem can be circumvented by a good choice of the active space and the number of roots for state averaging. The fourth wiggly arrow (at 43.0 fs) points to a time step where the same problem happens to the triplet states; note how the T_3 energy suddenly changes. Since in MOLCAS each multiplicity uses its own active space, this does not affect the singlet states and thus the trajectory. However, in systems with larger spin-orbit couplings, such state switches might lead to unphysical population transfer to the triplet states.

Ultimately, the user is responsible to check the trajectories for such problematic time steps. For the purposes of the tutorial, we will ignore these energy conservation problems. Note that this trajectory checking can also be carried out while the trajectories are still running; if a problematic trajectory is encountered, it can be terminated gracefully by creating an empty file called **`STOP`** in the run directory of that trajectory.

Discussion of Figure 2.3 In figure 2.3, the same data as in figure 2.2 is presented, the only difference being that in figure 2.3 all energies are relative to the energy of the lowest state. This is often useful if there are strong oscillations in the energies of all states that make it difficult to see the energy gaps between the states.

Note that in this plot, the grey line represents the difference between total energy and energy of the lowest state, and hence does not need to be a straight line.

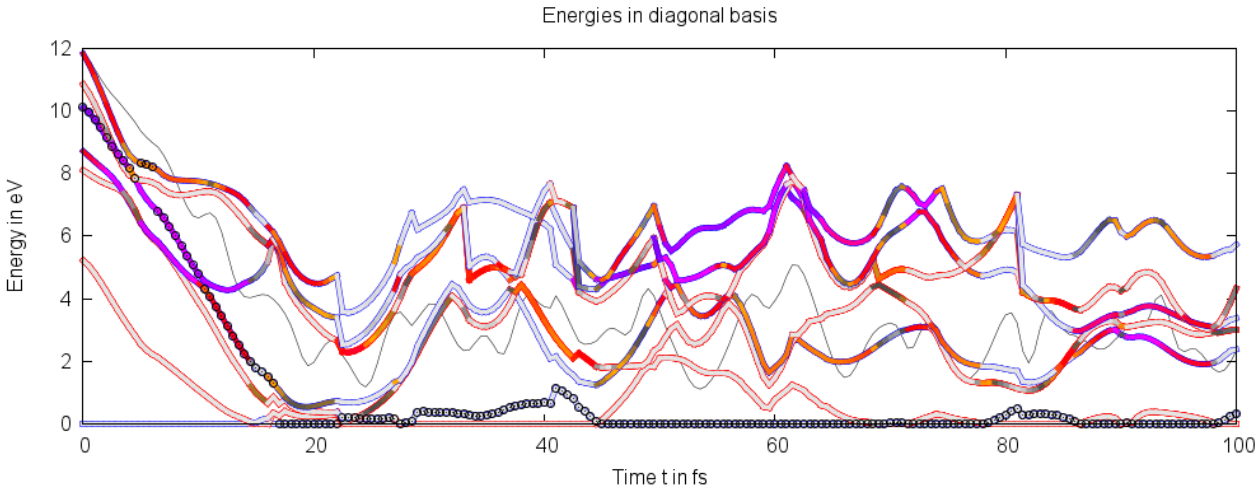

Figure 2.3: Plot of the *relative* potential energies for trajectory **`TRAJ_00002/`** with 4 singlet and 3 triplet states.

Discussion of Figure 2.4 In figure 2.4, the MCH populations are given depending on time. The system starts with 100% of the population in the S_2. Around 5 fs, part of the population is briefly transferred to the S_3. Then, at 10 fs population flows to the S_1, and at 17 fs to the S_0, where it stays for the remainder of the simulation. The population of the triplet states is approximately zero for all time steps, as was expected for this system.

It is instructive to observe the correlation between the population transfers in figure 2.4 with the hopping events in figure 2.2.

Discussion of Figure 2.5 In figure 2.5, the diagonal populations are given depending on time. The difference between the MCH and diagonal populations is due to the fact that the diagonal states are strictly ordered according to energy. This can be seen best in the second half of the figure, where population is occasionally exchanged (with 100% efficiency) between state 1 and state 4. These population transfers happen where the S_0 and T_1 states cross (e.g., before the crossing the S_0 is lower than T_1 and hence S_0 is state 1. After the crossing, S_0 becomes state 4, and the population is transferred to state 4 to conserve the spin multiplicity of the total wave function).

Note that in more complicated cases, the diagonal populations are of little use for interpretation purposes, so that most users will prefer to analyze the MCH populations.

Discussion of Figure 2.6 Figure 2.6 shows the surface hopping probabilities and the corresponding random numbers depending on time. In a nutshell, a surface hop happens whenever the random number lies within one of the colored bars. The color of the bar corresponds to the state into which the trajectory will hop. In the diagram, there are several hopping probabilities close to unity. This corresponds to the near-complete population transfer during the crossing of the singlet and triplet states.

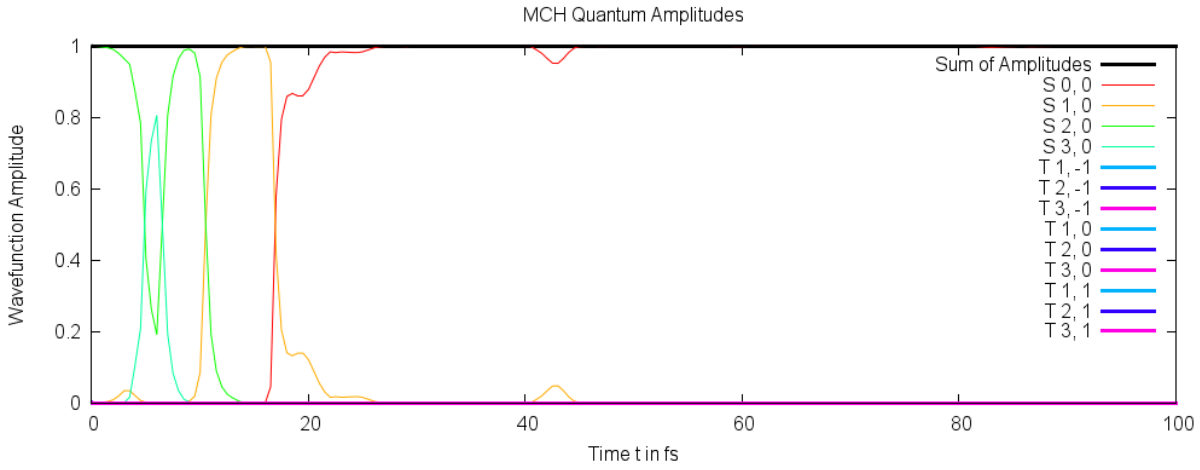

Figure 2.4: Plot of the excited-state populations in the MCH representation.

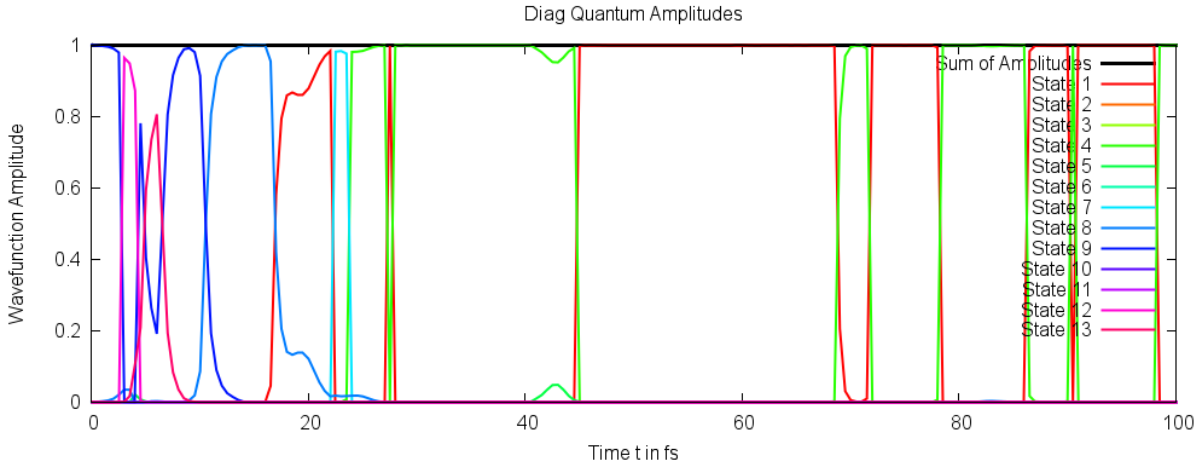

Figure 2.5: Plot of the excited-state populations in the diagonal representation.

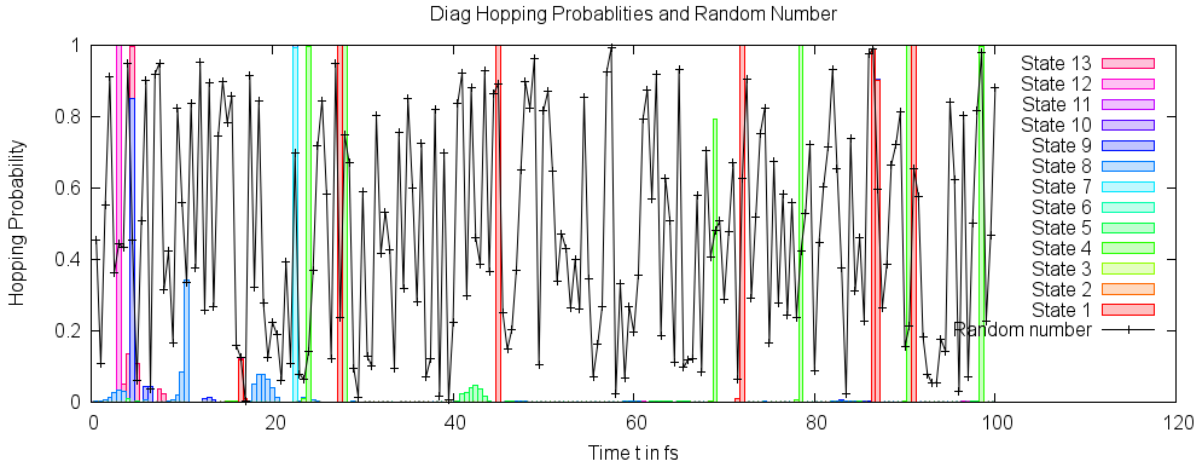

Figure 2.6: Plot of the hopping probabilities in the diagonal representation. Additionally, the random number for the surface hopping procedure is given.

2.8.2 Analyzing internal coordinates

The file **output.xyz** contains the cartesian coordinates of all timesteps. Oftentimes, one is interested in the variation of certain internal coordinates (like bond lengths, angles, etc.) during the dynamics. The SHARC tool **geo.py** can quickly calculate these values. Invoke the program and enter the internal coordinate specifications:

See Section 7.28 (p. 215) in the manual.

```
user@host> $SHARC/geo.py -g output.xyz -t 0.5
```

```
Enter the internal coordinate specifications:
r 1 2
d 3 1 2 4
end
Number of internal coordinate requests:   2
Number of geometries:    200
FINISHED!
```

See Section 8.12 (p. 251) in the manual.

The **-g** option specifies the filename of the input xyz geometry file, while the **-t** option specifies the timestep. **geo.py** writes the results to standard out, so redirect the output to some file:

```
user@host> $SHARC/geo.py -g output.xyz -t 0.5 > Geo.out
```

The file **Geo.out** contains a table with the specified internal coordinates:

```
#                1|                2|                3|
#             time|           r  1  2|       d  6  1  2  5|
            0.0000             1.3024              16.2632
            0.5000             1.3135              18.5857
            1.0000             1.3293              20.6828
                ⋮                  ⋮                   ⋮
```

Use GNUPLOT to plot this table.

```
user@host> gnuplot
```

The file **Geo.out** contains a table with the specified internal coordinates:

```
gnuplot> p "Geo.out" u 1:2 w l     # Plot column 2 versus 1
gnuplot> p "Geo.out" u 1:3 w l     # Plot column 3 versus 1
```

The results are shown in figures 2.7 and 2.8.

Discussion of the internal coordinates In figures 2.7 the C=N bond length is plotted over time. It can be easily seen that after excitation to the $\pi\pi^*$ state, the C=N bond stretches strongly and reaches more than 3 Åafter 100 fs. This is a clear sign that the $CH_2NH_2^+$ molecule is dissociating in this trajectory.

In figure 2.8, the dihedral angle H–C=N–H is plotted in degrees (confined to the interval $[-180°, 180°]$). It can be seen that after excitation the molecule undergoes torsion around the central bond.

In order to confirm these findings, it is recommended that you load **output.xyz** into MOLDEN (or another program) to watch the trajectory as a movie.

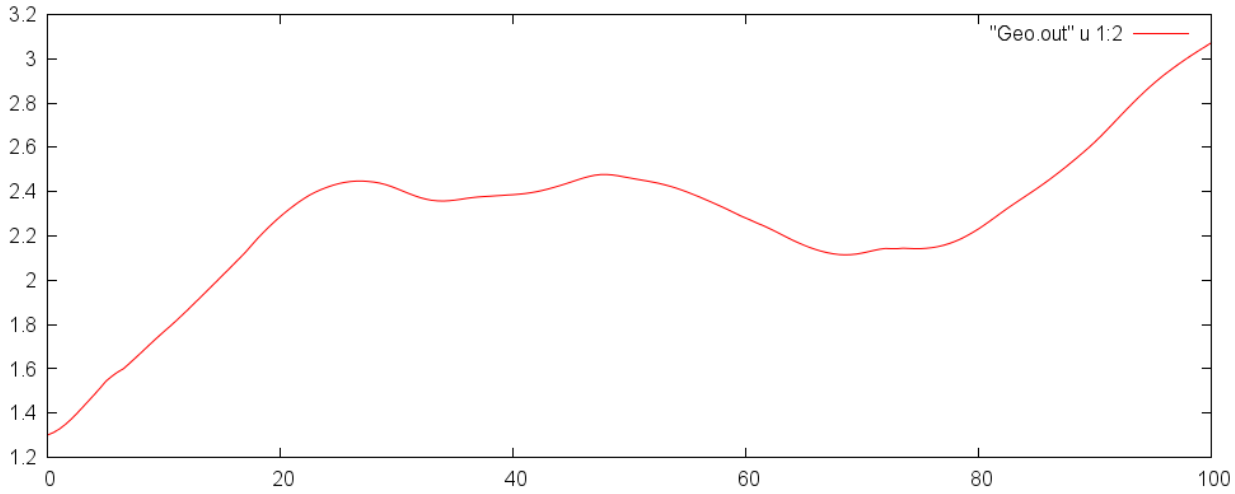

Figure 2.7: Value of the C=C bond length during the simulation.

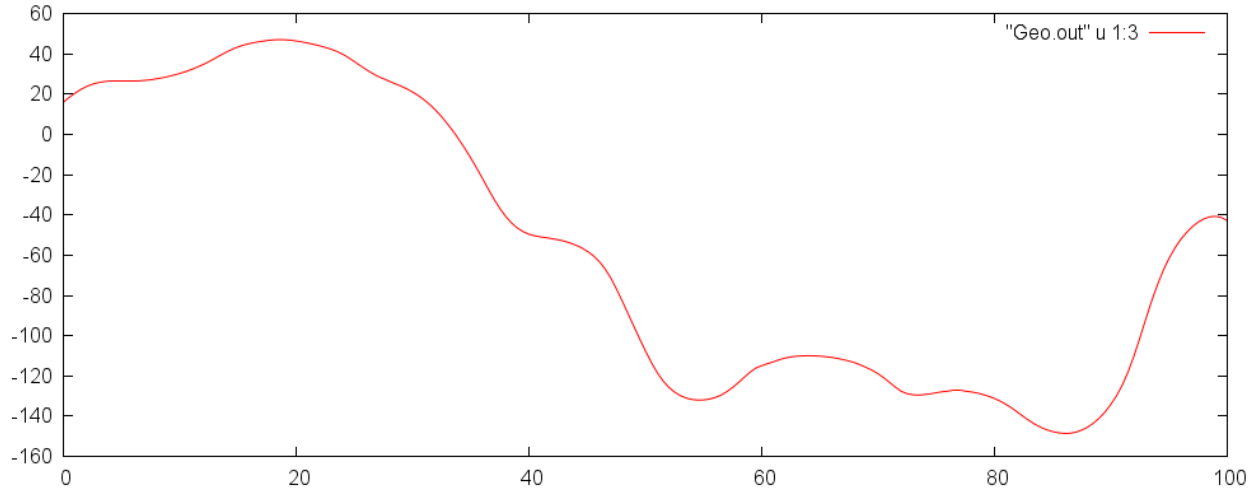

Figure 2.8: Value of one of the H–C=C–H dihedrals during the simulation.

2.9 Analyzing the Ensemble

For these analysis the tutorial assumes that you ran all nine trajectories (**TRAJ_00002/** to **TRAJ_00009/**).

2.9.1 Ensemble Diagnostics

It is always a good idea to inspect the trajectories before starting with the ensemble analysis, because within the large ensemble it might not be possible to spot problems of a single trajectory. There are two ways to inspect the trajectories—either manually checking them as described above, or the ensemble diagnostics tool, **diagnostics.py**. This script performs a number of sanity checks for all trajectories (file existence, consistency, energy conservation, intruder states), and allows automatically marking problematic trajectories to exclude them from the analysis steps.

See Section 7.21 (p. 207) in the manual.

If you are still in the directory **TRAJ_00002/**, go back to the root directory of the ensemble. Then, execute **diagnostics.py**:

```
user@host> cd ../..
user@host> $SHARC/diagnostics.py
```

```
  ================================================================================
||                                                                              ||
||             Diagnostic tool for trajectories from SHARC dynamics             ||
||                                                                              ||
||                    Author: Sebastian Mai and Moritz Heindl                   ||
||                                                                              ||
||                                  Version:4.0                                 ||
||                                    01.09.24                                  ||
||                                                                              ||
  ================================================================================


This script reads output.dat files from SHARC trajectories and checks:
* missing files
* normal termination
* total energy conservation
* total population conservation
* discontinuities in potential and kinetic energy

--------------------Paths to trajectories-------------------

Please enter the paths to all directories containing the "TRAJ_0XXXX" directories.
E.g. Sing_2/ and Sing_3/.
Please enter one path at a time, and type "end" to finish the list.
Path:  [end] (autocomplete enabled) Singlet_2/
['TRAJ_00005', 'TRAJ_00004', 'TRAJ_00008', 'TRAJ_00009', 'TRAJ_00002',
'TRAJ_00006', 'TRAJ_00010', 'TRAJ_00007', 'TRAJ_00003']
Found 9 subdirectories in total.

Path:  [end] (autocomplete enabled) <ENTER>

'paths': ['Singlet_2/']
Total number of subdirectories: 9

['nstates', '4', '0', '3']
--------------------Diagnostic settings--------------------

Please, adjust the diagnostic settings according to your preferences.
```

```
You can use the following commands:
show            Prints the current settings
help            Prints explanations for the keys
end             Save and continue
<key> <value>   Adjust setting.

Current settings:
        missing_output : True
       missing_restart : True
    normal_termination : True
         always_update : False
           etot_window : 0.2
             etot_step : 0.1
             epot_step : 0.7
             ekin_step : 0.7
            pop_window : 1e-07
            hop_energy : 1.0
             intruders : True
        extractor_mode : default

?  [end] <ENTER>
########################Full input########################

paths                       ['Singlet_2/']
settings                    {'missing_restart': True,
                             'etot_step': 0.1,
                             'hop_energy': 1.0,
                             'epot_step': 0.7,
                             'ekin_step': 0.7,
                             'intruders': True,
                             'pop_window': 1e-07,
                             'missing_output': True,
                             'normal_termination': True,
                             'etot_window': 0.2}

Do you want to do the specified analysis? [True] <ENTER>

Checking the directories...
~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Singlet_2/TRAJ_00002 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    Output files:     .lis .. .log .. .dat .. .xyz .. OK
    Restart files:    ctrl .. traj .. restart/ ..     OK
    Progress:         [========================]     100.0 of 100.0 fs
    Status:                                           FINISHED
    Energy:           Large dE during hop             at 6.50 fs      # a hop over >1eV might be suspicious
    Population:                                       OK
    Intruder states:                                  OK



~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Singlet_2/TRAJ_00003 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    Output files:     .lis .. .log .. .dat .. .xyz .. OK
    Restart files:    ctrl .. traj .. restart/ ..     OK
    Progress:         [========================]     100.0 of 100.0 fs
    Status:                                           FINISHED
    Energy:           Large dE during hop             at 11.50 fs
    Population:                                       OK
    Intruder states:                                  OK
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Singlet_2/TRAJ_00004 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    Output files:     .lis .. .log .. .dat .. .xyz .. OK
    Restart files:    ctrl .. traj .. restart/ ..     OK
    Progress:         [========================]     100.0 of 100.0 fs
    Status:                                           FINISHED
    Energy:           Large step in Epot              at 13.50 fs      # time step might be too long here
    Population:                                       OK
    Intruder states:                                  OK



~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Singlet_2/TRAJ_00005 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    Output files:     .lis .. .log .. .dat .. .xyz .. OK
    Restart files:    ctrl .. traj .. restart/ ..     OK
    Progress:         [========================]     100.0 of 100.0 fs
    Status:                                           FINISHED
    Energy:           Large step in Etot              at 29.00 fs    # problem due to active space switch
    Population:                                       OK
    Intruder states:                                  OK



~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Singlet_2/TRAJ_00006 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    Output files:     .lis .. .log .. .dat .. .xyz .. OK
    Restart files:    ctrl .. traj .. restart/ ..     OK
    Progress:         [========================]     100.0 of 100.0 fs
    Status:                                           FINISHED
    Energy:           Large step in Etot              at 3.50 fs
    Population:                                       OK
    Intruder states:                                  OK



~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Singlet_2/TRAJ_00007 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    Output files:     .lis .. .log .. .dat .. .xyz .. OK
    Restart files:    ctrl .. traj .. restart/ ..     OK
    Progress:         [========================]     100.0 of 100.0 fs
    Status:                                           FINISHED
    Energy:           Large step in Etot              at 21.00 fs
    Population:                                       OK
    Intruder states:                                  OK



~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Singlet_2/TRAJ_00008 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    Output files:     .lis .. .log .. .dat .. .xyz .. OK
    Restart files:    ctrl .. traj .. restart/ ..     OK
    Progress:         [========================]     100.0 of 100.0 fs
    Status:                                           FINISHED
    Energy:           Large step in Epot              at 14.50 fs
    Population:                                       OK
    Intruder states:                                  OK
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Singlet_2/TRAJ_00009 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    Output files:     .lis .. .log .. .dat .. .xyz .. OK
    Restart files:    ctrl .. traj .. restart/ ..     OK
    Progress:         [=================        ]     71.5 of 100.0 fs
    Status:                                           CRASHED   # crashed after 71 fs (convergence problem?)
    Energy:           Large step in Etot              at 46.00 fs
    Population:                                       OK
    Intruder states:                                  OK



~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Singlet_2/TRAJ_00010 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    Output files:     .lis .. .log .. .dat .. .xyz .. OK
    Restart files:    ctrl .. traj .. restart/ ..     OK
    Progress:         [=====                    ]     20.5 of 100.0 fs
    Status:                                           CRASHED   # crashed after 20 fs (convergence problem?)
    Energy:                                           OK
    Population:                                       OK
    Intruder states:                                  OK   # but no problems found



================================== Summary ==================================

                    Trajectory Files? Status Length  T_use
                                              (fs)   (fs)

          Singlet_2/TRAJ_00006     OK FINISH  100.0    3.5   [------------------------]
          Singlet_2/TRAJ_00002     OK FINISH  100.0    6.5   [=-----------------------]
          Singlet_2/TRAJ_00003     OK FINISH  100.0   11.5   [==----------------------]
          Singlet_2/TRAJ_00004     OK FINISH  100.0   13.5   [===---------------------]
          Singlet_2/TRAJ_00008     OK FINISH  100.0   14.5   [===---------------------]
          Singlet_2/TRAJ_00010     OK  CRASH   20.5   20.0   [=====                   ]
          Singlet_2/TRAJ_00007     OK FINISH  100.0   21.0   [=====-------------------]
          Singlet_2/TRAJ_00005     OK FINISH  100.0   29.0   [=======-----------------]
          Singlet_2/TRAJ_00009     OK  CRASH   71.5   46.0   [===========------        ]

This many trajectories can be used for an analysis up to the given time:
up to  20.0 fs:      4  trajectories
up to  40.0 fs:      1  trajectories
up to  60.0 fs:      0  trajectories
up to  80.0 fs:      0  trajectories
up to  100.0 fs:     0  trajectories

------------------- Trajectory Flagging -------------------

You can now flag the trajectories according to their maximum usable time.
In this way, you can restrict the analysis tools to the set of trajectories with sufficient simulation time.

Do you want to flag the trajectories? [True] no
```

In the output, **`diagnostics.py`** prints for each directory a summary of the performed checks and their results. For example, for trajectory **`Singlet_2/TRAJ_00002/`**, the script reports that all output and restart files are there, and that the trajectory ran for 100 out of 100 fs (finished). It also reports that at 6.5 fs there is a

problem because during a hop the potential energy changed by a large amount (it prints the first time that any problem occurs, there might be more problems occurring later). Recalling Figure 2.2, at this time the trajectory performed a hop from S_3 back to S_2. Such hops across large energy differences might be suspicious because they should be physically unlikely (nonadiabatic coupling becomes stronger the closer two states are).

For trajectory **Singlet_2/TRAJ_00004/**, the script reports that the potential energy changed by a large amount within one step. This could be a sign of an active space switch which leads to a large change in all state energies. In the case of **Singlet_2/TRAJ_00004/**, however, it is simply because the potential energy surface is extremely steep and the time step possibly too long to properly handle this situation. The user is invited to generate the energy plot for **Singlet_2/TRAJ_00004/** and inspect this situation.

For trajectory **Singlet_2/TRAJ_00005/**, the script reports that the total energy changed too much within one step. As already discussed for Figure 2.2, this can happen if the active space composition changes suddenly. Another possible reason could be that the computed gradient was incorrect. It is possible to distinguish between these cases by checking whether potential and total energy both show the sudden change (then it is likely a problem with the energy computation) or whether only the total energy changes while the potential energies are smooth (then it is likely a problem with the gradients).

For trajectories **Singlet_2/TRAJ_00009/** and **Singlet_2/TRAJ_00010/**, the script reports **CRASHED**, which is most likely due to convergence problems within MOLCAS.

Note that a large number of reported problems is often a sign that the method is badly chosen, e.g., the active space, basis set, state-averaging, etc. These problems are typically much more common with multi-configurational/multi-reference methods (like CASSCF) than with single-reference methods (TD-DFT, ADC(2)). It is ultimately in the responsibility of the user to check and avoid these problems, or to judge whether these problems can be ignored because they do not affect the conclusions drawn from the simulations.

For the tutorial, we will from here ignore these problems by rerunning **diagnostics.py** with relaxed check thresholds.

```
user@host> $SHARC/diagnostics.py
```

```
  ================================================================================
||                                                                              ||
||             Diagnostic tool for trajectories from SHARC dynamics             ||
||                                                                              ||
||                    Author: Sebastian Mai and Moritz Heindl                   ||
||                                                                              ||
||                                  Version:4.0                                 ||
||                                    01.09.24                                  ||
||                                                                              ||
  ================================================================================


This script reads output.dat files from SHARC trajectories and checks:
* missing files
* normal termination
* total energy conservation
* total population conservation
* discontinuities in potential and kinetic energy

--------------------Paths to trajectories-------------------

Please enter the paths to all directories containing the "TRAJ_0XXXX" directories.
E.g. Sing_2/ and Sing_3/.
Please enter one path at a time, and type "end" to finish the list.
Path:  [end] (autocomplete enabled) Singlet_2/
```

```
['TRAJ_00005', 'TRAJ_00004', 'TRAJ_00008', 'plot.gp', 'TRAJ_00009', 'TRAJ_00002',
'TRAJ_00006', 'TRAJ_00010', 'TRAJ_00007', 'TRAJ_00003']
Found 9 subdirectories in total.

Path:  [end] (autocomplete enabled) <ENTER>

'paths': ['Singlet_2/']
Total number of subdirectories: 9

['nstates', '4', '0', '3']
--------------------Diagnostic settings--------------------

Please, adjust the diagnostic settings according to your preferences.
You can use the following commands:
show            Prints the current settings
help            Prints explanations for the keys
end             Save and continue
<key> <value>   Adjust setting.

Current settings:
        missing_output : True
       missing_restart : True
    normal_termination : Truee
         always_update : False
           etot_window : 0.2
             etot_step : 0.1
             epot_step : 0.7
             ekin_step : 0.7
            pop_window : 1e-07
            hop_energy : 1.0
             intruders : True
        extractor_mode : default

?  [end] etot_window 3.0    # extremely large thresholds
?  [end] etot_step 3.0    # to ignore all problems
?  [end] epot_step 3.0    #
?  [end] ekin_step 3.0    #
?  [end] hop_energy 3.0    #
?  [end]
#########################Full input#########################

paths                      ['Singlet_2/']
settings                   {'missing_restart': True,
                            'etot_step': 3.0,
                            'hop_energy': 3.0,
                            'epot_step': 3.0,
                            'ekin_step': 3.0,
                            'intruders': True,
                            'pop_window': 1e-07,
                            'missing_output': True,
                            'normal_termination': True,
                            'etot_window': 3.0}

Do you want to do the specified analysis? [True]

Checking the directories...

# ... omitting trajectory summaries ...

================================== Summary ==================================
```

```
                  Trajectory Files? Status Length  T_use
                                              (fs)   (fs)

          Singlet_2/TRAJ_00010     OK  CRASH   20.5   20.0   [=====                   ]
          Singlet_2/TRAJ_00009     OK  CRASH   71.5   71.0   [=================       ]
          Singlet_2/TRAJ_00004     OK FINISH  100.0  100.0   [========================]
          Singlet_2/TRAJ_00005     OK FINISH  100.0  100.0   [========================]
          Singlet_2/TRAJ_00006     OK FINISH  100.0  100.0   [========================]
          Singlet_2/TRAJ_00007     OK FINISH  100.0  100.0   [========================]
          Singlet_2/TRAJ_00002     OK FINISH  100.0  100.0   [========================]
          Singlet_2/TRAJ_00003     OK FINISH  100.0  100.0   [========================]
          Singlet_2/TRAJ_00008     OK FINISH  100.0  100.0   [========================]

This many trajectories can be used for an analysis up to the given time:
up to  20.0 fs:      9  trajectories
up to  40.0 fs:      8  trajectories
up to  60.0 fs:      8  trajectories
up to  80.0 fs:      7  trajectories
up to  100.0 fs:      7  trajectories

-------------------- Trajectory Flagging -------------------

You can now flag the trajectories according to their maximum usable time.
In this way, you can restrict the analysis tools to the set of trajectories with sufficient simulation time.

Do you want to flag the trajectories? [True] <ENTER>
Threshold for T_use (fs): [100.0] <ENTER>

Flagged  7 trajectories for analysis.
Excluded 2 trajectories from analysis.
```

With the relaxed thresholds, all trajectories are reported to have no problems. Nevertheless, two trajectories are shorter than 100 fs because they crashed before. Now one has two choices—either analyze all nine trajectories, but only to the length of the shortest one (20 fs), or neglecting trajectories so that a longer simulation time can be analyzed.

The choice suggested by **`diagnostics.py`** is the latter, because then a total of 700 fs can be analyzed (7×100 fs, vs. the alternative choices of 8×71 fs or 9×20 fs). The two shorter trajectories are then marked by **`diagnostics.py`** by creating a file called **`DONT_ANALYZE`** in their directories. All other analysis scripts will then ignore those trajectories. With this, the ensemble is prepared for the ensemble analysis procedures.

2.9.2 Ensemble Populations

See Section 7.31 (p. 218) in the manual.

Among the main results of a SHARC simulation are the time-dependent excited-state populations within the simulated ensemble. In order to obtain these populations, the populations of all trajectories have to be summed up and normalized to the number of trajectories.

The script **`populations.py`** can be used to calculate various excited-state populations. There are several concepts, e.g.:

- Count, for each timestep ,the number of trajectories in each classical state. These are the "classical" populations.
- For each timestep, calculate the sum of the squares of the quantum amplitudes of each state. These sums are called the "quantum" populations.
- Count, for each timestep, the number of trajectories whose expectation values are within a certain interval. This can be used to obtain populations which correspond to certain classes of states (e.g. count all trajectories with large oscillator strength to find the approximate $\pi\pi^*$ population).

See Section 8.5 (p. 244) in the manual.

Note that the rigorous computation of electronic populations including a change of representation is a complex topic, which explains the large number of possible population modes offered by **`populations.py`**. The modes used below (mode 3 for classical populations and mode 9 for quantum populations) should work in most cases. However, when spin-orbit mixing is strong or when looking into diabatic populations, consider using the Wigner-transformed populations after reading the relevant section in the manual.

In the following, an example is given on the usage of **`populations.py`**, and subsequently the results of using the different concepts are discussed.

```
user@host> $SHARC/populations.py
```

```
Script for population computation started...


  ================================================================================
||                                                                              ||
||                    Reading populations from SHARC dynamics                   ||
||                                                                              ||
||                             Author: Sebastian Mai                            ||
||                                                                              ||
||                                  Version:4.0                                 ||
||                                    01.09.24                                  ||
||                                                                              ||
  ================================================================================


This script reads output.lis files and calculates ensemble populations
(e.g. based on the classically occupied state or based on the quantum amplitudes).

--------------------Paths to trajectories------------------

Please enter the paths to all directories containing the "TRAJ_0XXXX" directories.
E.g. Sing_2/ and Sing_3/.
Please enter one path at a time, and type "end" to finish the list.
Path:  [end] (autocomplete enabled) Singlet_2/
['TRAJ_00005', 'TRAJ_00004', 'TRAJ_00008', 'plot.gp', 'TRAJ_00009', 'TRAJ_00002',
'TRAJ_00006', 'TRAJ_00010', 'TRAJ_00007', 'TRAJ_00003']
Found 9 subdirectories in total.

Path:  [end] (autocomplete enabled) <ENTER>
```

```
Total number of subdirectories: 9

-----------------------Analyze Mode-----------------------

This script can analyze the classical populations in different ways:
1  Number of trajectories in each diagonal state                                  from output.lis
2  Number of trajectories in each (approximate) MCH state                         from output.lis
3  Number of trajectories in each (approximate) MCH state (multiplets summed up)  from output.lis
4  Number of trajectories whose total spin value falls into certain intervals     from output.lis
5  Number of trajectories whose dipole moment falls into certain intervals        from output.lis
6  Number of trajectories whose oscillator strength falls into certain intervals  from output_data/fosc.out

It can also sum the quantum amplitudes:
7  Quantum amplitudes in diagonal picture                                  from output_data/coeff_diag.out
8  Quantum amplitudes in MCH picture                                       from output_data/coeff_MCH.out
9  Quantum amplitudes in MCH picture (multiplets summed up)                from output_data/coeff_MCH.out

It can also transform the classical diagonal populations to MCH basis:
12  Transform diagonal classical populations to MCH
                                                                    from output_data/coeff_class_MCH.out
13  Transform diagonal classical populations to MCH (multiplets summed up)
                                                                    from output_data/coeff_class_MCH.out
14  Wigner-transform classical diagonal populations to MCH
                                                                    from output_data/coeff_mixed_MCH.out
15  Wigner-transform classical diagonal populations to MCH (multiplets summed up)
                                                                    from output_data/coeff_mixed_MCH.out

It can also compute diabatic populations:
20 Quantum amplitudes in diabatic picture                           from output_data/coeff_diab.out
21 Transform diagonal classical populations to diabatic             from output_data/coeff_class_diab.out
22 Wigner-transform classical diagonal populations to diabatic      from output_data/coeff_mixed_diab.out
Analyze mode: 3

----------------------Number of states----------------------

Please enter the number of states as a list of integers
e.g. 3 0 3 for three singlets, zero doublets and three triplets.
Number of states: [4 0 3] <ENTER>


-----------------------Normalization-----------------------

Normalize the populations? [True] <ENTER>

----------------------Simulation time----------------------

Up to which simulation time should the analysis be performed? (Trajectories which are shorter are
continued with their last values.)
Simulation time (in fs):  [1000.0] 100

-----------------Setup for bootstrapping?-----------------

The population data can be analyzed by fitting with a kinetic model (via make_fitscript.py).
In order to estimate errors for these time constants (via bootstrapping),
additional data needs to be saved here.
Save data for bootstrapping? [False] yes
Directory for data? [bootstrap_data/] (autocomplete enabled) <ENTER>

----------------------Gnuplot script----------------------
```

```
Gnuplot script? [False] yes
Gnuplot script filename? [populations.gp] (autocomplete enabled) pop_class.gp

########################Full input##########################

normalize                  True
paths                      ['Singlet_2/']
gnuplot_out                pop_class.gp
bootstrap                  False
gnuplot                    True
run_extractor              False
states                     [4, 0, 3]
statemap                   {1: [1, 1, 0.0, 1],
                            2: [1, 2, 0.0, 2],
                            3: [1, 3, 0.0, 3],
                            4: [1, 4, 0.0, 4],
                            5: [3, 1, -1.0, 5],
                            6: [3, 2, -1.0, 6],
                            7: [3, 3, -1.0, 7],
                            8: [3, 1, 0.0, 5],
                            9: [3, 2, 0.0, 6],
                            10: [3, 3, 0.0, 7],
                            11: [3, 1, 1.0, 5],
                            12: [3, 2, 1.0, 6],
                            13: [3, 3, 1.0, 7]}
run_extractor_full         False
mode                       3
maxtime                    100.0
nstates                    7
nmstates                   13

Do you want to do the specified analysis? [True] <ENTER>

Checking the directories...
Singlet_2//TRAJ_00005         OK
Singlet_2//TRAJ_00004         OK
Singlet_2//TRAJ_00008         OK
Singlet_2//TRAJ_00009         DETECTED FILE dont_analyze     # excluded
Singlet_2//TRAJ_00002         OK
Singlet_2//TRAJ_00006         OK
Singlet_2//TRAJ_00010         DETECTED FILE dont_analyze     # excluded
Singlet_2//TRAJ_00007         OK
Singlet_2//TRAJ_00003         OK
Number of trajectories: 7                                    # only 7 trajectories left
Found dt=0.500000, nsteps=201, nstates=7

Singlet_2//TRAJ_00005/output.lis                             200
Singlet_2//TRAJ_00004/output.lis                             200
Singlet_2//TRAJ_00008/output.lis                             200
Singlet_2//TRAJ_00002/output.lis                             200
Singlet_2//TRAJ_00006/output.lis                             200
Singlet_2//TRAJ_00007/output.lis                             200
Singlet_2//TRAJ_00003/output.lis                             200
Shortest trajectory: 100.000000
Longest trajectory: 100.000000
Number of trajectories: 7

Writing to pop.out ...
Gnuplot script written to "pop_class.gp"
Writing to bootstrap_data/ ...
```

The incoherent sum of the quantum amplitudes can be calculated with mode 9. Rerun **populations.py**.

```
user@host> $SHARC/populations.py
```

```
⋮          ⋮          ⋮          ⋮          ⋮          ⋮          ⋮

-----------------------Analyze Mode-----------------------

This script can analyze the classical populations in different ways:
1  Number of trajectories in each diagonal state                                  from output.lis
2  Number of trajectories in each (approximate) MCH state                         from output.lis
3  Number of trajectories in each (approximate) MCH state (multiplets summed up)  from output.lis
4  Number of trajectories whose total spin value falls into certain intervals     from output.lis
5  Number of trajectories whose dipole moment falls into certain intervals        from output.lis
6  Number of trajectories whose oscillator strength falls into certain intervals  from output_data/fosc.out

It can also sum the quantum amplitudes:
7  Quantum amplitudes in diagonal picture                                  from output_data/coeff_diag.out
8  Quantum amplitudes in MCH picture                                       from output_data/coeff_MCH.out
9  Quantum amplitudes in MCH picture (multiplets summed up)                from output_data/coeff_MCH.out

It can also transform the classical diagonal populations to MCH basis:
12  Transform diagonal classical populations to MCH
                                                                    from output_data/coeff_class_MCH.out
13  Transform diagonal classical populations to MCH (multiplets summed up)
                                                                    from output_data/coeff_class_MCH.out
14  Wigner-transform classical diagonal populations to MCH
                                                                    from output_data/coeff_mixed_MCH.out
15  Wigner-transform classical diagonal populations to MCH (multiplets summed up)
                                                                    from output_data/coeff_mixed_MCH.out

It can also compute diabatic populations:
20 Quantum amplitudes in diabatic picture                           from output_data/coeff_diab.out
21 Transform diagonal classical populations to diabatic             from output_data/coeff_class_diab.out
22 Wigner-transform classical diagonal populations to diabatic      from output_data/coeff_mixed_diab.out
Analyze mode: 9

Run data_extractor.x for each trajectory prior to performing the analysis?
For many or long trajectories, this might take some time.
Run data_extractor.x? [True] <ENTER>
Run data_extractor.x only if output.dat newer than output_data/ [True] <ENTER>

⋮          ⋮          ⋮          ⋮          ⋮          ⋮          ⋮

------------------Setup for bootstrapping?------------------

The population data can be analyzed by fitting with a kinetic model (via make_fitscript.py).
In order to estimate errors for these time constants (via bootstrapping),
additional data needs to be saved here.
Save data for bootstrapping? [False] <ENTER>

-----------------------Gnuplot script-----------------------

Gnuplot script? [False] yes
Gnuplot script filename? [populations.gp] (autocomplete enabled) pop_quant.gp
```

```
⋮          ⋮          ⋮          ⋮          ⋮          ⋮          ⋮

Overwrite pop.out?  [False] <ENTER>

Please enter the output filename:  (autocomplete enabled) pop_quant.out
Writing to pop_quant.out ...
```

Third, we obtain the number of trajectories whose oscillator strength falls into one of these intervals: $0 < f_{\mathrm{osc}} < 10^{-4}$, $10^{-4} < f_{\mathrm{osc}} < 1^{-1}$ and $10^{-1} < f_{\mathrm{osc}}$. Rerun **populations.py** again.

```
user@host> $SHARC/populations.py
```

```
⋮          ⋮          ⋮          ⋮          ⋮          ⋮          ⋮

------------------------Analyze Mode------------------------

This script can analyze the classical populations in different ways:
This script can analyze the classical populations in different ways:
1  Number of trajectories in each diagonal state                                    from output.lis
2  Number of trajectories in each (approximate) MCH state                           from output.lis
3  Number of trajectories in each (approximate) MCH state (multiplets summed up)    from output.lis
4  Number of trajectories whose total spin value falls into certain intervals       from output.lis
5  Number of trajectories whose dipole moment falls into certain intervals          from output.lis
6  Number of trajectories whose oscillator strength falls into certain intervals    from output_data/fosc.out

It can also sum the quantum amplitudes:
7  Quantum amplitudes in diagonal picture                                      from output_data/coeff_diag.out
8  Quantum amplitudes in MCH picture                                           from output_data/coeff_MCH.out
9  Quantum amplitudes in MCH picture (multiplets summed up)                    from output_data/coeff_MCH.out

It can also transform the classical diagonal populations to MCH basis:
12  Transform diagonal classical populations to MCH
                                                                               from output_data/coeff_class_MCH.out
13  Transform diagonal classical populations to MCH (multiplets summed up)
                                                                               from output_data/coeff_class_MCH.out
14  Wigner-transform classical diagonal populations to MCH
                                                                               from output_data/coeff_mixed_MCH.out
15  Wigner-transform classical diagonal populations to MCH (multiplets summed up)
                                                                               from output_data/coeff_mixed_MCH.out

It can also compute diabatic populations:
20 Quantum amplitudes in diabatic picture                                    from output_data/coeff_diab.out
21 Transform diagonal classical populations to diabatic                      from output_data/coeff_class_diab.out
22 Wigner-transform classical diagonal populations to diabatic               from output_data/coeff_mixed_diab.out
Analyze mode: 6

Run data_extractor.x for each trajectory prior to performing the analysis?
For many or long trajectories, this might take some time.
Run data_extractor.x? [True] no          # Was already done above

⋮          ⋮          ⋮          ⋮          ⋮          ⋮          ⋮

-------------------------Intervals-------------------------

Please enter the interval limits, all on one line.
```

```
Interval limits:  1e-4  1e-1          # Outer limits 0 and infinity are automatically assumed

⋮           ⋮           ⋮           ⋮           ⋮           ⋮           ⋮

------------------Setup for bootstrapping?-----------------

The population data can be analyzed by fitting with a kinetic model (via make_fitscript.py).
In order to estimate errors for these time constants (via bootstrapping),
additional data needs to be saved here.
Save data for bootstrapping? [False] <ENTER>

----------------------Gnuplot script----------------------

Gnuplot script? [False] yes
Gnuplot script filename? [populations.gp] (autocomplete enabled) pop_fosc.gp

⋮           ⋮           ⋮           ⋮           ⋮           ⋮           ⋮

Overwrite pop.out?  [False] <ENTER>

Please enter the output filename:  (autocomplete enabled) pop_fosc.out
Writing to pop_fosc.out ...
```

Use the produced GNUPLOT scripts to plot the obtained populations.

```
user@host> gnuplot pop_class.gp
user@host> gnuplot pop_quant.gp
user@host> gnuplot pop_fosc.gp
```

This will create the files **pop_class.gp.png**, **pop_quant.gp.png** and **pop_fosc.gp.png**. They are shown in figures 2.9, 2.10 and 2.11. In 2.9, the classical populations are given. In figure 2.10, the incoherent sum of the quantum amplitudes is given (obtained by using mode 9 in **populations.py**). In figure 2.11, the 5 trajectories are classified depending on their oscillator strengths.

Discussion of Figure 2.9 In figure 2.9 it can be seen that between 0 and 30 fs, all trajectories changed from the initial S_2 state through the S_1 state to the S_0 ground state. The triplet states remain completely unpopulated.

Discussion of Figure 2.10 In figure 2.10 the quantum populations are shown. For sufficiently large ensembles, figure 2.9 should closely follow figure 2.10. Consistency between the classical and quantum populations can be improved by using the decoherence correction (input option in **setup_traj.py**).

Discussion of Figure 2.11 In figure 2.11 the ensemble population was classified according to the oscillator strength of the classically populated state. The chosen interval limits were 0.0001 and 0.1, giving three classes of states (below 0.0001, between 0.0001 and 0.1, and above 0.1). Initially, all trajectories are in the bright $\pi\pi^*$ state and are thus classified into the third class. During the dynamics, the dissociations/torsions/hops reduce the oscillator strength, so that the trajectories are classified into the intermediate class. Later, the trajectories decay to the ground state, which by definition has an oscillator strength of zero (since f_{osc} is proportional to the excitation energy). Note that in this example the ground state cannot be distinguished from the triplet state, which has negligible oscillator strength and thus would also be classified into the first class. In general, however, classifying the population according to oscillator strength sometimes allows to approximately obtain populations of $\pi\pi^*$ and $n\pi^*$ states.

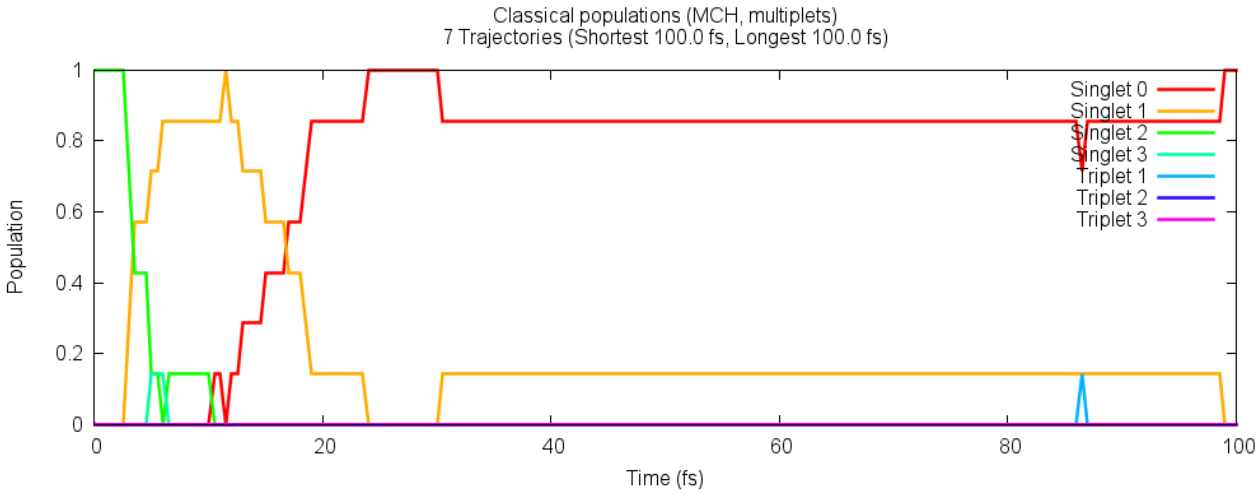

Figure 2.9: Classical populations for an ensemble of 7 trajectories.

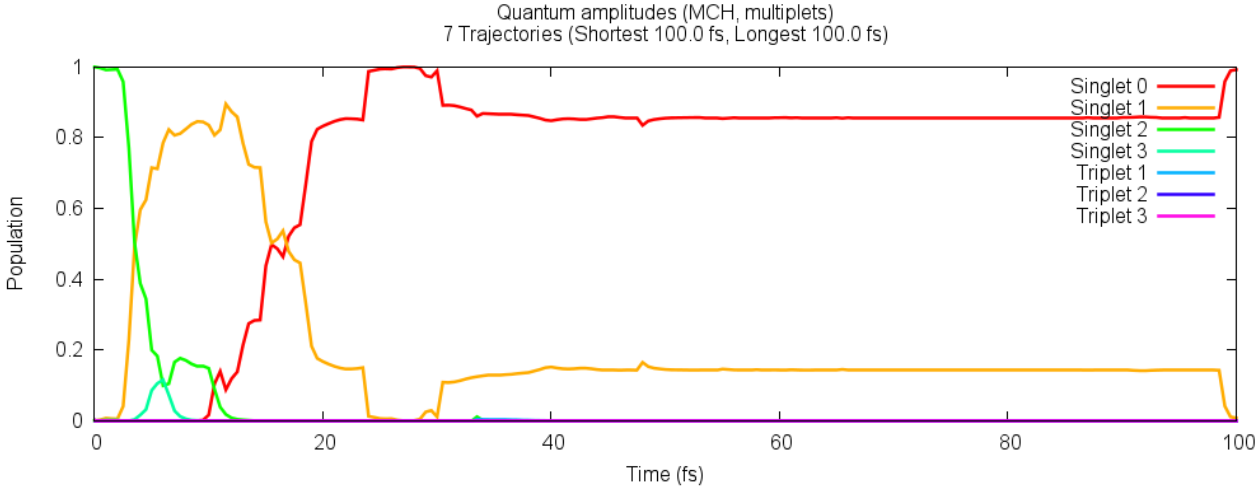

Figure 2.10: Quantum populations for an ensemble of 7 trajectories.

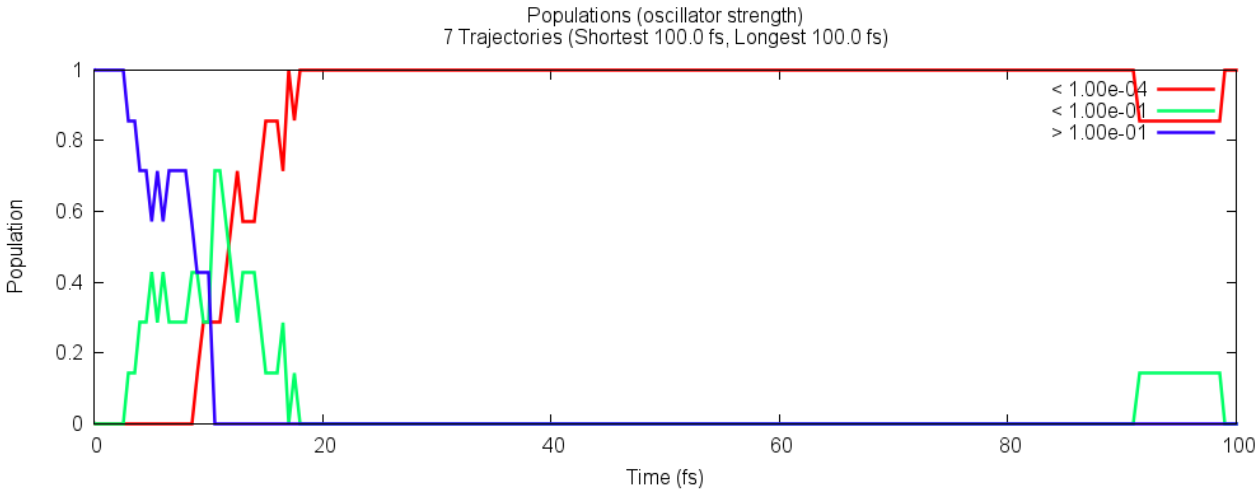

Figure 2.11: Populations classified based on oscillator strength for an ensemble of 7 trajectories.

2.9.3 Ensemble Populations Flow

In figure 2.9 it can be seen that, generally, population flows from the S_2 to the S_1 to the S_0. In order to quantify this population flow, one can use **`transition.py`**. This script counts the number of hops in all trajectories.

See Section 7.32 (p. 221) in the manual.

```
user@host> $SHARC/transition.py
```

```
Script for hop counting started...


  ================================================================================
||                                                                              ||
||                  Counting hopping events from SHARC dynamics                 ||
||                                                                              ||
||                             Author: Sebastian Mai                            ||
||                                                                              ||
||                                  Version:4.0                                 ||
||                                    01.09.24                                  ||
||                                                                              ||
  ================================================================================


This script reads output.lis files files and counts all hopping events
to produce a matrix with the transition counts.

--------------------Paths to trajectories-------------------

Please enter the paths to all directories containing the "TRAJ_0XXXX" directories.
E.g. S_2 and S_3.
Please enter one path at a time, and type "end" to finish the list.
Path:  [end] (autocomplete enabled) Singlet_2/
['TRAJ_00005', 'TRAJ_00004', 'TRAJ_00008', 'plot.gp', 'TRAJ_00009', 'TRAJ_00002',
'TRAJ_00006', 'TRAJ_00010', 'TRAJ_00007', 'TRAJ_00003']
Found 9 subdirectories in total.

Path:  [end] (autocomplete enabled) <ENTER>

Total number of subdirectories: 9

-----------------------Analyze Mode-----------------------

This script finds the transition matrix:
1        In MCH basis                                                  from output.lis
2        In MCH basis (ignoring hops within one multiplet)             from output.lis

This script can also print the transition matrix for each timestep:
3        In MCH basis                                                  from output.lis
4        In MCH basis (ignoring hops within one multiplet)             from output.lis
5        In MCH basis [cumulative]                                     from output.lis
6        In MCH basis [cumulative] (ignoring hops within one multiplet)  from output.lis

Analyze mode: 2

---------------------Number of states---------------------

Please enter the number of states as a list of integers
e.g. 3 0 3 for three singlets, zero doublets and three triplets.
Number of states: [4 0 3] <ENTER>
```

```
----------------------Simulation time----------------------

Up to which simulation time should the analysis be performed?
Simulation time (in fs):  [1000.0] 100


########################Full input#########################

paths                     ['Singlet_2/']
run_extractor             False
states                    [4, 0, 3]
mode                      2
maxtime                   100.0
nstates                   7
nmstates                  13

Do you want to do the specified analysis? [True] <ENTER>

Checking the directories...
Singlet_2//TRAJ_00002         OK
Singlet_2//TRAJ_00003         OK
Singlet_2//TRAJ_00004         OK
Singlet_2//TRAJ_00005         OK
Singlet_2//TRAJ_00006         OK
Singlet_2//TRAJ_00007         OK
Singlet_2//TRAJ_00008         OK
Singlet_2//TRAJ_00009         DETECTED FILE dont_analyze
Singlet_2//TRAJ_00010         DETECTED FILE dont_analyze
Number of trajectories: 7
Number of steps: 201


**************************Results**************************
Full transition matrix:
        |      S0      S1      S2      S3      T1      T2      T3
--------+-----------------------------------------------------------
  S0    |    1024       9       0       0       1       0       0
  S1    |       2     294       7       0       0       0       0
  S2    |       0       0      58       1       0       0       0
  S3    |       0       0       1       2       0       0       0
  T1    |       1       0       0       0       0       0       0
  T2    |       0       0       0       0       0       0       0
  T3    |       0       0       0       0       0       0       0

Sum transition matrix:
        |      S0      S1      S2      S3      T1      T2      T3
--------+-----------------------------------------------------------
  S0    |    1024      11       0       0       2       0       0
  S1    |       0     294       7       0       0       0       0
  S2    |       0       0      58       2       0       0       0
  S3    |       0       0       0       2       0       0       0
  T1    |       0       0       0       0       0       0       0
  T2    |       0       0       0       0       0       0       0
  T3    |       0       0       0       0       0       0       0

Difference transition matrix:
        |      S0      S1      S2      S3      T1      T2      T3     Sum
--------+-------------------------------------------------------------------
  S0    |       0       7       0       0       0       0       0       7
  S1    |      -7       0       7       0       0       0       0       0
```

```
S2    |      0      -7       0       0       0       0       0      -7
S3    |      0       0       0       0       0       0       0       0
T1    |      0       0       0       0       0       0       0       0
T2    |      0       0       0       0       0       0       0       0
T3    |      0       0       0       0       0       0       0       0
Sum   |     -7       0       7       0       0       0       0       0
```

The most important matrix in the output is the difference transition matrix, which shows the "net" hops. In our example, it shows that there were 7 net hops from the S_2 to the S_1 and 7 net hops from the S_1 to the S_0. No net hops occurred directly between the S_2 and S_0, or involving the triplet states. Hence, the population flow in the ensemble is clearly $S_2 \rightarrow S_1 \rightarrow S_0$.

2.9.4 Fitting Ensemble Populations including Error Estimation

The results in figure 2.9 show that internal conversion is an ultrafast process in $CH_2NH_2^+$. However, in order to facilitate comparison to experiments it is useful to obtain a time constant for the relaxation processes. SHARC allows fitting population data to combinations of unimolecular reactions, using **make_fit.py**. Here, we will fit the population data to the kinetic model $S_2 \rightarrow S_1 \rightarrow S_0$, which was the result of the population flow analysis.

See Section 7.33 (p. 221) in the manual.

Besides simply performing the fit, we will also obtain error estimates of the fitted time constants with the same script using the bootstrapping method. In this method, based on the original ensemble (of 7 trajectories), "resample" ensembles are generated by randomly drawing *with replacement* trajectories from the original ensemble (here, drawing 7 random trajectories). Each resample is then fitted in exactly the same way as the original ensemble, and after many resamples a distribution of possible fitting parameters is obtained. From this distribution, one can then find error measures for the fitting parameters.

See Section 8.10 (p. 247) in the manual.

In order to start the script, run:

```
user@host> $SHARC/make_fit.py
```

See Section 8.4 (p. 243) in the manual.

```
  ================================================================================
||                                                                              ||
||                     Direct fitting for SHARC populations                     ||
||                                                                              ||
||                             Author: Sebastian Mai                            ||
||                                                                              ||
||                                  Version:4.0                                 ||
||                                    01.09.24                                  ||
||                                                                              ||
  ================================================================================


This script fits SHARC populations (as generated with populations.py)
to general kinetic models based on first-order population transfer.


############################################################
######################Kinetics Model########################
############################################################


-----------------------Model Species----------------------

First, please specify the set of species used in your model kinetics.

Possible input:
+ <label> <label> ...   Adds one or several species to the set
- <label> <label> ...   Removes one or several species from the set
show                    Show the currently defined set of species
end                     Finish species input

Each label must be unique. Enter the labels without quotes.

Input: [end] + S0 S1 S23     # multiple labels can be added
  Species 'S0' added!
  Species 'S1' added!
  Species 'S23' added!                              # sum of S2 and S3 for simplicity
Input: [end] <ENTER>

Final species set:  ['S0', 'S1', 'S23']
```

```
----------------Model Elementary Reactions----------------

Second, please specify the set of elementary reactions in your model kinetics.

Possible input:
+ <species1> <species2> <rate_label>  Add a reaction from species1 to species2 with labelled rate constant
- <rate_label>                        Remove the reaction with the given rate constant
show                                  Show the currently defined set of reactions (as adjacency matrix)
end                                   Finish reaction input

Each rate label must be unique.

Input: [end] + S23 S1 k21    # one reaction at a time
  Reaction from 'S23' to 'S1' with rate label 'k21' added!
Input: [end] + S1 S0 k10     # one reaction at a time
  Reaction from 'S1' to 'S0' with rate label 'k10' added!
Input: [end] <ENTER>

Final reaction network:
     |  S0   S1  S23
-----+---------------
  S0 |   .    .    .
  S1 | k10    .    .
 S23 |   .  k21    .

(Initial species: rows; Final species: columns)

-----------------Model Initial Conditions-----------------

Third, please specify species with non-zero initial populations.

Possible input:
+ <species>       Declare species to have non-zero initial population
- <species>       Remove species from the set of non-zero initial populations
show              Show the currently defined non-zero initial populations
end               Finish initial condition input

Input: [end] + S23         # initial population is in S2
  Species 'S23' added!
Input: [end] <ENTER>

Final initial species set:  ['S23']

############################################################
###################### Fitting Data ########################
############################################################


----------------------Operation mode----------------------

This script can work with the following output:
* pop.out (file from populations.py)
* bootstrap_data/ (directory from populations.py)
Using only the pop.out allows fitting and obtaining time constants.
Using the bootstrap data instead additionally allows for realistic error estimates.

Do you want to use bootstrap data? [False] yes
```

```
How many bootstrap samples? [100] <ENTER>


--------------------Population data file--------------------

Please specify the path to the bootstrap data directory (as generated by populations.py).

Bootstrap data directory: [bootstrap_data/] (autocomplete enabled) <ENTER>
  Detected maximal time of   100.0 fs and 8 columns (time plus 7 data columns).

Do you want to write fitting curves for all bootstrap cycles? [False] <ENTER>

------------Population-to-Species Mapping for Fit-----------

Please specify which model species should be fitted to which data file columns.
For example, you can fit the label 'S0' to column 2:
  S0 = 2
You can also fit the sum of two species to a column:
  T1 T2 = 5
You can also fit a species to the sum of several columns:
  T_all = 5 6 7
You can even fit a sum of species to a sum of columns:
  T1 T2 = 5 6 7

On the right side, "~" can be used to indicate ranges:
  T1 T2 = 5~9

Possible input:
<species1> <species2> ... = <columns1> <column2> ...            Set one mapping
show                                                            Show mapping
end                                                             Finish mapping input
reset                                                           Redo the mapping input

Each species label must be used at most once.
Each column number (except for '1', which denotes the time) must be used at most once.

Set of species:        ['S0', 'S1', 'S23']
Set of column numbers: [2, 3, 4, 5, 6, 7, 8]
Input: [end] S0 = 2       # fit S0 to column 2 data
Input: [end] S1 = 3       # fit S1 to column 3 data
Input: [end] S23 = 4 5    # fit S23 to sum of column 4 and 5
Input: [end] <ENTER>
Final mappings:
     S0  =  2
     S1  =  3
     S23  =  4  5


############################################################
#################### Fitting procedure #####################
############################################################


-------------Initial guesses------------

Please check the initial guesses for the parameters

Possible input:
label = value     Set an initial guess (detects type automatically and computes k=1/t for rates)
show              Show the currently defined non-zero initial populations
```

```
end               Finish initial condition input


  time constant ( k10          ):     100.0000 fs
  time constant ( k21          ):     120.0000 fs
  initial pop   ( S23          ):       1.0000

Input: [end] <ENTER>
Final guess parameters:
  time constant ( k10          ):     100.0000 fs
  time constant ( k21          ):     120.0000 fs
  initial pop   ( S23          ):       1.0000



------Optimize initial populations------

Do you want to optimize the initial populations (otherwise only the rates)? [True] no


--------Constrained optimization--------

Do you want to restrict all rates/initial populations to be non-negative? [True] <ENTER>

########################Full input#########################

bootstrap_cycles          100
bounds                    True
columns_groups            [[2], [3], [4, 5]]
data                      [ ... ]
do_bootstrap              True
initial                   [2]
initset                   set(['S23'])
maxtime                   100.0
ncol                      8
ngroups                   3
ninitial                  1
nrates                    2
nspec                     3
ntraj                     7
opt_init                  False
p0                        [0.01, 0.008333333333333333]
popfile                   /user/mai/NewSHARC/SHARC_2.0/TUTORIAL/2_full/Tutorial/traj/bootstrap_data
rank                      0
rate_matrix               [['', '', ''], ['k10', '', ''], ['', 'k21', '']]
ratemap                   0: 'k10', 1: 'k21', 'k10': 0, 'k21': 1
rates                     [[(1, 0)], [(2, 1)]]
rateset                   set(['k10', 'k21'])
species_groups            [['S0'], ['S1'], ['S23']]
specmap                   0: 'S0', 1: 'S1', 2: 'S23', 'S1': 1, 'S0': 0, 'S23': 2
summation                 [[0], [1], [2]]
write_bootstrap_fits      False
y0                        [1.0]

Do you want to continue? [True] <ENTER>


######################### Fitting ########################
```

```
------------- Iterations -------------

   Iteration     Total nfev        Cost      Cost reduction    Step norm     Optimality
       0              1         1.0020e+02                                  4.40e+03
       1              2         6.0222e+01      4.00e+01       1.30e-02       1.96e+03
       2              3         2.6394e+01      3.38e+01       2.60e-02       6.83e+02
       3              4         1.2458e+01      1.39e+01       3.27e-02       1.96e+02
       4              5         6.8237e+00      5.63e+00       5.06e-02       5.85e+01
       5              6         4.8320e+00      1.99e+00       8.03e-02       2.70e+01
       ⋮              ⋮                ⋮              ⋮              ⋮              ⋮
      17             18         4.6205e+00      1.73e-08       3.43e-05       1.46e-03
‘ftol‘ termination condition is satisfied.
Function evaluations 18, initial cost 1.0020e+02, final cost 4.6205e+00, first-order optimality 1.46e-03.


----------- Final parameters -----------

time constant ( k10          ):      16.4593 fs +/-       0.5543 fs (   3.37 %)
time constant ( k21          ):       4.1820 fs +/-       0.2735 fs (   6.54 %)
initial pop   ( S23          ):       1.0000

These time constants include errors that assume that the population data
is free of uncertainty. Bootstrapping analysis is following now.

Raw data and fitted functions written to "fit_results.txt".

GNUPLOT script written to "fit_results.gp".

###################### Bootstrapping ######################

 Cycle          k10          k21          S23  Time
     1      33.1210       4.2232       1.0000  0:00:00.573773
     2      16.7930       3.7431       1.0000  0:00:00.681325
     3      10.7393       5.2129       1.0000  0:00:00.600873
     4      27.8368       3.4376       1.0000  0:00:00.804838
     5      11.7184       4.6316       1.0000  0:00:00.920739
     ⋮            ⋮            ⋮            ⋮               ⋮
   100      49.7747       3.8927       1.0000  0:00:00.634471

>>>>>>>>>>>>> Finished the bootstrapping cycles ...









---------------------------------- Analysis for time constant "k10" ----------------------------------

    Arithmetic analysis:               19.6161 +/-      10.9649
                                            ( +/-      55.90 %)

    Geometric analysis:                17.4018  +       10.3242  -        6.4798
                                            (  +       59.33 %  -       37.24 %)
```

```
    Minimum and maximum:                9.2170      and            54.7688

    Histogram:
    ==========
                            #                                    | 28
                            #     #                              | 25
                            #     #                              | 23
                            #     #                              | 21
                            #     #                              | 18
                            #     #                              | 16
                            #     #                              | 14
                            #     #     #                        | 11
                            #     #     #           #            | 9
                            #     #     #     #     #            | 7
                      #     #     #     #     #     #            | 4
                      #     #     #     #     #     #     #      | 2
        |     |     |     |     |     |     |     |     |     |
    4.302 5.690 7.524 9.950 13.16 17.40 23.01 30.43 40.25 53.22

---------------------------------- Analysis for time constant "k21" ----------------------------------

    Arithmetic analysis:                4.2623 +/-       0.9002
                                            ( +/-      21.12 %)

    Geometric analysis:                 4.1687  +        0.9879  -        0.7987
                                            (  +       23.70 %  -       19.16 %)

    Minimum and maximum:                2.6746      and             6.1591

    Histogram:
    ==========
                                         #                       | 24
                                         #                       | 22
                                         #                       | 20
                                   #     #                       | 18
                            #      #     #                       | 16
                            #      #     #     #                 | 14
                            #      #     #     #     #           | 12
                            #      #     #     #     #           | 10
                      #     #      #     #     #     #           | 8
                      #     #      #     #     #     #           | 6
                #     #     #      #     #     #     #           | 4
                #     #     #      #     #     #     #           | 2
        |     |     |     |     |     |     |     |     |     |
    2.202 2.502 2.843 3.230 3.669 4.169 4.736 5.381 6.113 6.945




--------------------------------- Analysis for initial population "S23" ---------------------------------

    Arithmetic analysis:                1.0000 +/-       0.0000
                                            ( +/-       0.00 %)

    Geometric analysis:                 1.0000  +        0.0000  -       -0.0000
                                            (  +        0.00 %  -       -0.00 %)
```

```
    Minimum and maximum:                1.0000      and             1.0000

    Histogram:
    ==========
     #                                                          | 100
     #                                                          | 91
     #                                                          | 83
     #                                                          | 75
     #                                                          | 66
     #                                                          | 58
     #                                                          | 50
     #                                                          | 41
     #                                                          | 33
     #                                                          | 25
     #                                                          | 16
     #                                                          | 8
        |     |     |     |     |     |     |     |     |     |
    1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000


----------- Final parameters -----------

time constant ( k10          ):      16.4593 fs +/-      10.9649 fs (  66.62 %)
time constant ( k21          ):       4.1820 fs +/-       0.9002 fs (  21.52 %)
initial pop   ( S23          ):       1.0000


Output (analysis and full fitted data) written to "fit_bootstrap.txt".
```

Unlike the fitting scripts used in the previous SHARC release, **make_fit.py** directly carries out the kinetic model fit in one program, without calling **maxima** or **gnuplot**. It is therefore much faster, and additionally allows fitting of models that are too complex to solve analytically with **maxima**.

The main results of the script can be found under **Final parameters**, obtained after 17 iterations. The table presents the fitted values of all parameters. It can be seen that the $S_2 \rightarrow S_1$ time constant is 4.2 fs, and the time constant for $S_1 \rightarrow S_0$ is 16.5 fs.

For visual inspection of the fit results, the script writes two new files, **fit_results.txt** and **fit_results.gp**. The former is a simple text file that contains three columns. The first column is the time axis of the global fit, i.e., the time axis of the data, but continued to accommodate all data sets to be fitted (in the present case, there are three data sets to be fitted as defined in the Population-to-species mapping). The second column is the data and the third column is the fitted kinetic model functions. This file can be conveniently plotted with GNUPLOT:

```
user@host> gnuplot fit_results.gp
```

The result of this plot is shown in Figure 2.12.

Discussion of Figure 2.12 From the figure, it can be seen that the $S_2 \rightarrow S_1$ time constant is 4.2 fs, and the time constant for $S_1 \rightarrow S_0$ is 16.5 fs. Note that the fit is not very good, because the number of trajectories is low and the relaxation processes in $CH_2NH_2^+$ are so fast that first-order reactions are not optimal to describe them. For slower processes in larger molecules, these kinetic model fits might work better.

Discussion of the bootstrapping results Because we provided bootstrapping data to the script, after the main data fit the script automatically continues the run and performs the requested number of bootstrap cycles. In each cycle, it will write the parameters fitted for the current cycle. Note that if the bootstrapping

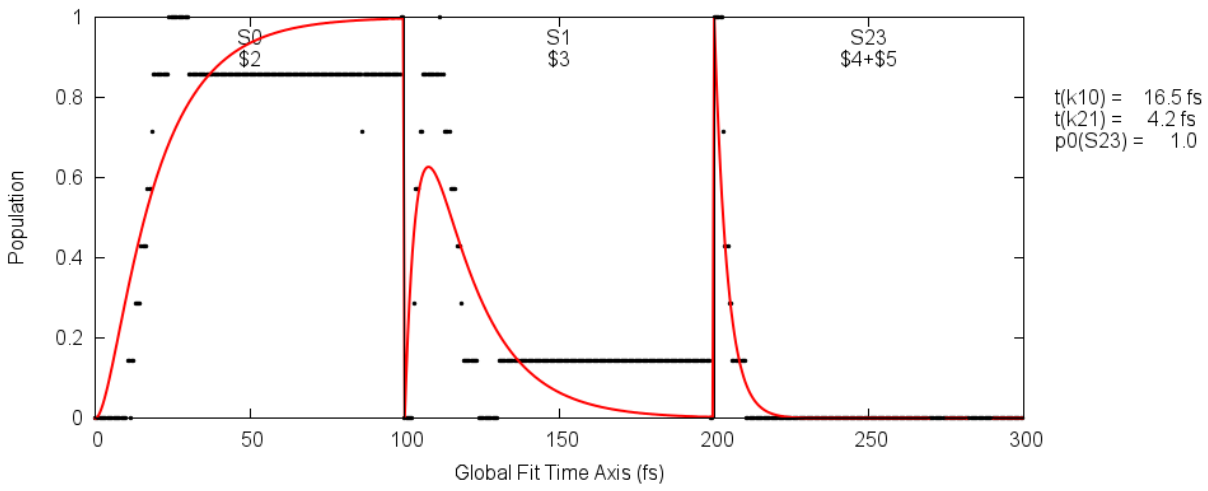

Figure 2.12: Kinetic model fit of the classical populations.

iterations take too long and the results seem to be converged already, pressing **`Ctrl+C`** allows skipping the remaining iterations and directly leads to the final analysis.

The result of the bootstrapping procedure is presented in a summary for each fitting parameter. Note that by default also the initial populations are treated as fitting parameters, even if they are fixed in the shown example.

The results show that the $S_2 \rightarrow S_1$ relaxation process in the trajectories had a time constant of 4.2±0.9 fs (using the arithmetic analysis). Most fits yielded a time constant between 3 and 6 fs.

For the $S_1 \rightarrow S_0$ process, the result is 21.3±13.6 fs. According to the histogram, most of the fits yielded a value between 13 and 20 fs, although there were many fits with values around 30 fs and even some above 50 fs. This broad distribution leads to the large error found. Clearly, it is due to the small number of trajectories—it is very likely that in some resamples one particular trajectory is missing, which leads to a large change of the fitted parameters. Also note that the distribution of this parameter is very skewed (with a long right tail), leading to a strong deviation of the fitted value from the main data (16.5 fs) and the average of the bootstrap results (21.3 fs). In this situation, one might want to combine the main data value with the error from bootstrapping (16.5±13.6 fs) or use the results of the geometric analysis (which gives $18.4^{+11.8}_{-7.2}$ fs).

Generally, the errors get smaller as more trajectories are employed, and hence, the fitting errors are a good tool to judge whether enough trajectories were computed. For some time constants, it might also be necessary to run the trajectories for longer time to reduce the errors. In any case, the obtained errors tell nothing about the inherent method error—using surface hopping in combination with a given quantum chemistry method. It is not possible to quantify this method error with **`make_fit.py`**; only through comparison with reference data or experiment can the method error be judged.

2.9.5 Hopping Geometries

Another aspect one might be interested in are certain critical geometries from the trajectories. **crossing.py** is a script that collects those geometries from all trajectories where a surface hop between two specified states occurred. Its usage is comparable to **populations.py**.

See Section 7.34 (p. 224) in the manual.

```
user@host> $SHARC/crossing.py
```

```
Script for hopping geometry extraction started...


  ================================================================================
||                                                                              ||
||                 Reading hopping geometries from SHARC dynamics               ||
||                                                                              ||
||                             Author: Sebastian Mai                            ||
||                                                                              ||
||                                  Version:4.0                                 ||
||                                    01.09.24                                  ||
||                                                                              ||
  ================================================================================


This script reads output.lis files and output.xyz files to produce a list of
all geometries where certain surface hops (or other events) occured.

--------------------Paths to trajectories-------------------

Please enter the paths to all directories containing the "TRAJ_0XXXX" directories.
E.g. S_2 and S_3.
Please enter one path at a time, and type "end" to finish the list.
Path:  [end] (autocomplete enabled) Singlet_2/
['TRAJ_00005', 'TRAJ_00004', 'TRAJ_00008', 'plot.gp', 'TRAJ_00009', 'TRAJ_00002',
'TRAJ_00006', 'TRAJ_00010', 'TRAJ_00007', 'TRAJ_00003']
Found 9 subdirectories in total.

Path:  [end] (autocomplete enabled) <ENTER>

Total number of subdirectories: 9

-----------------------Analyze Mode-----------------------

This script can find geometries where:
1        A change of MCH state occured (ignoring hops within one multiplet)              from output.lis

Analyze mode: 1

---------------------Number of states---------------------

Please enter the number of states as a list of integers
e.g. 3 0 3 for three singlets, zero doublets and three triplets.
Number of states: [4 0 3] <ENTER>


--------------States involved in surface hop--------------

In this analysis mode, all geometries are fetched where a trajectory switches from a given MCH state
to another given MCH state.
```

```
Please enter the old MCH state involved as "mult state", e.g., "1 1" for S0, "1 2" for S1, or "3 1" for T1:
State 1: 1  2      # Only hops from S1

Please enter the new MCH state involved (mult state):
State 2: 1  1      # to S0

Direction:
1       Forwards      # Only S1 -> S0
2       Backwards     # Only S0 -> S1
3       Two-way       # Both S1 -> S0 and S0 -> S1

Direction mode: [3] 1

########################Full input#########################

paths                       ['Singlet_2/']
tostates                    [[1, 1], [1, 2]]
run_extractor               False
states                      [4, 0, 3]
mode                        1
dirmode                     1
nstates                     7
fromstates                  [[1, 1], [1, 2]]
nmstates                    13

Do you want to do the specified analysis? [True] <ENTER>

Checking the directories...
Singlet_2//TRAJ_00002         OK
Singlet_2//TRAJ_00003         OK
Singlet_2//TRAJ_00004         OK
Singlet_2//TRAJ_00005         OK
Singlet_2//TRAJ_00006         OK
Singlet_2//TRAJ_00007         OK
Singlet_2//TRAJ_00008         OK
Singlet_2//TRAJ_00009         DETECTED FILE dont_analyze
Singlet_2//TRAJ_00010         DETECTED FILE dont_analyze
Number of trajectories: 7

Writing to crossing.xyz ...
```

The script writes a files called **`crossing.xyz`**, which contains all geometries (9 geometries in this example) where a hop from the S_1 to the S_0 occurred. This file can in turn be analyzed with **`geo.py`** in order to calculate internal coordinates (e.g., to find whether a bond or angle controls access to the S_1/S_0 crossing seam, or how many different pathways allow this transition).

2.9.6 Optimizing from Hopping Geometries

The output of **crossing.py** can also be used to optimize minimum-energy crossing points and conical intersections (e.g., to find how many independent crossing seam regions, i.e., reaction pathways, are involved). In SHARC, this optimization can be automatically setup if ORCA is installed (ORCA is needed to perform the actual optimization of the nuclei, whereas SHARC provides the necessary energies and gradients).

See Section 7.43 (p. 237) in the manual.

The setup of the optimizations can be started with

```
user@host> $SHARC/setup_orca_opt.py
```

See Section 8.20 (p. 259) in the manual.

```
Script for setup of optimizations with ORCA and SHARC started...


  ================================================================================
||                                                                              ||
||                    Setup optimizations with ORCA and SHARC                   ||
||                                                                              ||
||                      Author: Moritz Heindl, Sebastian Mai                    ||
||                                                                              ||
||                                  Version:4.0                                 ||
||                                    24.04.24                                  ||
||                                                                              ||
  ================================================================================

This script automatizes the setup of the input files ORCA+SHARC optimizations.

-----------------------Path to ORCA-----------------------

Please specify path to ORCA directory (SHELL variables and ~ can be used,
will be expanded when interface is started).

Path to ORCA: [$ORCADIR/] (autocomplete enabled) <ENTER>

----------Choose the quantum chemistry interface----------

Please specify the quantum chemistry interface (enter any of the following numbers):
  1 SHARC_ADAPTIVE          HYBRID interface for adaptive sampling
  2 SHARC_AMS_ADF                  (Not Available! Use SHARC_LEGACY to work with this interface)
  3 SHARC_ANALYTICAL          FAST interface for analytical model Hamiltonians with sympy
  4 SHARC_ASE_DB            HYBRID interface for saveing data to ASE db
  5 SHARC_BAGEL                    (Not Available! Use SHARC_LEGACY to work with this interface)
  6 SHARC_COLUMBUS                 (Not Available! Use SHARC_LEGACY to work with this interface)
  7 SHARC_DO_NOTHING          FAST interface for testing (zero E/grad/NAC/DM, unity overlaps/phases)
  8 SHARC_ECI               HYBRID interface for excitonic HF/CI with multiple fragments
  9 SHARC_FALLBACK          HYBRID interface for calling a fallback interface if primary interface fails
 10 SHARC_GAUSSIAN       AB INITIO interface for GAUSSIAN16 for single-reference methods (CIS, TDDFT)
 11 SHARC_LEGACY             BASIC interface for running legacy interfaces via file I/O (AMS-ADF, BAGEL, COLUMBUS, MOLPRO,
 12 SHARC_LVC                 FAST interface for linear/quadratic vibronic coupling models (LVC, QVC, LVC/MM)
 13 SHARC_MNDO           AB INITIO interface for the MNDO program (OM2-MRCI)
 14 SHARC_MOLCAS         AB INITIO interface for OpenMolcas (>v23) for multireference calculations (CASSCF/RASSCF, MS/XMS-
 15 SHARC_MOLPRO                  (Not Available! Use SHARC_LEGACY to work with this interface)
 16 SHARC_MOPACPI        AB INITIO interface for the MOPAC-PI program
 17 SHARC_NUMDIFF           HYBRID interface for numerical derivatives (grad, NACdr, SOCdr, DMdr)
 18 SHARC_NWCHEM         AB INITIO interface for NWChem (TDDFT)
 19 SHARC_OPENMM                  (Not Available!)
 20 SHARC_ORCA           AB INITIO interface for ORCA v5-6 (CIS/TDDFT)
 21 SHARC_PYSCF                   (Not Available! Use SHARC_LEGACY to work with this interface)
 22 SHARC_QMMM              HYBRID interface for QM/MM (electrostatic embedding, link atom scheme)
```

```
 23 SHARC_QMOUT               FAST interface for frozen-nuclei dynamics (constant E/SOC/DM, unit overlaps, zero grad/NAC)
 24 SHARC_SPAINN              FAST interface for SPaiNN
 25 SHARC_TINKER                   (Not Available!)
 26 SHARC_TURBOMOLE      AB INITIO interface for TURBOMOLE (RICC2/ADC2)
 27 SHARC_UMBRELLA          HYBRID interface for adding umbrella-sampling-style restraints (harmonic bonds, angles, dihedr

Interface number: 14

The following interface was selected:
 14 SHARC_MOLCAS         AB INITIO interface for OpenMolcas (>v23) for multireference calculations (CASSCF/RASSCF, MS/XMS-
-------------------------Geometry-------------------------

Please specify the geometry file (xyz format, Angstroms):
Geometry filename: [geom.xyz] (autocomplete enabled) crossing.xyz
Number of geometries: 9

---------------------Number of states----------------------


Please enter the number of states as a list of integers
e.g. 3 0 3 for three singlets, zero doublets and three triplets.
Number of states: 4  0  3


Please enter the molecular charge for each chosen multiplicity
e.g. 0 +1 0 for neutral singlets and triplets and cationic doublets.
Molecular charges per multiplicity: [0 1 0] 1 0 1
Number of states: [4, 0, 3]
Total number of states: 13

1: [1, 1, 0.0], 2: [1, 2, 0.0], 3: [1, 3, 0.0], 4: [1, 4, 0.0], 5: [3, 1, -1.0], 6: [3, 2, -1.0], 7: [3, 3, -1.0], 8: [3,

---------------------States to optimize---------------------

Do you want to optimize a minimum? (no=optimize crossing): [True] no

Please specify the first state involved in the optimization
e.g. 3 2 for the second triplet state.
State: [1 1] <ENTER>

Please specify the second state involved in the optimization
e.g. 3 2 for the second triplet state.
Root: [1 2] <ENTER>
Multiplicities of both states identical, optimizing a conical intersection.

Please enter the values for the maximum allowed displacement per timestep
(choose smaller value if starting from a good guess and for large sigma or small alpha).
Maximum allowed step:  [0.3] <ENTER>

================================================================================
||                                                                            ||
||                          MOLCAS interface setup                            ||
||                                                                            ||
================================================================================



Specify path to MOLCAS.
Path to MOLCAS: [$MOLCAS] (autocomplete enabled) <ENTER>
```

```
Specify a scratch directory. The scratch directory will be used to run the calculations.
Path to scratch directory: (autocomplete enabled) $TMPDIR
Specify a path to a MOLCAS template file.
Template path: (autocomplete enabled) ../MOLCAS.template
Specify the number of CPUs to be used.
Number of CPUs: [1] <ENTER>
Specify the amount of RAM to be used.
Memory (MB): [1000] 500
Initial wavefunction: MO Guess

Please specify the path to a MOLCAS JobIph file containing suitable starting MOs for the CASSCF calculation.
Please note that this script cannot check whether the wavefunction file and the Input template are consistent!
Do you have initial wavefunction files for multiplicities 1 3? [True] <ENTER>
JobIph files (1) or RasOrb files (2)? 2
Initial wavefunction file for multiplicity 1: [MOLCAS.1.RasOrb.init] (autocomplete enabled) ../MOLCAS.RasOrb
Initial wavefunction file for multiplicity 3: [MOLCAS.3.RasOrb.init] (autocomplete enabled) ../MOLCAS.RasOrb

  ================================================================================
||                                Run mode setup                                ||
  ================================================================================


-------------------------Run script------------------------


Where do you want to perform the calculations? Note that this script cannot check whether the path is valid.
Run directory? (autocomplete enabled) orca_opt



########################Full input#########################

orca                      $ORCADIR/
geom_location             crossing.xyz
ngeom                     11
natom                     6
states                    [4, 0, 3]
nstates                   13
charge                    [0, 1, 0]
statemap                  1: [1, 1, 0.0], 2: [1, 2, 0.0], 3: [1, 3, 0.0], 4: [1, 4, 0.0], 5: [3, 1, -1.0], 6: [3, 2, -1.0]
maxmult                   3
opttype                   cross
cas.root1                 1
cas.root2                 2
calc_ci                   True
use_nacs                  True
maxstep                   0.3
cwd                       /user/mai/Documents/NewSHARC/SHARC_4.0/TUTORIAL/2_full/Tutorial/traj
needed_requests           'nacdr', 'h'
here                      False
copydir                   orca_opt

Do you want to setup the specified calculations? [True]

  ================================================================================
||                            Setting up directory...                           ||
  ================================================================================
```

The script will create 9 subdirectories in the given setup directory, **orca_opt/**. Each of these 9 directories

will contain the input files for an optimization using ORCA's external optimizer feature, where the energies and gradients will be provided by **$SHARC/orca_External** (this is done automatically), which itself calls the MOLCAS interface to do the computations.

In order to run one of these optimizations, execute:

```
user@host> cd orca_opt/geom_9/
user@host> sh run_EXTORCA.sh&
```

The progress of the optimization can be followed in **orca.log**. In this file, note the lines following **EXTERNAL SHARC JOB**, which are written by SHARC and show the computed energies and gradients. Close to a box labeled **Geometry convergence**, ORCA reports the convergence status of the optimization. After 25 cycles, the optimization should converge; the energies of S_0 and S_1 at convergence should be −94.263728 and −94.263586 Hartree. This is a very good result, as the energy gap is only 0.004 eV; if the gap was much larger, then the optimization should be repeated (starting from the last step) with adjusted optimization parameters (**Sigma** and **Alpha**, see also the manual).

2.9.7 Essential Dynamics Analysis

Another possibility to investigate nuclear motion in the trajectories is given by the essential dynamics analysis. This analysis simply takes all trajectories and identifies collective motion, which can be useful to find reaction paths or to construct reduced-dimensionality models.

See Section 7.35 (p. 224) in the manual.

The interactive script can be started with

```
user@host> $SHARC/trajana_essdyn.py
```

See Section 8.8 (p. 246) in the manual.

```
  ================================================================================
||                                                                              ||
||                Essential dynamics analysis for SHARC dynamics                ||
||                                                                              ||
||                     Author: Felix Plasser, Andrew Atkins                     ||
||                                                                              ||
||                                  Version:4.0                                 ||
||                                   01.04.2025                                 ||
||                                                                              ||
  ================================================================================


This script reads output.xyz files and calculates the essential dynamics
(i.e., Shows you which are the most important motions).

--------------------Paths to trajectories-------------------

Please enter the paths to all directories containing the "TRAJ_0XXXX" directories.
E.g. Sing_2/ and Sing_3/.
Please enter one path at a time, and type "end" to finish the list.
Path:  [end] (autocomplete enabled) Singlet_2/
['TRAJ_00005', 'TRAJ_00004', 'TRAJ_00008', 'plot.gp', 'TRAJ_00009', 'TRAJ_00002',
'TRAJ_00006', 'TRAJ_00010', 'TRAJ_00007', 'TRAJ_00003']
Found 9 subdirectories in total.

Path:  [end] (autocomplete enabled) <ENTER>

Total number of subdirectories: 9

-----------------Path to reference structure---------------

Please enter the path to the equilibrium structure of your system (in the same
atomic order as that given in the dynamics output)

Path:  [ref.xyz] (autocomplete enabled) ../MOLCAS.freq.molden

Please give the type of coordinate file [molden] (autocomplete enabled) <ENTER>

Do you wish to use mass weighted coordinates? [True] <ENTER>

---------Number of total steps in your trajectories---------

Number of time steps:  [201] <ENTER>

--------------The time step of your calculation-------------

Length of time step:  [0.5] <ENTER>

------------------Time steps to be analysed-----------------
```

```
Please enter the time step intervals for which the statistical analysis should be carried out.

Time step interval:  [0 200] 0  40

Do you want to add another time interval for analysis? [False] <ENTER>

----------------------Results directory--------------------
Please give the name of the subdirectory to be used for the results (use to save similar analysis
in separate subdirectories).
Name for subdirectory? [essdyn] (autocomplete enabled) <ENTER>

Preparing essential dynamics analysis ...
Checking the directories...
Singlet_2//TRAJ_00005         OK
Singlet_2//TRAJ_00004         OK
Singlet_2//TRAJ_00008         OK
Singlet_2//TRAJ_00009         DETECTED FILE dont_analyze
Singlet_2//TRAJ_00002         OK
Singlet_2//TRAJ_00006         OK
Singlet_2//TRAJ_00010         DETECTED FILE dont_analyze
Singlet_2//TRAJ_00007         OK
Singlet_2//TRAJ_00003         OK
Number of trajectories: 7
Reading trajectory Singlet_2//TRAJ_00005/output.xyz ...
Reading trajectory Singlet_2//TRAJ_00004/output.xyz ...
Reading trajectory Singlet_2//TRAJ_00008/output.xyz ...
Reading trajectory Singlet_2//TRAJ_00002/output.xyz ...
Reading trajectory Singlet_2//TRAJ_00006/output.xyz ...
Reading trajectory Singlet_2//TRAJ_00007/output.xyz ...
Reading trajectory Singlet_2//TRAJ_00003/output.xyz ...
Processing data ...
```

The output of this script is a directory **ESS_DYN/essdyn/**, which contains two subdirectories with the results of the total covariance analysis (**total_cov/**, giving the most active modes) and of the analysis of the average trajectory (**cross_av/**, giving the most active *coherent* modes). Each directory will contain one output file for the chosen time step interval (steps 0 to 40) called **0-40.molden**. The content of the file is similar to that of a frequency calculation, containing the average geometry of the molecule in the interval and the essential dynamics modes. The "frequency" entries for the essential modes give the total activity of the mode, with larger values indicating more active modes. In order to find the most important motions of the molecule, visualize the essential modes with the largest "frequencies". A vector representation of the most important mode in **ESS_DYN/essdyn/cross_av/0-40.molden** is shown in Figure 2.13.

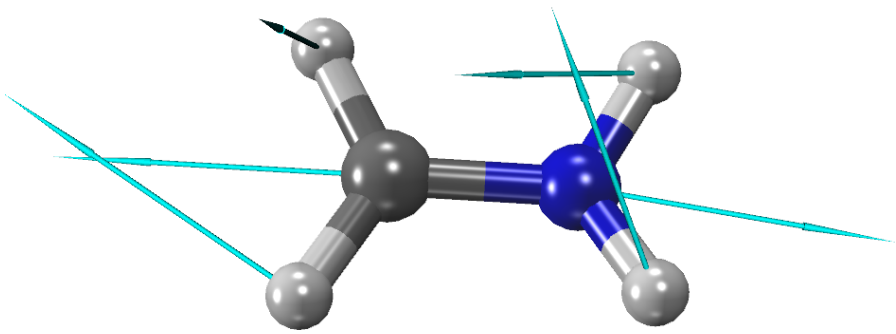

Figure 2.13: Vector representation of the most active essential mode.

2.9.8 Normal Mode Analysis

See Section 7.29 (p. 216) in the manual.

An alternative to the essential dynamics analysis is a normal mode analysis. The difference is that the essential dynamics analysis automatically identifies a suitable set of modes, whereas the normal mode analysis uses a set of precomputed normal modes (e.g., normal modes of the ground state minimum).

In SHARC4, the script **trajana_nma.py** is not present anymore. Instead, a normal mode analysis can be performed with **geo_NM.py** and **data_collector.py**.

In the first step, the normal mode coordinates need to be defined. We will use the normal mode coordinates from the ground state frequency calculation that defined the Wigner distribution. Convert the frequency Molden file into a normal mode file:

See Section 7.36 (p. 226) in the manual.

```
user@host> $SHARC/wigner.py -l MOLCAS.freq.molden
```

This creates the normal mode file **V0.txt**.

In the second step, in each trajectory, we will compute the normal mode coordinates, similar to when we computed the bond lengths and angles with **geo.py**. Use **geo_NM.py** to create the output files containing the normal mode coordinates. For this, in each trajectory folder execute:

See Section 8.19 (p. 259) in the manual.

```
user@host> $SHARC/geo_NM.py -v /path/to/V0.txt > Geo_nm.out
```

In the third step, the actual normal mode activity analysis is performed. We need to run **data_collector.py** twice. In the first run, we will compute for each normal mode the average value over all trajectories, then compute the standard deviation over all time steps. In the second run, we will compute the standard deviation over all time steps directly, without prior averaging.

```
================================================================================
||                                                                            ||
||                   Reading table data from SHARC dynamics                   ||
||                                                                            ||
||                  Authors: Sebastian Mai, Severin Polonius                  ||
||                                Version:4.0                                 ||
||                                  01.04.25                                  ||
||                                                                            ||
================================================================================

This script collects table data from SHARC trajectories, smooths them, synchronizes them,
convolutes them, and computes averages and similar statistics.

-------------------Paths to trajectories--------------------

Please enter the paths to all directories containing the "TRAJ_0XXXX" directories.
E.g. Sing_2/ and Sing_3/.
Please enter one path at a time, and type "end" to finish the list.
Path:  [end] (autocomplete enabled) Singlet_1
['TRAJ_00005']
Found 1 subdirectories in total.

Path:  [end] (autocomplete enabled) Singlet_2
['TRAJ_00010', 'TRAJ_00003', 'header', 'TRAJ_00002', 'TRAJ_00008', 'TRAJ_00004', 'TRAJ_00007',
 'TRAJ_00009', 'TRAJ_00006', 'TRAJ_00001']
Found 10 subdirectories in total.

Path:  [end] (autocomplete enabled) Singlet_3
['TRAJ_00004', 'TRAJ_00001']
Found 12 subdirectories in total.

Path:  [end] (autocomplete enabled) <ENTER>
```

```
Total number of subdirectories: 12

Checking the directories...
Number of trajectories: 12

Do you want to see all common files before specifying the filepath to analyse?:
Yes or no?: [True] <ENTER>
Checking for common files...

List of files common to the trajectory directories:

 Index Number of appearance   Relative file path
-------------------------------------------------------
     0                  12   ./Geo_nm.out
     1                  12   ./output.lis

Please give the relative file path of the file you want to collect:
File path or index: [0] <ENTER>

------------------------Data columns-----------------------

Number of columns in the file:   23

Please select the data columns for the analysis:
For T column:
  only enter one (positive) column index.
  If 0, the line number will be used instead.
For X column:
  enter one or more column indices.
  If 0, all entries of that column will be set to 1.
  If negative, the read numbers will be multiplied by -1.
For Y column:
  enter as many column indices as for X.
  If 0, all entries of that column will be set to 1.
  If negative, the read numbers will be multiplied by -1.

T column (time): [1] <ENTER>
X columns: [2] (range comprehension enabled) 8~19
Y columns: [0 0 0 0 0 0 0 0 0 0 0 0] (range comprehension enabled) <ENTER>
Selected columns:
T: 1     X: [8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]    Y: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

----------------------Analysis procedure----------------------

Show possible workflow options? [False] <ENTER>

--------------0 Collecting--------------

Do you want to write Type 1 files? [False] <ENTER>

--------------1 Smoothing---------------

Do you want to apply smoothing to the individual trajectories? [False] <ENTER>

------------2 Synchronizing-------------

Do you want to synchronize the data? [True] <ENTER>

---------3 Convoluting along X----------
```

```
Do you want to apply convolution in X direction? [False] <ENTER>

--------------4 Averaging---------------

Do you want to average the data columns across all trajectories? [False]  yes
Choose one of the following options:
1  Arithmetic average and standard deviation
2  Geometric average and standard deviation
Choose one of the options: [1] <ENTER>

-----------5 Total statistics-----------

Do you want to compute the total mean and standard deviation over all time steps? [False]  yes
Choose one of the following options:
1  Arithmetic average and standard deviation
2  Geometric average and standard deviation
Choose one of the options: [1] <ENTER>


##################################Full input###################################

paths                       ['/global/mai/SHARC_4.0_beta_tests/TUTORIAL/Tutorial/traj/Singlet_1', ...]
filepath                    ./Geo_nm.out
allfiles                    ['Singlet_1/TRAJ_00005/./Geo_nm.out', ...]
ncol                        23
colT                        1
colX                        [8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
nX                          12
colY                        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
nY                          12
write_type1                 False
smoothing
synchronizing               True
averaging                   'pre': None, 'post': None
statistics                  'pre': None, 'post': None
convolute_X
sum_Y                       False
integrate_X
convolute_T
integrate_T
type3_to_type2              False

Do you want to do the specified analysis? [True] <ENTER>

>>>>>>>>>>>>>>>>>>>> Started data analysis

Collecting the data ...
  Progress: [==================================================] 100%  Done
>>>> Skipping output to file ...

Synchronizing temporal data ...
  Progress: [==================================================] 100%  Done
>>>> Writing output to file "collected_data_1_8910111213141516171819_000000000000_sy.type2.txt"...

Computing averages ...
  Progress: [==================================================] 100%  Done
>>>> Writing output to file "collected_data_1_8910111213141516171819_000000000000_sy_av.type2.txt"...

Computing total statistics ...
```

```
  Progress: [==================================================] 100%  Done
>>>> Writing output to file "collected_data_1_8910111213141516171819_000000000000_sy_av_st.type2.txt"...
```

The second run should be carried out exactly in the same way, just answering with **no** to the question **Do you want to average the data columns across all trajectories?**.

The two runs produce two files that are relevant for the normal mode activity analysis, these are on one hand **collected_data_1_8910111213141516171819_000000000000_sy_st.type2.txt** and on the other hand **collected_data_1_8910111213141516171819_000000000000_sy_av_st.type2.txt**. The first file contains the following results:

- Column 1: Time t
- Columms 2–13: Total averages over all trajectories and all time steps from zero to t, one column per normal mode
- Columns 14–25: Total standard deviations over all trajectories and all time steps from zero to t, one column per normal mode

The second file contains the following results:

- Column 1: Time t
- Columms 2–13: Averages over the average trajectory and all time steps from zero to t, one column per normal mode
- Columns 26–37: Standard deviations over the average trajectory and all time steps from zero to t, one column per normal mode

The normal mode activity can now be computed by taking the total standard deviation of a mode and subtracting the standard deviation of the average trajectory for the same mode. In the example, to get the activity of mode 7 (the first vibrational normal mode) at time 200 fs, one would take the last lines from both files, and compute the difference of column 14 in the **sy_st** file and column 26 in the **sy_av_st** file.

2.9.9 Ensemble Motion Plots

See Section 7.36 (p. 226) in the manual.

In addition to the statistical analysis of the nuclear motion (**trajana_essdyn.py** and **trajana_nma.py**), it is often helpful to plot some bond length, angle, internal coordinate, etc. for all trajectories, in what could be called "hair figures" or heat maps. For such plots (and many others), **data_collector.py** can merge the corresponding per-trajectory data into files which can then be conveniently plotted.

In the following, we will collect the time-dependent C=N bond length from all trajectories and plot them. The first step is to compute this bond length for each trajectory. This can be achieved with **geo.py** and a simple Bash loop:

```
user@host> echo "r 1 2" > Geo.inp
user@host> for i in Singlet_2/TRAJ_000*;
user@host> do
user@host> $SHARC/geo.py -t 0.5 -g $i/output.xyz < Geo.inp > $i/Geo.out;
user@host> done
```

This will create a file **Geo.out** for each trajectory with the bond length between atoms 1 and 2.

Then, start the **data_collector.py** to merge the data into a single file (note that the excluded trajectories are ignored):

```
user@host> $SHARC/data_collector.py
```

```
Script for data collecting started...


================================================================================
||                                                                            ||
||                   Reading table data from SHARC dynamics                   ||
||                                                                            ||
||                  Authors: Sebastian Mai, Severin Polonius                  ||
||                                Version:4.0                                 ||
||                                  26.01.24                                  ||
||                                                                            ||
================================================================================


This script collects table data from SHARC trajectories, smooths them, synchronizes them,
convolutes them, and computes averages and similar statistics.

--------------------Paths to trajectories------------------

Please enter the paths to all directories containing the "TRAJ_0XXXX" directories.
E.g. Sing_2/ and Sing_3/.
Please enter one path at a time, and type "end" to finish the list.
Path:  [end] (autocomplete enabled) Singlet_2/
['TRAJ_00005', 'TRAJ_00004', 'TRAJ_00008', 'plot.gp', 'TRAJ_00009', 'TRAJ_00002',
'TRAJ_00006', 'TRAJ_00010', 'TRAJ_00007', 'TRAJ_00003']
Found 9 subdirectories in total.

Path:  [end] (autocomplete enabled) <ENTER>

Total number of subdirectories: 9

Checking the directories...
Number of trajectories: 7

Do you want to see all common files before specifying the filepath to analyse?:
```

```
Yes or no?: [True] <ENTER>
Checking for common files...

List of files common to the trajectory directories:

 Index Number of appearance   Relative file path
------------------------------------------------------------
     0                    7   ./Geo.out
     1                    7   ./nma_nma.txt
     2                    7   ./nma_nma_av.txt
     3                    7   ./nma_nma_std.txt
     4                    7   ./output.lis
     5                    7   output_data/coeff_MCH.out
     6                    7   output_data/coeff_diab.out
     7                    7   output_data/coeff_diag.out
     8                    7   output_data/energy.out
     9                    7   output_data/expec.out
    10                    7   output_data/expec_MCH.out
    11                    7   output_data/fosc.out
    12                    7   output_data/fosc_act.out
    13                    7   output_data/prob.out
    14                    7   output_data/spin.out

Please give the relative file path of the file you want to collect:
File path or index: [0] ./Geo.out

------------------------Data columns------------------------

Number of columns in the file:   2

Please select the data columns for the analysis:
For T column:
  only enter one (positive) column index.
  If 0, the line number will be used instead.
For X column:
  enter one or more column indices.
  If 0, all entries of that column will be set to 1.
  If negative, the read numbers will be multiplied by -1.
For Y column:
  enter as many column indices as for X.
  If 0, all entries of that column will be set to 1.
  If negative, the read numbers will be multiplied by -1.

T column (time): [1] <ENTER>
X columns: [2] (range comprehension enabled) <ENTER>
Y columns: [0] (range comprehension enabled) <ENTER>
Selected columns:
T: 1     X: [2]    Y: [0]

---------------------Analysis procedure---------------------

Show possible workflow options? [False] <ENTER>

--------------0 Collecting--------------

Do you want to write Type 1 files? [False] <ENTER>

---------------1 Smoothing--------------

Do you want to apply smoothing to the individual trajectories? [False] <ENTER>
```

```
------------2 Synchronizing------------

Do you want to synchronize the data? [True] <ENTER>


----------3 Convoluting along X---------

Do you want to apply convolution in X direction? [False] yes

Choose one of the following convolution kernels:
1  Gaussian function
2  Lorentzian function
3  Rectangular window function
4  Log-normal function
Choose one of the functions: [1] <ENTER>
Choose width of the smoothing function (in units of the X columns): [1.0] 0.2
Size of the grid along X: [25] 50

Choose minimum and maximum of the grid along X:
Enter either a single number a (X grid from  xmin-a*width  to  xmax+a*width)
        or two numbers a and b (X grid from  a  to  b)
Xrange: [1.0] <ENTER>

------------6 Sum over all Y------------

Do you want to sum up all Y values? [False] <ENTER>

-----------7 Integrate along X----------

Do you want to integrate in X direction? [False] <ENTER>

----------8 Convoluting along T---------

Do you want to apply convolution in T direction? [False] <ENTER>

----------9 Integrating along T---------

Do you want to integrate in T direction? [False] <ENTER>

-------10 Convert to Type2 dataset------

If you performed integration along X, the data might be better formatted as Type2 dataset.
Do you want to output as Type2 dataset? [False] <ENTER>


########################Full input#########################

paths                       ['Singlet_2/']
filepath                    ./Geo.out
allfiles                    ['Singlet_2//TRAJ_00002/./Geo.out',
                             'Singlet_2//TRAJ_00003/./Geo.out',
                             'Singlet_2//TRAJ_00004/./Geo.out',
                             'Singlet_2//TRAJ_00005/./Geo.out',
                             'Singlet_2//TRAJ_00006/./Geo.out',
                             'Singlet_2//TRAJ_00007/./Geo.out',
                             'Singlet_2//TRAJ_00008/./Geo.out']
ncol                        2
colT                        1
colX                        [2]
```

```
nX                      1
colY                    [0]
nY                      1
write_type1             False
smoothing               {}
synchronizing           True
averaging               {}
statistics              {}
convolute_X             {'function': <__main__.gauss instance at 0x1a7ed88>,
                         'npoints': 50, 'xrange': [1.0]}
sum_Y                   False
integrate_X             {}
convolute_T             {}
integrate_T             False
type3_to_type2          False

Do you want to do the specified analysis? [True] <ENTER>

>>>>>>>>>>>>>>>>>>>> Started data analysis

Collecting the data ...
  Progress: [==================================================] 100%
>>>> Writing output to file "collected_data_1_2_0.type1.txt"...

Synchronizing temporal data ...
  Progress: [==================================================] 100%
>>>> Writing output to file "collected_data_1_2_0_sy.type2.txt"...

Convoluting data (along X column) ...
  Progress: [==================================================] 100%
>>>> Writing output to file "collected_data_1_2_0_sy_cX.type3.txt"...
```

This run of the script produces three output files:

- **collected_data_1_2_0.type1.txt**,
- **collected_data_1_2_0_sy.type2.txt**, and
- **collected_data_1_2_0_sy_cX.type3.txt**.

You can plot the contained data using GNUPLOT (do not type the line break):

```
gnuplot> p "collected_data_1_2_0_sy.type2.txt" u 1:2 w l, "" u 1:3 w l,
         "" u 1:4 w l, "" u 1:5 w l, "" u 1:6 w l, "" u 1:7 w l, "" u 1:8 w l
```

or (if using GNUPLOT 5.0 or higher):

```
gnuplot> p for [col=2:8] "collected_data_1_2_0_sy.type2.txt" u 1:col w l
```

For the 3D data in **collected_data_1_2_0_sy_cX.type3.txt**, use:

```
gnuplot> set view map
gnuplot> sp "collected_data_1_2_0_sy_cX.type3.txt" u 1:2:3 w pm3d
```

The results of these two plot commands are shown in Figures 2.14 and 2.15.

Using **data_collector.py**, it is also possible to compute the mean and standard deviations of the bond lengths (giving similar information as **trajana_nma.py** was doing for the normal modes). In order to do so, do not apply convolution in X direction, and then ask for an averaging and/or statistics analysis.

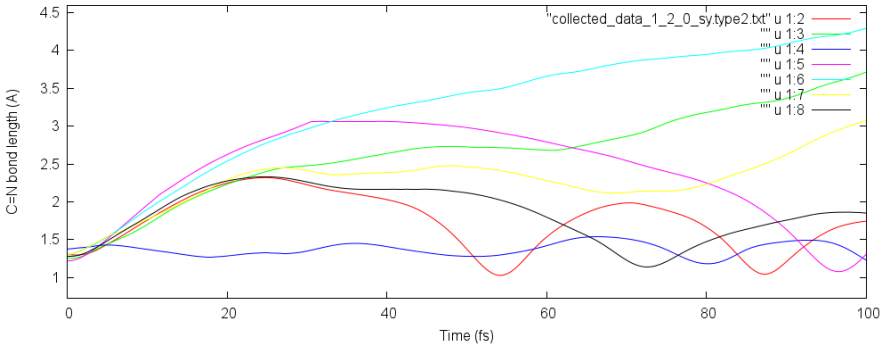

Figure 2.14: All C=N bond lengths from the 7 trajectories.

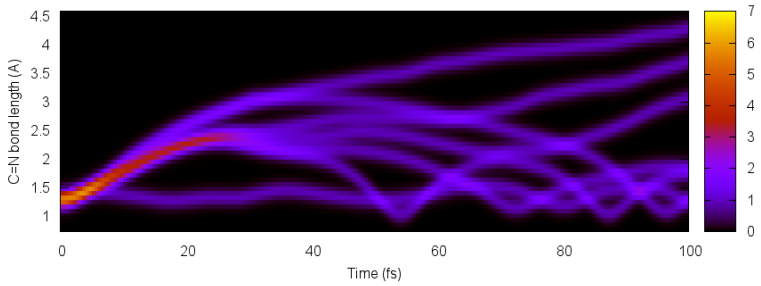

Figure 2.15: Convolution of the C=N bond lengths from the 7 trajectories.

2.9.10 Transient Spectra

Using **data_collector.py**, it is also possible to compute transient absorption spectra, given that enough states were included in the simulations. Here, we will simulate the transient spectrum for $CH_2NH_2^+$, although probably the spectrum will be incomplete due to the missing of higher excited states.

See Section 7.36 (p. 226) in the manual.

Again, start the **data_collector.py** to merge and post-process the data:

```
user@host> $SHARC/data_collector.py
```

```
Script for data collecting started...


================================================================================
||                                                                            ||
||                   Reading table data from SHARC dynamics                   ||
||                                                                            ||
||                  Authors: Sebastian Mai, Severin Polonius                  ||
||                                Version:4.0                                 ||
||                                  26.01.24                                  ||
||                                                                            ||
================================================================================


This script collects table data from SHARC trajectories, smooths them, synchronizes them,
convolutes them, and computes averages and similar statistics.

--------------------Paths to trajectories------------------

Please enter the paths to all directories containing the "TRAJ_0XXXX" directories.
E.g. Sing_2/ and Sing_3/.
Please enter one path at a time, and type "end" to finish the list.
Path:  [end] (autocomplete enabled) Singlet_2/
['TRAJ_00005', 'TRAJ_00004', 'TRAJ_00008', 'plot.gp', 'TRAJ_00009', 'TRAJ_00002',
'TRAJ_00006', 'TRAJ_00010', 'TRAJ_00007', 'TRAJ_00003']
Found 9 subdirectories in total.

Path:  [end] (autocomplete enabled) <ENTER>

Total number of subdirectories: 9

Checking the directories...
Number of trajectories: 7

Do you want to see all common files before specifying the filepath to analyse?:
Yes or no?: [True] <ENTER>
Checking for common files...

List of files common to the trajectory directories:

 Index Number of appearance   Relative file path
-------------------------------------------------------------
     0                    7   ./Geo.out
     1                    7   ./nma_nma.txt
     2                    7   ./nma_nma_av.txt
     3                    7   ./nma_nma_std.txt
     4                    7   ./output.lis
     5                    7   output_data/coeff_MCH.out
     6                    7   output_data/coeff_diab.out
     7                    7   output_data/coeff_diag.out
```

```
     8                   7   output_data/energy.out
     9                   7   output_data/expec.out
    10                   7   output_data/expec_MCH.out
    11                   7   output_data/fosc.out
    12                   7   output_data/fosc_act.out
    13                   7   output_data/prob.out
    14                   7   output_data/spin.out

Please give the relative file path of the file you want to collect:
File path or index: [0] output_data/fosc_act.out

-----------------------Data columns-----------------------

Number of columns in the file:   27

Please select the data columns for the analysis:
For T column:
  only enter one (positive) column index.
  If 0, the line number will be used instead.
For X column:
  enter one or more column indices.
  If 0, all entries of that column will be set to 1.
  If negative, the read numbers will be multiplied by -1.
For Y column:
  enter as many column indices as for X.
  If 0, all entries of that column will be set to 1.
  If negative, the read numbers will be multiplied by -1.

T column (time): [1] <ENTER>
X columns: [2] (range comprehension enabled) 2~14
Y columns: [0 0 0 0 0 0 0 0 0 0 0 0 0] (range comprehension enabled) 15~27
Selected columns:
T: 1     X: [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
         Y: [15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27]

---------------------Analysis procedure---------------------

Show possible workflow options? [False] <ENTER>

--------------0 Collecting--------------

Do you want to write Type 1 files? [False]  <ENTER>

---------------1 Smoothing--------------

Do you want to apply smoothing to the individual trajectories? [False] <ENTER>

-------------2 Synchronizing------------

Do you want to synchronize the data? [True] <ENTER>

----------3 Convoluting along X---------

Do you want to apply convolution in X direction? [False] yes

Choose one of the following convolution kernels:
1  Gaussian function
2  Lorentzian function
3  Rectangular window function
4  Log-normal function
```

```
Choose one of the functions: [1] <ENTER>
Choose width of the smoothing function (in units of the X columns): [1.0] <ENTER>
Size of the grid along X: [25] 50

Choose minimum and maximum of the grid along X:
Enter either a single number a (X grid from  xmin-a*width  to  xmax+a*width)
        or two numbers a and b (X grid from  a  to  b)
Xrange: [1.0] <ENTER>

------------6 Sum over all Y------------

Do you want to sum up all Y values? [False] yes

-----------7 Integrate along X----------

Do you want to integrate in X direction? [False] <ENTER>

----------8 Convoluting along T---------

Do you want to apply convolution in T direction? [False] <ENTER>

----------9 Integrating along T---------

Do you want to integrate in T direction? [False] <ENTER>

-------10 Convert to Type2 dataset------

If you performed integration along X, the data might be better formatted as Type2 dataset.
Do you want to output as Type2 dataset? [False] <ENTER>


########################Full input#########################

paths                       ['Singlet_2/']
filepath                    output_data/fosc_act.out
allfiles                    ['Singlet_2//TRAJ_00002/output_data/fosc_act.out',
                             'Singlet_2//TRAJ_00003/output_data/fosc_act.out',
                             'Singlet_2//TRAJ_00004/output_data/fosc_act.out',
                             'Singlet_2//TRAJ_00005/output_data/fosc_act.out',
                             'Singlet_2//TRAJ_00006/output_data/fosc_act.out',
                             'Singlet_2//TRAJ_00007/output_data/fosc_act.out',
                             'Singlet_2//TRAJ_00008/output_data/fosc_act.out']
ncol                        27
colT                        1
colX                        [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
nX                          13
colY                        [15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27]
nY                          13
write_type1                False
smoothing                   {}
synchronizing               True
averaging                   {}
statistics                  {}
convolute_X                 {'function': <__main__.gauss instance at 0x28e0c20>,
                             'npoints': 50, 'xrange': [1.0]}
sum_Y                       True
integrate_X                 {}
convolute_T                 {}
integrate_T                 False
type3_to_type2              False
```

```
Do you want to do the specified analysis? [True] <ENTER>



>>>>>>>>>>>>>>>>>>>> Started data analysis

Collecting the data ...
  Progress: [==================================================] 100%
>>>> Writing output to "collected_data_1_234567891011121314_151617181920212222324252627.type1.txt"...

Synchronizing temporal data ...
  Progress: [==================================================] 100%
>>>> Writing output to "collected_data_1_234567891011121314_151617181920212222324252627_sy.type2.txt"...

Convoluting data (along X column) ...
  Progress: [==================================================] 100%
>>>> Writing output to "collected_data_1_234567891011121314_151617181920212222324252627_sy_cX.type3.txt"...

Summing all Y values ...
  Progress: [==================================================] 100%
>>>> Writing output to "collected_data_1_234567891011121314_151617181920212222324252627_sy_cX_sY.type3.txt"...
```

The last of the four produced output files contains the total transient absorption spectrum. You can print it in the same way as the convoluted bond lengths in Figure 2.15. The simulated transient absorption spectrum is presented in Figure 2.16.

Using **`data_collector.py`**, it is also possible to convolute the spectrum with an instrument response function, or to integrate it along the time or energy axes.

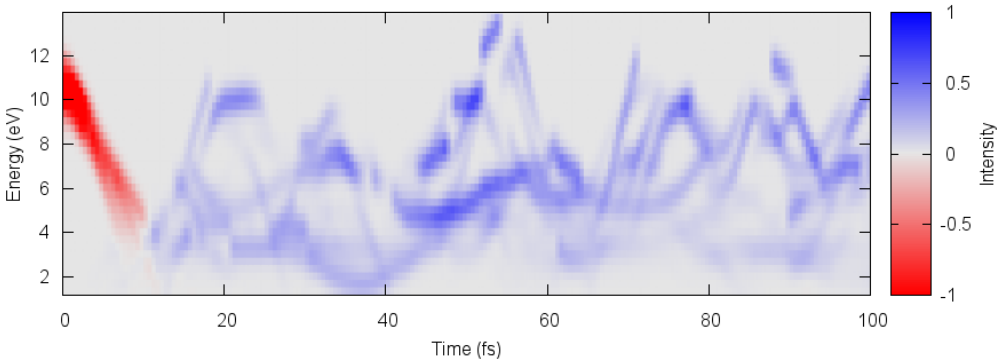

Figure 2.16: Simulated transient absorption spectrum of $CH_2NH_2^+$. Ground state bleech (negative absorption) is shown in red, and excited-state absorption (positive absorption) is shown in blue.

3 Specialized Tutorials

This chapter collects some more tutorials on advanced aspects of the SHARC suite. In order to keep this chapter short and simple, not the full input dialogues are shown, but only the relevant parts. It is thus a good idea to first work through the full tutorial in Chapter 2.

3.1 Using non-default atomic masses

Sometimes, in dynamics simulations one is interested in isotope effects. This necessitates the modification of the atomic mass of the corresponding atoms. The atomic masses are included in the geometry file, which is read by SHARC during initialization of the dynamics.

Furthermore, modification of the atomic masses influences the Wigner distribution of initial conditions. **wigner.py** has a facility to adjust the atomic mass during initial condition generation. Call **wigner.py** as usual, but include the **-m** option:

```
user@host> $SHARC/wigner.py freq.molden -m
```

Instead of directly producing the initial conditions file, **wigner.py** will start a dialog, where the user can modify the atomic masses. Initially, all atoms are assumed to use the default mass (the mass of the most common isotope). The user can then add atoms to the list of atoms with non-default masses, thereby specifying the mass.

```
Initial condition generation started...
Molden file                  = "MOLCAS.freq.molden"
OUTPUT file                  = "initconds"
Number of geometries         = 20
Random number generator seed = 16661
Temperature                  = 0.000000

Option -m used, please enter non-default masses:
+ number mass           add non-default mass <mass> for atom <number>
- number                remove non-default mass for atom <number> (default mass will be used)
show                    show non-default atom masses
end                     finish input for non-default masses

+ 1 14.054321         # Give atom number 1 the mass 14.054321
show
----------------------
Atom               Mass
   1    14.054321000000
----------------------
- 1                   # Give atom number 1 the default mass
+ 2 16.054321         # Give atom number 2 the mass 16.054321
+ 2 15.054321         # Give atom number 2 the mass 15.054321
end
************************************************
WARNING: Less than 3*N_atom normal modes extracted!
************************************************
```

```
Starting normal mode format determination...
Final format specifier: 2 [cartesian (Molpro, Molcas)]
Multiple possible flags have been identified:
  gaussian-type (Gaussian, Turbomole, Q-Chem, ADF, Orca)
  cartesian (Molpro, Molcas)
The most likely assumption is cartesian (Molpro, Molcas) coordinates.
These have been used in the creation of inital conditions.

You can override this behavior by setting the -f [int] flag in the command line:
  1     gaussian-type (Gaussian, Turbomole, Q-Chem, ADF, Orca)
  2     cartesian (Molpro, Molcas)
  3     columbus-type (Columbus)
  4     mass-weighted

Geometry:
 C   6.0   0.00000000  -0.00000000   0.06174827  12.00000000
 N   7.0   0.00000000  -0.00000000   2.46709935  15.05432100
 H   1.0   1.77794247  -0.00000014  -0.94890055   1.00782000
 H   1.0   1.62604543   0.00000016   3.46865331   1.00782000
 H   1.0  -1.62604543  -0.00000016   3.46865331   1.00782000
 H   1.0  -1.77794247   0.00000014  -0.94890055   1.00782000
Assumed Isotopes: H-1 C-12 N-14
Isotopes with * are pure isotopes.

Frequencies (cm^-1) used in the calculation:
   1     977.8746
   2    1022.0225
   3    1215.0055
   4    1314.4262
   5    1418.2342
   6    1537.7263
   7    1679.1657
   8    1882.6738
   9    3329.5794
  10    3463.0910
  11    3679.5594
  12    3763.5119

Sampling initial conditions
Progress: [==================================================] 100%
```

Note that with the **+** command the mass of an already specified atom can also be changed.

Frequency calculation Please also note that the preceding frequency calculation also has to use the same atomic masses as the run of **wigner.py**. For MOLCAS, after running **molcas_input.py** it will be necessary to adjust the input file by adding the appropriate keywords (consult the MOLCAS manual). For MOLPRO, **molpro_input.py** can setup frequency calculations with non-standard atomic masses. For COLUMBUS, the user can edit the **geom** before starting the frequency calculation.

3.2 Inactive states

Sometimes, states should be included in the dynamics simulation in order to calculate transient absorption spectra, transient ionization spectra or simply in order to see how these states evolve during the dynamics. When it is not expected that these states are actually occupied during the simulation, SHARC has a mechanism to neglect all couplings between these states (called inactive henceforth) and the occupied states. This allows also to neglect the gradients and non-adiabatic couplings involving these states, thereby saving considerable amounts of computation time.

It is only possible to make the highest states in each multiplicity inactive. With **`states 3 0 3`** and **`actstates 2 0 1`** in the SHARC input, one would include S_0, S_1, S_2, T_1, T_2 and T_3 in the simulation, but restrict the actual dynamics to S_0, S_1 and T_1. This is useful, e.g., to compute a transient absorption spectrum involving the higher states.

It is advisable to make inactive all states which do not show any couplings with the occupied states (e.g., ionic states in a dynamics simulation of a neutral molecule).

Inactive states in `excite.py` **`excite.py`** also allows to exclude states in the excitation process. Note, however, that this is unrelated to the inactive states in the dynamics. It would be possible to exclude a state in the excitation process but still keep it in the dynamics simulation. An example would be to excite to dark $n\pi^*$ states, which would not be selected if bright $\pi\pi^*$ states are present.

3.3 Ionization spectra

With some of the SHARC interfaces, it is possible to compute Dyson norms between pairs of neutral and ionic states. These Dyson norms are very simple estimates for the single-photon ionization probability, and therefore allow to compute approximate ionization spectra.

See Section 7.9 (p. 188) in the manual.

In order to compute ionization spectra, one needs first to prepare an **initconds** file, e.g., using **wigner.py**. Subsequently, one uses **setup_init.py** to setup the necessary single point calculations, **excite.py** to read out the resulting data, and **spectrum.py** to plot the ionization spectrum.

In order to prepare the computation (assuming you did the full tutorial beforehand):

```
user@host> mkdir ionization
user@host> cp MOLCAS.template initconds ionization/
user@host> cd ionization/
```

See Section 7.10 (p. 190) in the manual.

Then, modify **MOLCAS.template** by setting the number of states for averaging to **roots 4 3 3** and the number of electrons to **nactel 7**. The other settings (basis set, RAS2, inactive) should stay the same as in the full tutorial (**basis cc-pVDZ**, **ras2 4**, **inactive 5**). With this setup, the interface will do the odd multiplicities (doublets) with CAS(7,4) and the even multiplicities (singlets, triplets) with CAS(6,4), because the interface will always either use the given **nactel** or **nactel**−1, as appropriate.

See Section 6.0.1 (p. 76) in the manual.

3.3.1 setup_init.py Input

Carry out the **setup_init.py** input dialogue as usual, but request **4 3 3** states and then the computation of Dyson norms:

```
----------------------Number of states----------------------

Please enter the number of states as a list of integers
e.g. 3 0 3 for three singlets, zero doublets and three triplets.
Number of states: 4  3  3

Number of states: [4, 3, 3]
Total number of states: 19

⋮  ⋮  ⋮  ⋮  ⋮  ⋮  ⋮  ⋮

------------Ionization probability by Dyson norms-----------

Do you want to compute Dyson norms between neutral and ionic states?
Dyson norms? [False] yes
```

The latter question is only asked if you requested doublet states and the interface can compute Dyson norms (which the MOLCAS interface can). Finish the setup as usual and run the calculations.

3.3.2 excite.py Input

Use **excite.py** as usual to read the results of the excited-state calculations. Tell the script to consider the Dyson norms instead of the transition dipole moments, and tell it to assume excitation from state 5 (with **states 4 3 3**, state 5 is the lowest doublet state).

```
Use Dyson norms instead of dipole moments? [False] yes

┊  ┊  ┊  ┊  ┊  ┊  ┊  ┊

----------------------Considered states---------------------

From which state should the excitation originate (for computation of excitation energies
and oscillator strength)?
Lower state for excitation? [1] 5
```

The script will use the Dyson norms in the place of the x component of the transition dipole moments, while the y and z components will be set to zero. The remaining usage is identical to the regular case (transition dipole moments). **`spectrum.py`** can be used to obtain plottable spectra.

3.4 Initial Conditions from Dynamics Simulations

Instead of generating the initial geometries/velocities with **wigner.py**, one can sample them from a molecular dynamics simulation (usually in the ground state). Within SHARC, there are two possibilities: (i) convert SHARC trajectories, (ii) convert AMBER restart files. The first option is described below, for the second option, please refer to the Manual.

See Section 7.6 (p. 185) in the manual.

3.4.1 Converting SHARC Trajectories

In order to convert SHARC trajectories back into an **initconds** file, use the script **sharctraj_to_initconds.py**. Here, we will convert the trajectories ran within the full tutorial (chapter 2). More specifically, we will take the geometry from a random time step which is later than the first 100 steps (because in the beginning the system needs to relax), and earlier than the last 20 steps (because the last time step might be a crash and we do not want to start the new simulations at the related geometry).

See Section 7.4 (p. 183) in the manual.

This can be done with the following (assuming you are in the **traj/** directory):

```
user@host> $SHARC/sharctraj_to_initconds.py -S 100 -20 -o initconds_II Singlet_2/
Initial condition generation started...
directories                   = "['Singlet_2/']"
Random number generator seed  = 16661
Pick randomly from these steps = 100 to -20  (negative indices are counted from the end)
OUTPUT file                   = "initconds_II"
Structure     0: Singlet_2/TRAJ_00002/output.dat  Step:   178/  200  (Reference geometry)
Structure     1: Singlet_2/TRAJ_00002/output.dat  Step:   131/  200  (Saved for initconds)
Structure     2: Singlet_2/TRAJ_00003/output.dat  Step:   150/  200  (Saved for initconds)
Structure     3: Singlet_2/TRAJ_00004/output.dat  Step:   169/  200  (Saved for initconds)
Structure     4: Singlet_2/TRAJ_00005/output.dat  Step:   116/  200  (Saved for initconds)
Structure     5: Singlet_2/TRAJ_00006/output.dat  Step:   181/  200  (Saved for initconds)
Structure     6: Singlet_2/TRAJ_00007/output.dat  Step:   143/  200  (Saved for initconds)
Structure     7: Singlet_2/TRAJ_00008/output.dat  Step:   109/  200  (Saved for initconds)
```

The script will then proceed to check the trajectories. Those marked with files **CRASHED**, or **DONT_ANALYZE** will be ignored (like in all analysis scripts). The other trajectories are all 200 steps long, and therefore the script picks in the time step range 100–180. Each trajectory will give rise to one new initial condition, with the exception that the first trajectory will also be used to create the reference geometry (this is not a real equilibrium geometry, but the **initconds** file format requires some reference geometry). In the end, the script will write the file **initconds_II** (according to the **-o** option), which can be used to compute spectra or setup new trajectories.

3.5 Setting up Laser Fields

In order to generate a laser field file for a SHARC simulation driven by a laser field, one can use the program **laser.x**.

See Section 7.12 (p. 196) in the manual.

```
user@host> $SHARC/laser.x
```

In this example, a short Gaussian laser pulse is set up, using a central wave length of 112 nm (equivalent to 11 eV, the excitation energy of $CH_2NH_2^+$ with the method used in the full tutorial (chapter 2). The laser file is prepared for a 100 fs long SHARC simulation with 0.5 fs step and 25 substeps per step (giving 100 fs/0.5 fs*25+1=5001 steps).

See Section 8.16 (p. 255) in the manual.

```
 Number of lasers:
1
           1
 Real-valued field (T) or not (F):
T
 T
 Set starting time, end of time and number of time steps (t0[fs],tEnd[fs],Nt) :
0 100 5001          # one step for each substep in sharc.x
  0.000000000000000E+000   100.000000000000             5001
 consequently, we have a step size of   2.000000000000000E-002
 Write additional files for debugging (T) or not (F):
F
 F
              (Empty line to increase readability. Press Enter.)
<ENTER>

 Choose polarization vector (e.g. 2.,0.,0. will be normalized):
0 0 1         # orientation of transition dipole moment of CH2NH2+
  0.000000000000000E+000  0.000000000000000E+000   1.00000000000000
 Choose type of envelope (1=Gaussian,2=Sinusoidal):
1
           1
 Choose field strength in (1) [GV/m] (2) [TW/cm^2] (3) [a.u.]:
2
 Enter field strength:
0.1
  0.100000000000000       TW/cm^2
 =   0.868551141020405       GV/m
 =   1.688139437973199E-003  a.u.
 Set FWHM,begin,center,center2 and end time of pulse (1 + 3 affect Gaussian, 2-5
  affect Sinus) [fs]:
25.0 0 50.0 0 0
   25.0000000000000       0.000000000000000E+000    50.000000000000
  0.000000000000000E+000  0.000000000000000E+000
 Choose central frequency in (1) [nm] (2) [eV] (3) [a.u.]:
2
 Enter central frequency:
11.0
   11.0000000000000       eV
 =    112.692053577037       nm
 =   0.404242584583466       a.u.
 Choose phase (as multiple of pi):
0
  0.000000000000000E+000
 Choose colored double pulse (b_1[fs]), which is multiplied with abs(omega-omega
 _0):
0
```

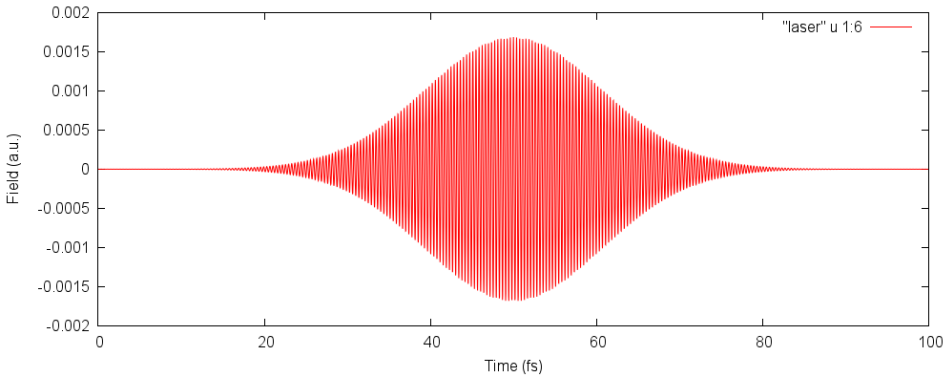

Figure 3.1: Plot of the z component of the generated laser field.

```
  0.000000000000000E+000
 Choose linear chirp (b_2[fs^2]):
0
  0.000000000000000E+000
 Choose third-order chirp (b_3[fs^3]):
0
  0.000000000000000E+000
 Choose fourth-order chirp (b_4[fs^4]):
0
  0.000000000000000E+000

 Done with input.
 Writing out laser field
```

The program writes a file called **`laser`**, which contains a table with the field strengths for each time step (with time in the first column, then real and imaginary part of x component, then of y, then of z). The field strength in z direction (the polarization chosen in the input section) is plotted in Figure 3.1.

3.6 Setting up LVC models

See Section 6.4 (p. 86) in the manual.

Linear vibronic coupling (LVC) models are analytical model potentials that can be automatically paramatrized from a small number of quantum chemistry calculations. Subsequently, running SHARC on these model potentials can be an extremely efficient (hundreds of time steps per second with **pysharc**) way to simulate nonadiabatic dynamics.

In order to setup an LVC model, one has to perform three main steps:

1. Obtain the reference potential from a frequency calculation,
2. Perform quantum chemistry calculations,
3. Collect results and derive parameters.

See Section 6.4.4 (p. 88) in the manual.

Here, we parametrize an LVC model for $CH_2NH_2^+$, using the settings and files from the full tutorial in Chapter 2. Note that, actually, a harmonic LVC model is a very bad model for this molecule, but for the goals of this tutorial, it is sufficient. To continue, we need the **MOLCAS.freq.molden**, **MOLCAS.template**, and **MOLCAS.RasOrb** files obtained in Chapter 2.

3.6.1 Reference potential

In the LVC models used in SHARC, all diabatic potentials are shifted versions of a harmonic reference potential. This harmonic potential is fully specified from a frequency calculation, and is saved in a **V0.txt** file. This file can be generated from **wigner.py**:

```
user@host> $SHARC/wigner.py -l MOLCAS.freq.molden
```

Note how with the **-l** option, **wigner.py** does not produce an **initconds** file, but only the **V0.txt**.

3.6.2 Performing the quantum chemistry calculations

The quantum chemistry calculations to parametrize the LVC model can be setup with **setup_LVCparam.py**:

```
user@host> $SHARC/setup_LVCparam.py
```

```
Script for setup of LVC parametrization started...

  ================================================================================
||                                                                              ||
||                    LVC parametrization for SHARC dynamics                    ||
||                                                                              ||
||             Author: Simon Kropf, Sebastian Mai, Severin Polonius             ||
||                                                                              ||
||                                  Version:4.0                                 ||
||                                    01.01.24                                  ||
||                                                                              ||
  ================================================================================

This script automatizes the setup of excited-state calculations
in order to parametrize LVC models for SHARC dynamics.
------------------------V0.txt file-----------------------

Ground-state file "V0.txt" detected. Do you want to use this?
Use file "V0.txt"? [True] <ENTER>

File "V0.txt" contains 6 atoms and we will use 12 frequencies/normal modes.
(others are zero)
```

```
----------------------Number of states----------------------

Please enter the number of states as a list of integers
e.g. 3 0 3 for three singlets, zero doublets and three triplets.
Number of states: 2  0  1

Please enter the molecular charge for each chosen multiplicity
e.g. 0 +1 0 for neutral singlets and triplets and cationic doublets.
Molecular charges per multiplicity: [0 1 0] 1 0 1

Number of states: [2, 0, 1]
Total number of states: 5

-----------Choose the quantum chemistry interface-----------

Please specify the quantum chemistry interface (enter any of the following numbers):
  1 SHARC_ADAPTIVE          HYBRID interface for adaptive sampling
  2 SHARC_AMS_ADF                  (Not Available! Use SHARC_LEGACY to work with this interface)
  3 SHARC_ANALYTICAL          FAST interface for analytical model Hamiltonians with sympy
  4 SHARC_ASE_DB            HYBRID interface for saveing data to ASE db
  5 SHARC_BAGEL                    (Not Available! Use SHARC_LEGACY to work with this interface)
  6 SHARC_COLUMBUS                 (Not Available! Use SHARC_LEGACY to work with this interface)
  7 SHARC_DO_NOTHING          FAST interface for testing (zero E/grad/NAC/DM, unity overlaps/phases)
  8 SHARC_ECI               HYBRID interface for excitonic HF/CI with multiple fragments
  9 SHARC_FALLBACK          HYBRID interface for calling a fallback interface if primary interface fails
 10 SHARC_GAUSSIAN       AB INITIO interface for GAUSSIAN16 for single-reference methods (CIS, TDDFT)
 11 SHARC_LEGACY             BASIC interface for running legacy interfaces via file I/O (AMS-ADF, BAGEL, ...
 12 SHARC_LVC                FAST interface for linear/quadratic vibronic coupling models (LVC, QVC, ...
 13 SHARC_MNDO           AB INITIO interface for the MNDO program (OM2-MRCI)
 14 SHARC_MOLCAS         AB INITIO interface for OpenMolcas (>v23) for multireference calculations ...
 15 SHARC_MOLPRO                  (Not Available! Use SHARC_LEGACY to work with this interface)
 16 SHARC_MOPACPI        AB INITIO interface for the MOPAC-PI program
 17 SHARC_NUMDIFF           HYBRID interface for numerical derivatives (grad, NACdr, SOCdr, DMdr)
 18 SHARC_NWCHEM         AB INITIO interface for NWChem (TDDFT)
 19 SHARC_OPENMM                  (Not Available!)
 20 SHARC_ORCA           AB INITIO interface for ORCA v5-6 (CIS/TDDFT)
 21 SHARC_PYSCF                   (Not Available! Use SHARC_LEGACY to work with this interface)
 22 SHARC_QMMM              HYBRID interface for QM/MM (electrostatic embedding, link atom scheme)
 23 SHARC_QMOUT              FAST interface for frozen-nuclei dynamics (constant E/SOC/DM, unit ...
 24 SHARC_SPAINN             FAST interface for SPaiNN
 25 SHARC_TURBOMOLE      AB INITIO interface for TURBOMOLE (RICC2/ADC2)
 26 SHARC_UMBRELLA          HYBRID interface for adding umbrella-sampling-style restraints (harmonic ...

Interface number: 14
The following interface was selected:
 14 SHARC_MOLCAS         AB INITIO interface for OpenMolcas (>v23) for multireference calculations ...
-----------------Spin-orbit couplings (SOCs)----------------

Do you want to compute spin-orbit couplings?

Spin-Orbit calculation? [True] <ENTER>
Will calculate spin-orbit matrix.
Calculate gradients of SOCs in linear approx.? [False] <ENTER>

--------------------Analytical gradients--------------------

Do you want to use analytical gradients for kappa terms? [True] <ENTER>

Analytical gradients for kappas: True
```

```
Do you want to calculate second order terms (gammas)? [False] <ENTER>
----------Analytical nonadiabatic coupling vectors----------

Do you want to use analytical nonadiabatic coupling vectors for lambda terms? [False] <ENTER>
Do you want to use analytical nonadiabatic coupling vectors for lambdas: False

-----------------------Normal modes-----------------------

Do you want to make LVC parameters for all normal modes? [True] <ENTER>

We will use the following normal modes: (7~18)

-----------------------Displacements----------------------

Do you want to use other displacements than the default of [0.05]? [False] <ENTER>

Script will use displacement magnitudes of:

(7~18): 0.05


-----------------------Intruder states----------------------

  Intruder states can be detected by small overlap matrix elements.
  Affected numerical kappa/lambda terms will be ignored and not written to the parameter file.

Do you want to check for intruder states? [True] <ENTER>

Ignore problematic states: False

------------------One-/Two-sided derivation-----------------

One-/Two-sided derivation of normal modes.
Choose for which normal modes you want to use one-sided derivation (7~18):
[None] (range comprehension enabled) <ENTER>

One-sided derivation will be used on: [None]

Do you want to fit an atomwise multipolar density representation for each state? [False] <ENTER>
================================================================================
||                                                                            ||
||                          MOLCAS interface setup                            ||
||                                                                            ||
================================================================================



Specify path to MOLCAS.
Path to MOLCAS: [$MOLCAS] (autocomplete enabled) /usr/license/openmolcas/


Specify a scratch directory. The scratch directory will be used to run the calculations.
Path to scratch directory: (autocomplete enabled) $TMPDIR/LVC/
Specify a path to a MOLCAS template file.
Template path: (autocomplete enabled) MOLCAS.template
Specify the number of CPUs to be used.
Number of CPUs: [1] <ENTER>
Specify the amount of RAM to be used.
Memory (MB): [1000] 500
Initial wavefunction: MO Guess
```

```
Please specify the path to a MOLCAS JobIph file containing suitable starting MOs for the CASSCF calculation.
Please note that this script cannot check whether the wavefunction file and the Input template are consistent!
Do you have initial wavefunction files for multiplicities 1 3? [True] <ENTER>
JobIph files (1) or RasOrb files (2)? 2
Initial wavefunction file for multiplicity 1: [MOLCAS.1.RasOrb.init] (autocomplete enabled) MOLCAS.RasOrb
Initial wavefunction file for multiplicity 3: [MOLCAS.3.RasOrb.init] (autocomplete enabled) MOLCAS.RasOrb


  ================================================================================
||                                Run mode setup                                ||
  ================================================================================


------------------------Run script------------------------

  This script can generate the run scripts for each initial condition in two modes:

    - In mode 1, the calculation is run in subdirectories of the current directory.

    - In mode 2, the input files are transferred to another directory (e.g. a local
      scratch directory), the calculation is run there, results are copied back and
      the temporary directory is deleted. Note that this temporary directory is not
      the same as the "scratchdir" employed by the interfaces.

  Note that in any case this script will create the input subdirectories in the
  current working directory.

  In case of mode 1, the calculations will be run in:
'/user/mai/Documents/NewSHARC/SHARC_2.1/TUTORIAL/LVC/setup'

Use mode 1 (i.e., calculate here)? [True] <ENTER>

---------------------Submission script--------------------

During the setup, a script for running all initial conditions sequentially in batch
mode is generated. Additionally, a queue submission script can be generated for all
initial conditions.
Generate submission script? [False] <ENTER>


#########################Full input#########################

v0f                     /user/mai/Documents/NewSHARC/SHARC_4.0/TUTORIAL/LVC/setup/V0.txt
atoms                   ...
normal_modes            7: [...]
                        8: [...]
                        9: [...]
                        10: [...]
                        11: [...]
                        12: [...]
                        13: [...]
                        14: [...]
                        15: [...]
                        16: [...]
                        .
                        (2 more)
                        .
freqencies              7: 0.0044375993
                        8: 0.0046569568
```

```
                          9: 0.005551635
                          10: 0.0059837902
                          11: 0.0064624626
                          12: 0.0070062134
                          13: 0.0076503229
                          14: 0.0085770748
                          15: 0.0151719177
                          16: 0.0157797932
                          .
                          (2 more)
                          .
states                    2
                          0
                          1
nstates                   5
charge                    0
                          1
                          0
needed_requests           'overlap', 'soc'
soc                       True
lambda_soc                False
ana_grad                  True
gammas                    False
ana_nac                   False
do_overlaps               True
displacement_magnitudes   7: 0.05
                          8: 0.05
                          9: 0.05
                          10: 0.05
                          11: 0.05
                          12: 0.05
                          13: 0.05
                          14: 0.05
                          15: 0.05
                          16: 0.05
ignore_problematic_states  False
multipolar_fit            False
fmw_normal_modes          7: [...]
                          8: [...]
                          9: [...]
                          10: [...]
                          11: [...]
                          12: [...]
                          13: [...]
                          14: [...]
                          15: [...]
                          16: [...]
                          .
                          (2 more)
                          .
displacements             10n: [...]
                          10p: [...]
                          11n: [...]
                          11p: [...]
                          12n: [...]
                          12p: [...]
                          13n: [...]
                          13p: [...]
                          14n: [...]
                          14p: [...]
```

```
                          .
                          (14 more)
                          .
result_path               DSPL_RESULTS
paths                     000_eq: DSPL_000_eq
                          007_n: DSPL_007_n
                          007_p: DSPL_007_p
                          008_n: DSPL_008_n
                          008_p: DSPL_008_p
                          009_n: DSPL_009_n
                          009_p: DSPL_009_p
cwd                       /user/mai/Documents/NewSHARC/SHARC_4.0/TUTORIAL/LVC/setup
here                      True
qsub                      False
Do you want to setup the specified calculations? [True] <ENTER>

  ================================================================================
||                            Setting up directories...                           ||
  ================================================================================


Progress: [================================================] 100%
```

The script will generate a subdirectory called **DSPL_RESULTS**. This newly created subdirectory contains the files **displacements.json**, **displacements.log**, and **all_run_dspl.sh**. The first file saves all settings from **setup_LVCparam.py** and is necessary when reading out the results; hence, it should not be touched. The second file is a simple summary of the setup, and is only for user inspection. The third file can be used to run all quantum chemistry calculations, similar to the cases of **setup_init.py** and **setup_traj.py**.

Additionally, the directory contains subdirectories **DSPL_XXX_YY** which contain the inputs for the calculations. **DSPL_000_eq** is the input for the reference calculation, and this calculation must be carried out first. Subsequently, the remaining $6N$ calculations can be run in any order or in parallel. Note that only **DSPL_000_eq** will be present if analytical gradients and nonadiabatic couplings are both requested.

3.6.3 Extracting the parameters

Once all **DSPL_XXX_YY** calculations are finished (**QM.out** present in each directory), the parameters for the LVC model can be extracted. To this end, simply start the relevant script:

```
user@host> $SHARC/create_LVCparam.py
```

The script is fully automatic, using the settings contained in **displacements.json**.

The output will look like:

```
Script for setup of displacements started...

  ================================================================================
||                                                                                ||
||                             Compute LVC parameters                             ||
||                                                                                ||
||            Authors: Severin Polonius, Sebastian Mai, Simon Kropf               ||
||                                                                                ||
||                                  Version:4.0                                   ||
||                                   01.04.2025                                   ||
||                                                                                ||
```

```
  ================================================================================

This script automatizes the setup of excited-state calculations for displacements
for SHARC dynamics.

Data extraction started ...
Number of states: 5
Number of atoms: 6
Kappas: analytical
Lambdas: numerical

Reading files ...

reading QMout_eq at: DSPL_000_eq/QM.out
h, dm, grad
reading displaced QMout at: DSPL_007_p/QM.out
reading displaced QMout at: DSPL_007_n/QM.out
reading displaced QMout at: DSPL_008_p/QM.out
reading displaced QMout at: DSPL_008_n/QM.out
reading displaced QMout at: DSPL_009_p/QM.out
reading displaced QMout at: DSPL_009_n/QM.out
reading displaced QMout at: DSPL_010_p/QM.out
reading displaced QMout at: DSPL_010_n/QM.out
reading displaced QMout at: DSPL_011_p/QM.out
reading displaced QMout at: DSPL_011_n/QM.out
reading displaced QMout at: DSPL_012_p/QM.out
reading displaced QMout at: DSPL_012_n/QM.out
reading displaced QMout at: DSPL_013_p/QM.out
reading displaced QMout at: DSPL_013_n/QM.out
reading displaced QMout at: DSPL_014_p/QM.out
reading displaced QMout at: DSPL_014_n/QM.out
reading displaced QMout at: DSPL_015_p/QM.out
reading displaced QMout at: DSPL_015_n/QM.out
reading displaced QMout at: DSPL_016_p/QM.out
reading displaced QMout at: DSPL_016_n/QM.out
reading displaced QMout at: DSPL_017_p/QM.out
reading displaced QMout at: DSPL_017_n/QM.out
reading displaced QMout at: DSPL_018_p/QM.out
reading displaced QMout at: DSPL_018_n/QM.out
gammas False

Finished!
LVC parameters written to file: LVC.template
```

The obtained LVC parameters are found in **`LVC.template`**. This file will be needed for setting up the SHARC trajectories.

3.6.4 Modifying the parameters

Removing states or modes from an **`LVC.template`** is straightforward:

```
user@host> $SHARC/modify_LVC_template.py -s "1 0 1" -m "7~15" LVC.template
```

will only keep the lowest singlet and the lowest triplet, and only modes 7 to 15. Using the template from the previous section, this means that the S_1 and modes 16–18 will be removed.

3.7 Running pysharc with NetCDF

Setting up trajectories with **pysharc** is actually very similar to setting up trajectories for regular **sharc.x**. Here, we will use the LVC model produced in section 3.6.

See Section 3.4.2 (p. 46) in the manual.

In order to prepare the trajectory setup, do the following steps:

1. Generate initial conditions using **wigner.py**. It is strongly advisable to use the same **molden** file that was used to generate **V0.txt** for the LVC model.
2. Use **setup_init.py** to prepare vertical excitation calculations. Use the LVC interface (interface number 7).
3. Run the vertical excitation calculations (they should be very fast).
4. Collect vertical excitation information and select initial states using **excite.py**.

At the end of these steps, one should have: **initconds.excited**, **V0.txt**, **LVC.template**. Then, one can the trajectories:

```
user@host> $SHARC/setup_traj.py
```

```
Script for setup of SHARC trajectories started...


  ================================================================================
||                                                                              ||
||                     Setup trajectories for SHARC dynamics                    ||
||                                                                              ||
||          Authors: Sebastian Mai, Philipp Marquetand, Severin Polonius        ||
||                                                                              ||
||                                  Version: 4.0                                ||
||                                 Date: 01.09.24                               ||
||                                                                              ||
  ================================================================================


This script automatizes the setup of the input files for SHARC dynamics.


  ================================================================================
||                               Initial conditions                             ||
  ================================================================================



This script reads the initial conditions (geometries, velocities, initial excited state)
from the initconds.excited files as provided by excite.py.

Please enter the filename of the initial conditions file.
Initial conditions filename: [initconds.excited] (autocomplete enabled) ../initconds.excited

File ../initconds.excited contains 20 initial conditions.
Number of atoms is 6
Reference energy 0.0000000000 a.u.
Excited states are in MCH representation.


Please enter the number of states as a list of integers
e.g. 3 0 3 for three singlets, zero doublets and three triplets.
Number of states: [2 0 1] <ENTER>
```

```
Please enter the molecular charge for each chosen multiplicity
e.g. 0 +1 0 for neutral singlets and triplets and cationic doublets.
Molecular charges per multiplicity: [0 1 0] <ENTER>
Number of states: [2 0 1]
Total number of states: 13

Do you want all states to be active? [True] <ENTER>

Do you want to see the content of the initconds file? [False] yes
Number of initial conditions in file:          20
Contents of the initconds file:

Legend:
?       Geometry and Velocity
.       not selected
#       selected

State 1:
             10         20
              |          |
 0 | .......... ??????????
State 2:
             10         20
              |          |
 0 | ..#....#.# ??????????
State 3:
             10         20
              |          |
 0 | .......... ??????????
State 4:
             10         20
              |          |
 0 | .......... ??????????
State 5:
             10         20
              |          |
 0 | .......... ??????????
Number of excited states and selections:
State    #InitCalc       #Selected
    1           10               0
    2           10               3
    3           10               0
    4           10               0
    5           10               0


Please enter a list specifying for which excited states trajectories should be set-up
e.g. 3 4 5 to select states 3, 4, and 5.
States to setup the dynamics: [2] (range comprehension enabled) <ENTER>

There can be 3 trajectories set up.

Please enter the index of the first initial condition in the initconds file to be setup.
Starting index: [1] <ENTER>

There can be 3 trajectories set up, starting in 1 states.

Please enter the total number of trajectories to setup.
Number of trajectories: [3] <ENTER>
```

```
Please enter a random number generator seed (type "!" to initialize the RNG from the system time).
RNG Seed:  [!] 1234


  ================================================================================
||                     Choose the quantum chemistry interface                     ||
  ================================================================================


Loading interface collection from $SHARC ...

-----------Choose the quantum chemistry interface-----------

Please specify the quantum chemistry interface (enter any of the following numbers):
  1 SHARC_ADAPTIVE          HYBRID interface for adaptive sampling
  2 SHARC_AMS_ADF                 (Not Available! Use SHARC_LEGACY to work with this interface)
  3 SHARC_ANALYTICAL          FAST interface for analytical model Hamiltonians with sympy
  4 SHARC_ASE _DB           HYBRID interface for saveing data to ASE db
  5 SHARC_BAGEL                   (Not Available! Use SHARC_LEGACY to work with this interface)
  6 SHARC_COLUMBUS                (Not Available! Use SHARC_LEGACY to work with this interface)
  7 SHARC_DO_NOTHING          FAST interface for testing (zero E/grad/NAC/DM, unity overlaps/phases)
  8 SHARC_ECI               HYBRID interface for excitonic HF/CI with multiple fragments
  9 SHARC_FALLBACK          HYBRID interface for calling a fallback interface if primary interface fails
 10 SHARC_GAUSSIAN       AB INITIO interface for GAUSSIAN16 for single-reference methods (CIS, TDDFT)
 11 SHARC_LEGACY          BASIC interface for running legacy interfaces via file I/O (AMS-ADF, BAGEL, ...
 12 SHARC_LVC               FAST interface for linear/quadratic vibronic coupling models (LVC, QVC, ...
 13 SHARC_MNDO           AB INITIO interface for the MNDO program (OM2-MRCI)
 14 SHARC_MOLCAS         AB INITIO interface for OpenMolcas (>v23) for multireference calculations ...
 15 SHARC_MOLPRO                 (Not Available! Use SHARC_LEGACY to work with this interface)
 16 SHARC_MOPACPI        AB INITIO interface for the MOPAC-PI program
 17 SHARC_NUMDIFF          HYBRID interface for numerical derivatives (grad, NACdr, SOCdr, DMdr)
 18 SHARC_NWCHEM         AB INITIO interface for NWChem (TDDFT)
 19 SHARC_OPENMM                 (Not Available!)
 20 SHARC_ORCA           AB INITIO interface for ORCA v5-6 (CIS/TDDFT)
 21 SHARC_PYSCF                  (Not Available! Use SHARC_LEGACY to work with this interface)
 22 SHARC_QMMM             HYBRID interface for QM/MM (electrostatic embedding, link atom scheme)
 23 SHARC_QMOUT              FAST interface for frozen-nuclei dynamics (constant E/SOC/DM, unit ...
 24 SHARC_SPAINN             FAST interface for SPaiNN
 25 SHARC_TINKER                 (Not Available!)
 26 SHARC_TURBOMOLE      AB INITIO interface for TURBOMOLE (RICC2/ADC2)
 27 SHARC_UMBRELLA         HYBRID interface for adding umbrella-sampling-style restraints ...

Interface number: 12

The following interface was selected:
 12 SHARC_LVC               FAST interface for linear/quadratic vibronic coupling models (LVC, QVC, ...


The following features are available from this interface:
{'h', 'dm', 'point_charges', 'soc', 'overlap', 'multipolar_fit', 'grad', 'nacdr', 'phases'}

  ================================================================================
||                       Surface Hopping dynamics settings                        ||
  ================================================================================


----------------Nonadiabatic dynamics method----------------

Please choose the dynamics method you want to employ.
1       Trajectory surface hopping dynamics using single surface potential
```

```
2       Semi-classical Ehrenfest dynamics using self-consistent potential
Method: [1] <ENTER>
----------------------Simulation time---------------------

Please enter the total simulation time.
Simulation time (fs): [1000.0] <ENTER>

Please enter the simulation timestep (0.5 fs recommended).
Simulation timestep (fs): [0.5] <ENTER>

Simulation will have 2001 timesteps.

Please choose the integrator you want to use
1       adaptive timestep Velocity-Verlet integrator
2       fixed timestep Velocity-Verlet integrator
Integrator: [2] <ENTER>

Please enter the number of substeps for propagation (25 recommended).
Nsubsteps: [25] <ENTER>

The trajectories can be prematurely terminated after they run for a certain time in the lowest state.
Do you want to prematurely terminate trajectories? [False] <ENTER>


---------------------Dynamics settings--------------------

Do you want to perform the dynamics in the diagonal representation (SHARC dynamics)
or in the MCH representation (regular TSH/SCP)?
SHARC dynamics? [True] <ENTER>
Do you want to include spin-orbit couplings in the dynamics?

Spin-Orbit calculation? [True] <ENTER>
Will calculate spin-orbit matrix.


Please choose the quantities to describe non-adiabatic effects between the states:
1       DDT     =  < a|d/dt|b >            Hammes-Schiffer-Tully scheme (not available)
2       DDR     =  < a|d/dR|b >            Original Tully scheme
3       ktdc    = sqrt(D2(dV)/dt2/(dV))/2  Curvature Driven TDC scheme
4       overlap = < a(t0)|b(t) >           Local Diabatization scheme

Coupling number: [4] <ENTER>
1       mixed gradients are calculated as linear combination of MCH gradients only
2       mixed gradients are calculated by correction of MCH gradients with non-adiabatic coupling vector
3       mixed gradients are calculated by rescaling of the MCH gradients according to time derivatives
        in diagonal and MCH representations
Gradient mixing scheme: [1] <ENTER>

For SHARC dynamics, the evaluation of the mixed gradients necessitates to calculate
non-adiabatic coupling vectors (Extra computational cost).
Include non-adiabatic couplings in the gradient transformation? [False] yes

During a surface hop, the kinetic energy has to be modified in order to conserve total energy.
There are several options to that:
1       Do not conserve total energy. Hops are never frustrated.
2       Adjust kinetic energy by rescaling the velocity vectors. Often sufficient.
3       Adjust ... along the vibrational velocity vector.
4       Adjust ... along the non-adiabatic coupling vector.
5       Adjust ... along the gradient difference vector.
6       Adjust ... along the projected non-adiabatic coupling vector.
```

```
7       Adjust ... along the effective non-adiabatic coupling vector.
8       Adjust ... along the projected effective non-adiabatic coupling vector.
EkinCorrect: [2] 4

If a surface hop is refused (frustrated) due to insufficient energy,
the velocity can either be left unchanged or reflected:
1       Do not reflect at a frustrated hop.
2       Reflect the full velocity vector.
3       Reflect the vibrational velocity vector.
4       Reflect ... along the non-adiabatic coupling vector.
5       Reflect ... along the gradient difference vector.
6       Reflect ... along the projected non-adiabatic coupling vector.
7       Reflect ... along the effective non-adiabatic coupling vector.
8       Reflect ... along the projected effective non-adiabatic coupling vector.
Reflect frustrated: [1] 4

Please choose a decoherence correction for the diagonal states:
1       No decoherence correction.
2       Energy-based decoherence scheme (Granucci, Persico, Zoccante).
3       Augmented fewest-switching surface hopping (Jain, Alguire, Subotnik).
Decoherence scheme: [2] 3

Please choose a surface hopping scheme for the diagonal states:
1       Surface hops off.
2       Standard SHARC surface hopping probabilities (Mai, Marquetand, Gonzalez).
3       Global flux surface hopping probabilities (Wang, Trivedi, Prezhdo).
Hopping scheme: [2] <ENTER>

Do you want to perform forced hops to the lowest state based on a energy gap criterion?
(Note that this ignores spin multiplicity)
Forced hops to ground state? [False] <ENTER>

Do you want to scale the energies and gradients?
Scaling? [False] <ENTER>

Do you want to damp the dynamics (Kinetic energy is reduced at each timestep by a factor)?
Damping? [False] <ENTER>

Do you want to use an atom mask for velocity rescaling or decoherence?
Atom masking? [False] <ENTER>

---------------Selection of Gradients and NACs--------------

In order to speed up calculations, SHARC is able to select which gradients and NAC vectors
it has to calculate at a certain timestep. The selection is based on the energy difference
between the state under consideration and the classical occupied state.

Select gradients? [False] <ENTER>
Select non-adiabatic couplings? [False] <ENTER>


-----------------Settings for large systems-----------------

Do you want to constrain some bond lengths (via a RATTLE)? [False] <ENTER>
Do you want to use a thermostat? [False] <ENTER>


------------------------Laser file------------------------

Do you want to include a laser field in the simulation? [False] <ENTER>
```

```
  ================================================================================
||                                Interface setup                                 ||
  ================================================================================


================================================================================
||                                                                            ||
||                           LVC interface setup                              ||
||                                                                            ||
================================================================================


Specify path to LVC.template (autocomplete enabled) ../LVC.template
Do you have an LVC.resources file? [False] <ENTER>
Do you want to use the Kabsch algorithm? [True] <ENTER>


  ================================================================================
||                                    PYSHARC                                     ||
  ================================================================================


The chosen interface can be run very efficiently with PYSHARC.
PYSHARC runs the SHARC dynamics directly within Python (with C and Fortran extension)
with minimal file I/O for maximum performance.
Setup for PYSHARC? [True] <ENTER>   # activate PySHARC

  ================================================================================
||                           Content of output.dat files                          ||
  ================================================================================


SHARC or PYSHARC can produce output in ASCII format (all features supported currently)
or in NetCDF format (more efficient file I/O, some features currently not supported).
Write output in NetCDF format? [True] <ENTER>   # use efficient output
Write nuclear and electronic date to separate NetCDF files? [False] <ENTER>

Do you want to write the gradients to the output.dat file ?
Write gradients? [False] <ENTER>

Do you want to write the non-adiabatic couplings (NACs) to the output.dat file ?
Write NACs? [False] <ENTER>

Do you want to write property matrices to the output.dat file  (e.g., Dyson norms)?
Write property matrices? [False] <ENTER>

Do you want to write property vectors to the output.dat file  (e.g., TheoDORE results)?
Write property vectors? [False] <ENTER>

Do you want to write the overlap matrix to the output.dat file ?
Write overlap matrix? [True] <ENTER>

Do you want to modify the output.dat writing stride?
Modify stride? [False] <ENTER>


  ================================================================================
||                                Run mode setup                                  ||
  ================================================================================
```

```
------------------------Run script------------------------

This script can generate the run scripts for each initial condition in two modes:

  - In mode 1, the calculation is run in subdirectories of the current directory.

  - In mode 2, the input files are transferred to another directory (e.g. a local scratch directory),
  the calculation is run there, results are copied back and the temporary directory is deleted.
  Note that this temporary directory is not the same as the "scratchdir" employed by the interfaces.

Note that in any case this script will create the input subdirectories in the current working directory.

In case of mode 1, the calculations will be run in:
/user/mai/Documents/NewSHARC/SHARC_4.0/TUTORIAL/LVC/run/traj

Use mode 1 (i.e., calculate here)? [True] <ENTER>

----------------------Submission script---------------------

During the setup, a script for running all initial conditions sequentially in batch mode is generated.
Additionally, a queue submission script can be generated for all initial conditions.

Generate submission script? [False] <ENTER>


#########################Full input##########################

select_directly          True
ninit                    20
natom                    6
repr                     MCH
diag                     False
eref                     0.0
eharm                    0.0
initf                    <_io.TextIOWrapper name='../initconds.excited' mode='r' encoding='UTF-8'>
states                   [2, 0, 1]
nstates                  5
charge                   [0, 1, 0]
statemap                 1: [1, 1, 0.0], 2: [1, 2, 0.0], 3: [3, 1, -1.0], 4: [3, 1, 0.0], 5: [3, 1, 1.0]
actstates                [2, 0, 1]
isactive                 [True, True, True, True, True]
show_content             True
n_issel                  [0, 3, 0, 0, 0]
setupstates              2
firstindex               1
ntraj                    3
needed_requests          'h', 'nacdr', 'grad', 'overlap', 'soc', 'dm'
method                   tsh
tmax                     1000.0
dtstep                   0.5
integrator               2
nsubstep                 25
kill                     False
surf                     diagonal
soc                      True
coupling                 4
phases_from_interface    False
gradcorrect              1
```

```
ekincorrect               4
reflect                   4
decoherence               ['afssh', '']
hopping                   sharc
force_hops                False
force_hops_dE             9999.0
scaling_for_sharc         False
damping                   False
atommaskarray             None
sel_g                     False
sel_t                     False
rattle                    False
use_thermostat            False
laser                     False
dipolegrad                False
ion                       False
pysharc                   True
netcdf                    True
netcdf_separate           False
write_grad                False
write_NAC                 False
write_property2d          False
write_property1d          False
write_overlap             True
stride                    [1]
log.infolevel             2
cwd                       /user/mai/Documents/NewSHARC/SHARC_4.0/TUTORIAL/LVC/run/traj
here                      True
copydir                   /user/mai/Documents/NewSHARC/SHARC_4.0/TUTORIAL/LVC/run/traj
qsub                      False

Do you want to setup the specified calculations? [True] <ENTER>


  ================================================================================
||                           Setting up directories...                          ||
  ================================================================================


Progress: [==================================================] 100%

3 trajectories setup, last initial condition was 10 in state 2.
```

Note that in this example, we have used some more advanced surface hopping settings, specifically rescaling along the nonadiabatic coupling vector, reflection at frustrated hops, and AFSSH decoherence. These options can be used without limitations in LVC, because nonadiabatic coupling vectors can be easily computed for LVC models. Note also that we used a simulation time of 1000 fs to demonstrate the high computational efficiency of the method.

The output of **`setup_traj.py`** is a typical directory structure for SHARC trajectories. In order to run one of them, change into a trajectory directory and run it:

```
user@host> cd Singlet_1/TRAJ_00003
user@host> sh run.sh
```

Note that the **`run.sh`** script automatically sources **`sharcvars.sh`**, so that all necessary paths and libraries are set, and starts **`pysharc_lvc.py`** instead of **`sharc.x`**. The trajectory should be finished within about a minute (ten to hundred steps per second, mostly dependent on the speed of the file system).

The output of the trajectory is very similar to a normal trajectory, with the following differences:

- **restart.traj** and **restart.ctrl** are only written when the trajectory is finished, not during the run,
- **output.xyz** is empty,
- **output.dat** only contains the file header, but no data,
- **output.dat.nc** is present (a binary NetCDF file containing the output data).

To extract the trajectory data, run:

```
user@host> $SHARC/data_extractor_NetCDF.x output.dat
```

Note that it might be necessary to source **sharcvars.sh** prior to running the **data_extractor_NetCDF.x**. In order to generate the **output.xyz** file, run **data_extractor_NetCDF.x** with the **-xyz** flag.

After the extraction step, the trajectory can be analyzed as usual (**populations.py**, **geo.py**, **data_collector.py**, ...). Note that **diagnostics.py** automatically chooses the correct data extractor for the trajectories, so it can also be used.

3.8 Setup, run, and analyze QM/MM trajectories for 3D solvent distributions

In this specialized tutorial, we go through all steps of running a QM/MM project in SHARC, from the generation of the Amber input files to the analysis of the solvent distribution. This tutorial assumes that you have installed Ambertools (and possibly full Amber). The tutorial's goal is to compute the solvent distribution and their time dependence around CH_2S in aqueous solution.

3.8.1 Prepare input files for system setup

Create a folder for the tutorial and a subfolder for the preparation of the Amber input files.

```
user@host> mkdir tutorial_qmmm
user@host> cd tutorial_qmmm
user@host> mkdir leap
user@host> cd leap
```

Create a **geom.xyz** file with the following content:

```
4

C   0.0000    -1.0330    -0.0000
S   0.0000     0.7910     0.0000
H  -0.9370    -1.6310    -0.0000
H   0.9370    -1.6310    -0.0000
```

Note that we intentionally use a long C–S bond length here to avoid problems in the detection of the atom types, so that the tutorial runs smoothly.

Next, the XYZ file needs to be translated into the PDB format that Amber can understand. You can use

```
user@host> openbabel -ixyz geom.xyz -opdb geom.pdb
```

or use some Online Converter based on OpenBabel to create the **geom.pdb** file.

Using the PDB file, you can run the **antechamber** module of Ambertools to identify atom types and compute partial charges:

```
user@host> antechamber -i geom.pdb -fi pdb -o thioformaldehyde.mol2 -fo mol2 -nc +0 -c bcc -rn mol
```

This uses a semi-empirical electronic structure method to compute the partial charges. For realistic projects, it is usually better to use RESP charges based on some ab initio method. Please see various Amber tutorials for this. Note that the **antechamber** program produces the **thioformaldehyde.mol2** file.

We also need to make sure that all needed force field parameters are present. This can be done with the **parmchk2** tool:

```
user@host> parmchk2 -i thioformaldehyde.mol2 -o thioformaldehyde.frcmod -f mol2
```

This produces the **thioformaldehyde.frcmod** file.

Now we have the **thioformaldehyde.mol2** and **thioformaldehyde.frcmod** and can proceed to build the system. Note that for realistic projects, preparing the force field parameters can be significantly more involved. However, later in this tutorial, we will show that trajectories can be re-equilibrated with SHARC. If such a SHARC-based re-equilibration is performed for sufficiently long time scales, the exact details of the Amber dynamics become irrelevant.

3.8.2 Build system and get topology file

Stay in the **leap** folder.

Prepare the file **tleap.in** with the following content:

```
source leaprc.water.tip3p
source leaprc.gaff2
mol = loadmol2 thioformaldehyde.mol2
solvateoct mol TIP3PBOX 15.
loadamberparams thioformaldehyde.frcmod
saveamberparm mol system.prmtop system.inpcrd
quit
```

and run

```
user@host> tleap -f tleap.in
```

This produces **system.inpcrd** and **system.prmtop**. The former file contains the coordinates of thioformaldehyde plus a 15 Ångstrom box of water and the latter file contains the all static properties of the system (topology, atom types, force field parameters, masses, etc). The system can be inspected with VMD.

3.8.3 Run Amber dynamics

In the next step, we run an Amber trajectory. The goals are (i) to clean up the water box to avoid empty cavities or collisions with the solute, (ii), to heat up the system to room temperature, (iii) to pressurize the system to 1 bar, (iv) to run a long production trajectory to sample geometries, and (v) to truncate the box of water molecules to something that is more convenient for this tutorial.

Change to another folder:

```
user@host> cd ..
user@host> mkdir amber
user@host> cd amber
user@host> cp ../leap/system.* .
```

and copy the Amber driver script:

```
user@host> cp $SHARC/../examples/Amber_MD/run_Amber1-5.sh .
```

This script runs the steps (i) to (v) using one of Amber's dynamics drivers plus its analysis tool **cpptraj**.

Inspect the driver script. Near the top you find the section with user-adjustable settings:

```
PMEMD_CUDA=$AMBERHOME/bin/pmemd.cuda_SPFP
PMEMD_SINGLE="$AMBERHOME/bin/pmemd"
PMEMD_MPI="mpirun -np $SLURM_NTASKS_PER_NODE $AMBERHOME/bin/pmemd.MPI"
DRIVER=$PMEMD_CUDA          # select your driver
RUN="true"

# -------------------------------------------------------------

DT=0.0002           # time step in ps, 2fs is ok when using SHAKE
NSTEPS_OPTI=1000    # 1000 steps to clean up box
```

```
NSTEPS_HEAT=25000   # 50ps heating
NSTEPS_EQUI=50000   # 100ps pressurizing
NSTEPS_PROD=250000  # 500ps production
SNAPSHOTS=100       # make 100 snapshots (one every 5ps)
NSTEPS_SNAPSHOTS=$(echo "$NSTEPS_PROD/$SNAPSHOTS" | bc)
NTF=2               # 1: all bonds, 2: omit X-H bonds (SHAKE)
JFASTW=0            # 4: flexible, 0: SHAKE
TEMP=300            # in K

SOLVENTS_TO_KEEP=100 # number of waters to keep
SOLVENT_NAME="WAT"   # name of the solvent molecule
```

Please set **DRIVER** to one of Amber's drivers that you have available. The various **PMEMD** drivers are only available for the full version of Amber, but you can use **sander** instead.

The tutorial assumes that you do not change the options below the line.

At the end of the driver script, there is a line **trajout $NEWBASE.$ext restartnc** which controls the output format of the files that get produced. In small projects, you can change this line to **trajout $NEWBASE.$ext rst7** to directly produce ASCII restart files. For large projects, the **restartnc** output should be chosen, even though it is a bit more complicated to handle in subsequent steps. The handling will be described below.

Run the script with

```
user@host> sh run_Amber1-5.sh
```

or submit it to a batch queuing system.

After the job is successful, the following output files will be produced, one for each step: **01_min.out**, **02_heat.out**, **03_equi.out**, **04_prod.out**, and **05_cppt.out**. In **04_prod.out**, you can check whether the density, temperature, and volume is constant. Grep for "TEMP", "VOLUME", and "Density". If you followed the instructions, they should be converged. Note that the pressure will be extremely noisy, because the system is relatively small. Oscillations of several hundred bar are normal here.

Besides the output files, the script will produce 100 files called **05_cppt.1** to **05_cppt.100**. Depending on the chosen output format, these are either ASCII or binary files in NetCDF format with Amber conventions. They contain the desired initial positions and velocities for the initial conditions. Note that they were truncated to 100 water molecules in step (v). Also note that step (v) produced a new prmtop file, because the system was truncated to 100 water molecules. This file is called **system_100.prmtop** and is needed in subsequent steps.

3.8.4 Prepare initial conditions

If you have chosen ASCII restart files in the previous step, these can now be transformed into a **initconds** file, which can be used as any other one in SHARC. The file can be prepared with (this command will not work if you have NetCDF restart files):

```
user@host> $SHARC/amber_to_initconds.py -t 2. -x system_100.prmtop 05_cppt.1 05_cppt.*
```

In this way, the first snapshot is saved as equilibrium geometry and also as first initial condition. This will also produce a **initconds.xyz**. Note that the **initconds** file can be very large for QM/MM projects, so this approach using ASCII restart files and **amber_to_initconds.py** is not very convenient.

See Section 7.4 (p. 183) in the manual.

For large projects, **amber_to_initconds.py** will run very long and the created **initconds** file will be many GB large, too large to handle conveniently. Therefore, it is more convenient to work with a dummy **initconds** file. This approach is compatible with NetCDF restart files and will be used within this specialized tutorial from here on.

First create an **initconds** file with one dummy atom and the same number of initial conditions as created with Amber:

```
user@host> $SHARC/wigner.py -n 100 --dummy_molecule
```

See Section 7.1 (p. 177) in the manual.

The **initconds** file only serves to store the information about the electronic states. The coordinates and velocities remain within the NetCDF restart files for now. The NetCDF restart files will be used to create the needed **QM.in** files or **geom** and **veloc** files directly within the initial conditions or trajectory folders.

3.8.5 Prepare template files for QM/MM

In the next step, we need to collect all input files needed to set up SHARC trajectories with QM/MM.

Go up one folder:

```
user@host> cd ..
```

First, we will create the QM/MM table file, the topology files for the full system and the QM subsystem, and some files for the SHARC driver. This can be conveniently done from the prmtop file:

```
user@host> $SHARC/setup_from_prmtop.py -f system_100.prmtop -q "1~4" --rattle-hx --atommask
```

See Section 7.16 (p. 200) in the manual.

Here, **1~4** indicates that the QM subsystem is comprosed of atoms 1–4. The output will be:

```
rattling all H-X bonds ...
writing file 'rattle'
making atom mask file ... (atom mask is '1~4')
writing file 'atommask'
writing file 'QMMM.table'
setting QM charges to zero ...
writing file 'system_100_chrg_0.prmtop'
truncating prmtop for QM-only ...
writing file 'system_100_qm_and_links_chrg0.prmtop'
```

This creates five files. **atommask** contains a list of **T** and **F** that specify which atoms participate in rescaling and decoherence. By default, only QM atoms are included here. The file **rattle** contains a list of the atom indices of all X–H bonds and their equilibrium bond lengths. The **QMMM.table** file contains the specification whether each atom is in the QM or MM region, its element, and the atom indices of its bonding partners. This is one input file for **SHARC_QMMM.py**. The files **system_100_chrg_0.prmtop** and **system_100_qm_and_links_chrg0.prmtop** are input files for the **SHARC_OPENMM.py** interface. Here, the file **system_100_chrg_0.prmtop** is intended for the **MML** child of **SHARC_QMMM.py**—it contains the topology and force field for the entire system, but with the charges of the QM region set to zero. The file **system_100_qm_and_links_chrg0.prmtop** is intended for the **MMS** child of **SHARC_QMMM.py**—it contains the topology and force field for only the QM region and possible link atom caps, and with the charges of all atoms set to zero.

See Section 6.30 (p. 147) in the manual.

We also need to create the template files for the QM/MM and OpenMM interfaces. These files are very simple and can be set up manually. One file is **OPENMM_MML.template**:

```
prmtop system_100_chrg_0.prmtop
```

See Section 6.7 (p. 92) in the manual.

and one file is **OPENMM_MMS.template**:

```
prmtop system_100_qm_and_links_chrg0.prmtop
```

You also need to prepare the **QMMM.template**, which should look like this:

See Section 6.30 (p. 147) in the manual.

```
qmmm_table QMMM.table
qm-program LVC
mm-program openmm
embedding subtractive
mms-dir ./MMS
mml-dir ./MML
qm-dir ./LVC
mm_dipole False
```

3.8.6 Prepare template file for the QM region

Now the only input file that is still missing is the template file for the QM region. Currently, QM/MM simulations can be done with any of **SHARC_GAUSSIAN.py**, **SHARC_ORCA.py**, **SHARC_TURBOMOLE.py**, **SHARC_MOLCAS.py**, and **SHARC_LVC.py**. For demonstration purposes, we use the LVC interface here, which involves an additional parametrization step, using OpenMolcas. The preparation with the ab initio interfaces is quicker, but the trajectories are orders of magnitude more expensive. The LVC parametrization is similar to the other example above.

See Section 6.4.4 (p. 88) in the manual.

Go into another folder:

```
user@host> cd lvc_param
```

First, a frequency calculation is needed. Prepare the **OpenMolcas** input with **molcas_input.py**, similar to Section 2.2, using CASSCF(6,4)/cc-pVDZ, 2 singlet states, and optimizing the ground state. Use the resulting Molden file to get a normal mode file (**V0.txt**):

See Section 7.1 (p. 177) in the manual.

```
user@host> $SHARC/wigner.py -l MOLCAS.freq.molden
```

Also use **molcas_input.py** to create a **MOLCAS.template**, which should look like this:

See Section 6.15.4 (p. 109) in the manual.

```
basis cc-pvdz
ras2 4
nactel 6
inactive 9
roots 2
cholesky
method CASSCF
```

Subsequently, use **setup_LVCparam.py** to set up the calculation needed for the parametrization. Use two states, and analytical kappas and lambdas. Very importantly, request the atomwise multipolar density representation, which is needed for LVC/MM. Use the **MOLCAS.RasOrb** file from the frequency calculation. With analytical kappas and lambdas, only one calculation is needed, so go into **DSPL_RESULTS/DSPL_000_eq** and run the calculation. Then change back to **DSPL_RESULTS/** and run

```
user@host> $SHARC/create_LVCparam.py
```

to obtain an **LVC.template** file. This file contains the multipolar charge parameters for the two singlet states.

3.8.7 Ground state reequilibration

In the example, there will be anisotropic hydrogen bonds around the CH_2S molecule. These cannot be reproduced with an Amber force field with only point charges, so we need to reequilibrate in the ground state using SHARC.

To set up trajectories in the ground state, we need to flag all 100 initial conditions in the **initconds** file for starting in the first state. Use **excite.py** with the options to (i) load the **initconds** file, (ii) generate a list of dummy states (2 singlets), (iii) use the MCH representation and set the reference energy to zero, (iv) provide a list of desired initial states, (v) choose the first state. This will create a **initconds.excited** file.

At this point, you should have the following ten files: **initconds.excited** (from using **excite.py**), **LVC.template** (from the parametrization), **OPENMM_MML.template**, **OPENMM_MMS.template**, **QMMM.template** (all three created manually), **QMMM.table**, **system_100_chrg_0.prmtop**, **system_100_qm_and_links_chrg0.prmtop**, **rattle**, and **atommask** (all five created with **setup_from_prmtop.py**).

Make a new folder:

```
user@host> mkdir traj_gs
user@host> cd traj_gs
```

and copy the ten input files into it.

Then launch **setup_traj.py**. Use the following options:

See Section 7.18 (p. 201) in the manual.

- Load **initconds.excited**,
- Use 2 states of charge 0,
- Setup all 100 trajectories,
- Select the DROPLET interface,
- Use "QMMM" as the child interface of the DROPLET interface,
- Provide the path to the **QMMM.template** file,
- Select the following dynamics options: TSH, 1ps, 2fs, fixed, any substeps, no early termination, overlaps, defaults for TSH options (do not matter in S_0), atom masking for atoms **1~4**
- Use RATTLE with the **rattle** file from before,
- Use Langevin thermostat with desired temperature (298.15 K), a second RNG seed, and default coupling constant,
- No laser,
- No **DRPOLET.resource** file,
- Define two droplet/tether potentials: the first computed from system size (use default density of water, pressure, potential shape parameter, and molar mass, with 101 molecules, origin at 0,0,0, affect all atoms), and the second with manual parameters (a 4 Å off-set, force constant 10^{-5} Hartree/Bohr^2, tether at the origin at 0,0,0, and affecting atoms **1~4**),
- No **QMMM.resource** file,
- Path to the previously generated **LVC.template** file, no **LVC.resources** file, but use Kabsch algorithm,
- In the next section, observe that it starts with **Setting up MML-interface (whole system)**, and correspondingly provide the path to the **OPENMM_MML.template** file,
- No **OPENMM.resources** file needed,
- For the section **Setting up MMS-interface (qm system)**, provide the path to the **OPENMM_MMS.template** file,
- No **OPENMM.resources** file needed either,
- Set up for PySHARC, enable fast queue,
- Write output in NetCDF format, no need to use separate output files, modify stride (every 100th step by simply entering **100**, then **500 100**, then **500 100**),
- Use mode 1, submit script as you like, link interface files

This will set up the entire set of trajectories in the ground state. Since we have used a dummy **initconds** file, the **geom** and **veloc** files will contain only one atom. These two files now have to be replaced by the actual

geom and **veloc** files based on the Amber restart files. This is achieved by **restartnc_to_xyz.py**. Go into the first **TRAJ** folder and type:

See Section 7.7 (p. 186) in the manual.

```
user@host> $SHARC/restartnc_to_xyz.py /path/to/system_100.prmtop /path/to/05_cppt.1 -t 2. -g
```

Note that you can optionally use the option **-r "1~4"** to reorder the coordinates of the water molecules by distance from the center of mass of the solute.

To do the same for all 100 trajectories, within the **Singlet_0** folder do (you can type everything without linebreaks):

```
user@host> for i in {1..100}; do
j=$(printf "%05d" "$i"); cd TRAJ_$j;
$SHARC/restartnc_to_xyz.py /path/to/system_100.prmtop /path/to/05_cppt.$i -t 2. -g;
cd -; done
```

You also have to replace the **atommask** files in each trajectory folder, because when **setup_traj.py** wrote them automatically, it assumed only 1 atom, instead of 304 atoms:

```
user@host> for i in TRAJ_00*; do cp ../../atommask $i; done
```

You can then run all 100 trajectories using their **run.sh** scripts. You might need to adjust the run scripts to your local cluster architecture and queuing system. Note that the SHARC driver command should be **driver.py -i QMMM**, because the QM/MM interface is the top-level interface.

3.8.8 Ground state solvent distribution preparation

Obtaining the ground state solvent distribution involves a few steps. First, we need to collect the coordinates of a particular time step from all trajectories, and align the solute molecule.

Within the **Singlet_0** folder, start

See Section 7.38 (p. 231) in the manual.

```
user@host> $SHARC/align_and_reorder_traj.py
```

and use these options:

- This directory **.** (containing all 100 **TRAJ_XXXXX** folders),
- Use **output.dat.nc**,
- Align yes, use reference geometry **geom.xyz** that was used in the beginning with OpenBabel/Antechamber in Section 3.8.1,
- Use atoms **1~4**,
- Write coordinates.

This creates files **frame_coord_mol_pers_0000X.nc** with **X** from 0 to 5, because in **setup_traj.py** we selected to write output every 100 steps and there are 500 steps in total. In particular, file **frame_coord_mol_pers_00005.nc** contains the 100 snapshots at the end of the 1000 fs trajectories.

Now we prepare a mask file for all atoms in the system. Start

```
user@host> cpptraj
```

and use the following commands:

```
parm system_100.prmtop
trajin frame_coord_mol_pers_00005.nc 1 1
mask @%c maskout "mask_c"
mask @%s maskout "mask_s"
mask @%h4 maskout "mask_h1"
mask @%HW maskout "mask_h"
mask @%OW maskout "mask_o"
```

```
run
quit
```

This produces files containing the indices of all atoms, separated by QM and MM region and element.

3.8.9 Ground state solvent distribution: RDFs

Now we can first get a radial distribution function, or RDFs, as (type without linebreak):

```
user@host> $SHARC/frames_to_RDF.py -w 0.1 -n 200 frame_coord_mol_pers_00005.nc
           mask_s mask_h RDF_s_h.txt
```

See Section 7.39 (p. 233) in the manual.

The RDF can be visualized with **gnuplot** or other software. The first column is the distance, the second column is the total RDF (not normalized with respect to droplet volume). The three further columns are RDFs weighted by Cartesian distance, the sum of the three columns is the total RDF. Note that the set of trajectories in this tutorial is too limited (too few trajectories, too short simulation time, too few water molecules surrounding the solute) to observe decent RDFs.

The previous command only produced the RDF for the file **frame_coord_mol_pers_00005.nc**. You can produce all RDFs with a Bash loop:

```
user@host> for i in {00000..00005}; do $SHARC/frames_to_RDF.py
           -w 0.1 -n 200 frame_coord_mol_pers_${i}.nc  mask_s mask_h
           RDF_s_h_${i}.txt; done
```

To plot the time-dependent RDFs, you can combine all RDF files into one large file:

```
user@host> $SHARC/data_merger.py RDF_s_h_%05i.txt
           --input-file ../TRAJ_00001/input --output RDFs_all.txt --ref_av_n 1
```

See Section 7.42 (p. 236) in the manual.

Here, we provide the path to one of the trajectory input files in order to extract the time step and the output stride automatically. If the input file is not provided, the time step should be explicitly given. With the command, we also take the first RDF and use it as reference. The resulting output file **RDFs_all.txt** has four columns: time, distance, value of the RDF at that time and distance, and value of the reference RDF at the same distance. One can use **gnuplot** to easily plot the time-dependent RDF as:

```
gnuplot> p "RDFs_all.txt" u 2:3:1 w l pal
```

or plot the time-dependent difference RDF as:

```
gnuplot> set view map
gnuplot> sp "RDFs_all.txt" u 2:1:($3-$4) w pm3d
```

Note that using **RDF_to_scattering.py**, one can convert the RDFs to X-ray scattering results. For a full set of scattering results, you need to prepare mask files for each element, then compute RDFs for each element pair, then transform all RDFs to scattering signals. The resulting scattering files can also be merged with **data_merger.py**. To combine the merged data for all the element pairs, you can join the files, e.g., with **paste -d**.

See Section 7.41 (p. 235) in the manual.

3.8.10 Ground state solvent distribution: 3D-SDFs

Now we get the distribution (type without linebreak):

```
user@host> $SHARC/frames_to_dx.py -w 1. -f 1. -n 20 frame_coord_mol_pers_00005.nc
           mask_h frame_5_h.dx
```

See Section 7.40 (p. 234) in the manual.

This creates **frame_5_h.dx**, containing a distribution of the water hydrogens at a resolution of 1 Angstrom. Repeat the last command for oxygen using **mask_o frame_5_o.dx**. The distributions can be visualized with VMD, although the set of trajectories in this tutorial is too limited to see the actual shape of the distribution.

When visualising with VMD, you need to load the xyz file and then the dx file. For the dx file, use the Isosurface representation. To observe the distribution, the isovalue needs to be chosen correctly. Water at standard conditions has about 0.0336 water molecules per cubic Angstrom. Our grid cell volume is one cubic Angstrom. We have 100 snapshots, so we would expect 3.36 oxygen atoms per grid cell, and 6.72 hydrogen atoms. For the example figure, we use a relative isovalue of 1.5, so in VMD we use an isovalue of 5.04 for oxygen and 10.08 for hydrogen. The result is shown in Figure 3.2. With more trajectories and more water molecules, at some point one can see that there are two hydrogen bonds in the molecular plane left and right of the sulfur, as in our previous publication in Figure 2.

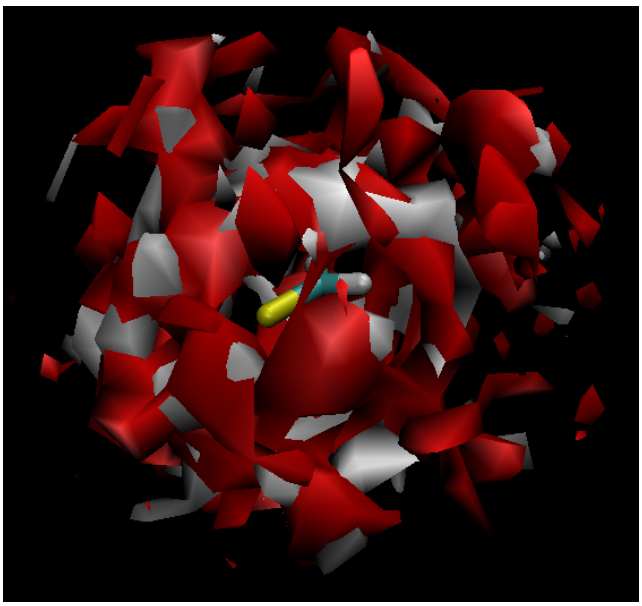

Figure 3.2: Solvent distribution around CH_2S. Oxygen in red, hydrogen in white.

3.8.11 Excitation and running in the excited state

Next, we want to excite the molecule at the end of the ground state equilibration trajectories. Create a new folder:

```
user@host> mkdir init_from_gs
```

Within this folder, use **setup_init.py** to set up the 101 initial condition calculations based on the previously used **initconds** file. During **setup_init.py**, set 2 singlet states and use the QM/MM interface exactly as above.

Because the **initconds** file contains only one dummy atom, we need to replace the **QM.in** files with the actual geometries. This time, we use the coordinates from the last time step of the ground state trajectories. For **ICOND_00000**, use:

```
user@host> $SHARC/sharctraj_to_xyz.py  /path/to/TRAJ_00001/geom
           /path/to/TRAJ_00001/output.dat.nc -q -s -1 > QM.in2
user@host> mv QM.in2 QM.in
```

See Section 7.8 (p. 187) in the manual.

Note that we use the same coordinates for **ICOND_00001** as for **ICOND_00001**. Also here, you can use the option **-r "1~4"** to sort the waters by distance.

For the other 100 folders, use:

```
user@host> for i in {1..100}; do
j=$(printf "%05d" "$i"); cd ICOND_$j;
```

```
$SHARC/sharctraj_to_xyz.py /path/to/TRAJ_$j/geom /path/to/TRAJ_$j/output.dat.nc -q -s -1 > QM.in2;
mv QM.in2 QM.in; cd -; done
```

This replaces the **QM.in** files containing the dummy molecule with the geometries from the last time step (**-s -1**) of the ground state trajectories.

Subsequently, run the calculations in all **ICOND** folders. In each folder, simply execute:

```
user@host> $SHARC/SHARC_QMMM.py QM.in &> QM.log
```

You can use **all_run_init.sh** to do all 100 calculations sequentially.

Then start

```
user@host> $SHARC/excite.py
```

Use the previous **initconds** file, read excited-state information, use the present directory (containing the 101 **ICOND** folders), use the MCH representation, and perform a delta-pulse selection between 2 and 4 eV, using the S_0 as initial state (use any RNG seed). The selection scheme should provide some excited trajectories, for example

```
Number of initial conditions excited:
State   Selected   InRange   Total
    1          0         0     100
    2         17       100     100
```

Using the **initconds.excited** file from just now, set up new trajectories using **setup_traj.py** in a new folder. Use the same settings as above, although the output stride could be made smaller to see the dynamics on a finer time scale. After the setup script is finished, use **sharctraj_to_xyz.py -g -s -1** with the ground state trajectories to update the **geom** and **veloc** files. Do not forget to copy the **atommask** files. Then, the excited-state trajectories can be executed.

As discussed above, this tutorial is too small to observe the solvent distribution dynamics. As shown in our previous publication in Figure 4, using a sufficient number of trajectories allows to simulate a switch in the positions of the hydrogen bonds in the excited state.

3.9 Frozen-nuclei dynamics

Within SHARC, users can run frozen-nuclei dynamics by using the so-called SHARC–QMout interface. Such dynamics is essentially a simple propagation of the electronic wave function at fixed nuclear positions. It might be useful to identify processes that occur even without nuclear motion, like intersystem crossing in complexes of heavy metals.

See Section 6.2 (p. 82) in the manual.

To perform frozen-nuclei dynamics, you have to first obtain initial conditions. In this quick tutorial, we take the SO_2 molecule and perform an optimization and frequency calculation with ORCA.

See Section 3.3 (p. 38) in the manual.

```
! BP86 def2-svp OPT FREQ

* xyz 0 1
  S      0.000000   -0.000000   -0.037458
  O      0.000000    1.276829    0.718729
  O      0.000000   -1.276829    0.718729
*
```

Load the necessary ORCA environment, then run

```
user@host> orca orca.inp > orca.log
```

While the calculation is running, prepare the resources and template files for the ORCA interface, which we will use to generate the electronic structure information at the frozen geometries.

The **ORCA.template** looks like

```
basis def2-svp
functional BP86
```

and the **ORCA.resources** like (replace orcadir and/or scratchdir to existing paths on your machine)

```
orcadir $ORCADIR
scratchdir $TMPDIR
```

Alternatively, the resources can be entered during the input dialogue of the setup script.

After the ORCA calculation is finished, we invoke **ORCA_hess_freq.py**, which generates the **orca.hess.molden** file, which contains the normal mode coordinates:

```
user@host> $SHARC/ORCA_hess_freq.py orca.hess
```

ORCA_hess_freq.py is a new script in SHARC4 that extracts the normal modes in a more robust way from ORCA (and it even works with explicit symmetry in ORCA6). It produces a **orca.hess.molden** file.

Generate the file **initconds**, which contains the initial conditions with the geometries where we want to run the frozen-nuclei dynamics:

```
user@host> $SHARC/wigner.py -n 50 orca.hess.molden
```

In a new directory, set up the initial computations, which will provide energies, transition dipole moments, and spin–orbit couplings at the sampled geometries:

```
user@host> $SHARC/setup_init.py
```

using

```
  ================================================================================
||                                                                              ||
||                  Setup initial conditions for SHARC dynamics                 ||
||                                                                              ||
||                    Authors: Sebastian Mai, Severin Polonius                  ||
||                                                                              ||
||                                 Version: 4.0                                 ||
||                                Date: 01.04.25                                ||
||                                                                              ||
  ================================================================================

This script automatizes the setup of excited-state calculations for initial conditions
for SHARC dynamics.

------------------Initial conditions file------------------

Initial conditions file "initconds" detected. Do you want to use this?
Use file "initconds"? [True] <ENTER>

File "initconds" contains 50 initial conditions.
Number of atoms is 3

----------------Range of initial conditions-----------------

Please enter the range of initial conditions for which an excited-state calculation
should be performed as two integers separated by space.
Initial condition range: [1 50] <ENTER>

Script will use initial conditions 1 to 50 (50 in total).

----------------Number of states and charge-----------------

Please enter the number of states as a list of integers
e.g. 3 0 3 for three singlets, zero doublets and three triplets.
Number of states: 3 0 3

Please enter the molecular charge for each chosen multiplicity
e.g. 0 +1 0 for neutral singlets and triplets and cationic doublets.
Molecular charges per multiplicity: [0] 0 0 0
Number of states: [12] <ENTER>
Total number of states: 12

Loading interface collection from /user/sharc/sharc_main/bin ...

-----------Choose the quantum chemistry interface-----------

Please specify the quantum chemistry interface (enter any of the following numbers):
  1 SHARC_ADAPTIVE          HYBRID interface for adaptive sampling
  2 SHARC_AMS_ADF                 (Not Available! Use SHARC_LEGACY to work with this interface)
  3 SHARC_ANALYTICAL          FAST interface for analytical model Hamiltonians with sympy
  4 SHARC_ASE_DB            HYBRID interface for saveing data to ASE db
  5 SHARC_BAGEL                   (Not Available! Use SHARC_LEGACY to work with this interface)
  6 SHARC_COLUMBUS                (Not Available! Use SHARC_LEGACY to work with this interface)
  7 SHARC_DO_NOTHING          FAST interface for testing (zero E/grad/NAC/DM, unity overlaps/phases)
  8 SHARC_ECI               HYBRID interface for excitonic HF/CI with multiple fragments
  9 SHARC_FALLBACK          HYBRID interface for calling a fallback interface if primary interface fails
 10 SHARC_GAUSSIAN       AB INITIO interface for GAUSSIAN16 for single-reference methods (CIS, TDDFT)
 11 SHARC_LEGACY             BASIC interface for running legacy interfaces via file I/O (AMS-ADF, ...
```

```
 12 SHARC_LVC                FAST interface for linear/quadratic vibronic coupling models (LVC, ...
 13 SHARC_MNDO           AB INITIO interface for the MNDO program (OM2-MRCI)
 14 SHARC_MOLCAS         AB INITIO interface for OpenMolcas (>v23) for multireference calculations ...
 15 SHARC_MOLPRO                   (Not Available! Use SHARC_LEGACY to work with this interface)
 16 SHARC_MOPACPI        AB INITIO interface for the MOPAC-PI program
 17 SHARC_NUMDIFF           HYBRID interface for numerical derivatives (grad, NACdr, SOCdr, DMdr)
 18 SHARC_NWCHEM         AB INITIO interface for NWChem (TDDFT)
 19 SHARC_OPENMM                   (Not Available!)
 20 SHARC_ORCA           AB INITIO interface for ORCA v5-6 (CIS/TDDFT)
 21 SHARC_PYSCF                    (Not Available! Use SHARC_LEGACY to work with this interface)
 22 SHARC_QMMM              HYBRID interface for QM/MM (electrostatic embedding, link atom scheme)
 23 SHARC_QMOUT               FAST interface for frozen-nuclei dynamics (constant E/SOC/DM, unit ...
 24 SHARC_SCHNARC                  (Not Available!)
 25 SHARC_SPAINN              FAST interface for SPaiNN
 26 SHARC_TURBOMOLE      AB INITIO interface for TURBOMOLE (RICC2/ADC2)
 27 SHARC_UMBRELLA          HYBRID interface for adding umbrella-sampling-style restraints (harmonic ...

Interface number: 20

The following interface was selected:
 20 SHARC_ORCA           AB INITIO interface for ORCA v5-6 (CIS/TDDFT)

The following features are available from this interface:
{'grad_pc', 'overlap', 'dm', 'ion', 'h', 'point_charges', 'theodore', 'molden', 'phases', 'grad', 'soc'}

----------------Spin-orbit couplings (SOCs)-----------------

Do you want to compute spin-orbit couplings?

Spin-Orbit calculation? [True] <ENTER>

----------------Overlaps to reference states----------------

Do you want to compute the overlaps between the states at the equilibrium geometry
and the states at the initial condition geometries?
Reference overlaps? [False] <ENTER>

--------------TheoDORE wave function analysis---------------

Do you want to run TheoDORE to obtain one-electron descriptors for the electronic wave functions?
TheoDORE? [False] <ENTER>

================================================================================
||                                                                            ||
||                           ORCA interface setup                             ||
||                                                                            ||
================================================================================

Please specify path to ORCA directory (SHELL variables and ~ can be used,
will be expanded when interface is started).

Path to ORCA: (autocomplete enabled) $ORCADIR

---------------------Scratch directory----------------------

Please specify an appropriate scratch directory. This will be used to run the ORCA calculations.
The scratch directory will be deleted after the calculation. Remember that this script cannot
check whether the path is valid, since you may run the calculations on a different machine.
The path will not be expanded by this script.
Path to scratch directory: (autocomplete enabled) $TMPDIR
```

```
------------------ORCA input template file------------------

Template filename: (autocomplete enabled) ORCA.template

Do you have a 'ORCA.resources' file? [False] yes
Specify the path: [ORCA.resources] (autocomplete enabled) ORCA.resources

  ================================================================================
||                                Run mode setup                                ||
  ================================================================================

------------------------Run script------------------------

This script can generate the run scripts for each initial condition in two modes:
  - In mode 1, the calculation is run in subdirectories of the current directory.
  - In mode 2, the input files are transferred to another directory (e.g. a local scratch directory),
  the calculation is run there, results are copied back and the temporary directory is deleted.
  Note that this temporary directory is not the same as the "scratchdir" employed by the interfaces.

Note that in any case this script will create the input subdirectories in the current working directory.

In case of mode 1, the calculations will be run in:
/user/sharc/sharc_tutorial

Use mode 1 (i.e., calculate here)? [True]  <ENTER>

--------------------Submission script---------------------

During the setup, a script for running all initial conditions sequentially in batch mode is generated.
Additionally, a queue submission script can be generated for all initial conditions.

Generate submission script? [False] <ENTER>

#########################Full input#########################

cwd                       /global/lorenz/sharc_tutorial/prepare
ninit                     50
natom                     3
initf                     <_io.TextIOWrapper name='initconds' mode='r' encoding='UTF-8'>
irange                    [1, 50]
states                    [3, 0, 3]
nstates                   3
charge                    [0, 0, 0]
needed_requests           'h', 'dm'
soc                       True
refov                     False
theodore                  False
here                      True
qsub                      False

Do you want to setup the specified calculations? [True] <ENTER>

Do you want to link the interface files? [False]  <ENTER>

  ================================================================================
||                           Setting up directories...                          ||
  ================================================================================

Progress: [================================================] 100%
```

Run all initial conditions calculations sequentially:

```
user@host> sh all_run_init.sh
```

In case you requested **reference overlap** in the setup, **ICOND_00000** has to be run first, before you start all other initial condition calculations (but note that in the example above, reference overlaps are turned off).

For the excitation of the molecule, we need to generate an excitation laser pulse. In this example, a short Gaussian laser pulse is set up, using a central wave length of 3.5 eV. The laser file is prepared for a 10 fs long SHARC simulation with 0.05 fs step and 1 substep per step (giving 10 fs/0.05 fs*1+1=201 steps).

```
user@host> laser.x
```

with the input:

```
 Number of lasers:
1
           1
 Real-valued field (T) or not (F):
T
 T
 Set starting time, end of time and number of time steps (t0[fs],tEnd[fs],Nt) :
0 10 201          # one step for each substep in sharc.x
  0.000000000000000E+000   10.000000000000             201
 consequently, we have a step size of   5.000000000000000E-002
 Write additional files for debugging (T) or not (F):
F
 F
              (Empty line to increase readability. Press Enter.)
<ENTER>

 Choose polarization vector (e.g. 2.,0.,0. will be normalized):
1 0 0
 1.00000000000000       0.000000000000000E+000  0.000000000000000E+000
 Choose type of envelope (1=Gaussian,2=Sinusoidal):
1
           1
 Choose field strength in (1) [GV/m] (2) [TW/cm^2] (3) [a.u.]:
1
 Enter field strength:
0.1
   1.00000000000000       GV/m
 =   0.132720936399648       TW/cm^2
 =   1.944690381149866E-003  a.u.
 Set FWHM,begin,center,center2 and end time of pulse (1 + 3 affect Gaussian, 2-5
  affect Sinus) [fs]:
4. 0 5. 0 0
   4.0000000000000       0.000000000000000E+000    5.000000000000
  0.000000000000000E+000  0.000000000000000E+000
 Choose central frequency in (1) [nm] (2) [eV] (3) [a.u.]:
2
 Enter central frequency:
3.5   # according to the energy of the bright S1
   3.50000000000000       eV
 =    354.240566952000       nm
 =   0.128622627614792       a.u.

 Choose phase (as multiple of pi):
0
  0.000000000000000E+000
 Choose colored double pulse (b_1[fs]), which is multiplied with abs(omega-omega
```

```
 _0):
0
  0.000000000000000E+000
 Choose linear chirp (b_2[fs^2]):
0
  0.000000000000000E+000
 Choose third-order chirp (b_3[fs^3]):
0
  0.000000000000000E+000
 Choose fourth-order chirp (b_4[fs^4]):
0
  0.000000000000000E+000

 Done with input.
 Writing out laser field
```

This writes a file called **laser**.

To prepare the actual dynamics simulations, we need to label some state in the **initconds** file as the initial state. Because we want to use the laser pulse to excite the molecule, we set the S_0 as the initial state. The labelling is done with **excite.py**:

```
user@host> $SHARC/excite.py
```

using

```
  ================================================================================
||                                                                              ||
||                     Excite initial conditions for SHARC                      ||
||                                                                              ||
||                             Author: Sebastian Mai                            ||
||                                                                              ||
||                                  Version:4.0                                 ||
||                                    01.04.25                                  ||
||                                                                              ||
  ================================================================================

This script automatizes to read-out the results of initial excited-state calculations
for SHARC.
It calculates oscillator strength (in MCH and diagonal basis) and stochastically
determines initial states for trajectories.

------------------Initial conditions file------------------

Initial conditions file "initconds" detected. Do you want to use this?
Use file "initconds"? [True] <Enter>

File "initconds" contains 50 initial conditions.
Number of atoms is 3

---------------Generate excited state lists---------------

Using the following options, excited state lists can be added to the initial conditions:

1       Generate a list of dummy states
2       Read excited-state information from ab initio calculations (from setup_init.py)

How should the excited-state lists be generated? [2] <Enter>
```

```
Please enter the path to the directory containing the ICOND subdirectories.
Path to ICOND directories: (autocomplete enabled) .

/global/lorenz/sharc_tutorial/prepare
Directory contains 51 subdirectories.

----------------Excited-state representation----------------

This script can calculate the excited-state energies and oscillator strengths in two representations.
These representations are:
- MCH representation: Only the diagonal elements of the Hamiltonian are taken into account.
  The states are the spin-free states as calculated in the quantum chemistry code.
  This option should be used if the ground state is spin-pure.
- diagonal representation: The Hamiltonian including spin-orbit coupling is diagonalized.
  The states are spin-corrected, fully adiabatic. Note that for this the excited-state calculations ...
  This option should be used if the ground state is spin-mixed.

Do you want to use the diagonal representation (True=diag, False=MCH)? [False] <Enter>

----------------------Reference energy----------------------

Reference energy read from file
/global/lorenz/sharc_tutorial/prepare/ICOND_00000/QM.out
E_ref= -548.409106830400

------------------Excited-state selection-------------------

Using the following options, the excited states can be flagged as valid initial states for dynamics:

1       Unselect all initial states
2       Provide a list of desired initial states
3       Simulate delta-pulse excitation based on excitation energies and oscillator strengths

How should the excited states be flagged? [3] 2

---------------------Excitation window----------------------

Enter the energy window for exciting the trajectories.
Range (eV): [0.0 10.0]

Script will allow excitations only between 0.000000 eV and 10.000000 eV.

---------------------Considered states----------------------

Please give a list of all states which should be
flagged as valid initial states for the dynamics.
Note that this is applied to all initial conditions.
# State map for MCH states:
#State        Mult        M_s        Quant
1        1        +0.0        1
2        1        +0.0        2
3        1        +0.0        3
4        3        -1.0        1
5        3        -1.0        2
6        3        -1.0        3
7        3        +0.0        1
8        3        +0.0        2
9        3        +0.0        3
10        3        +1.0        1
11        3        +1.0        2
```

```
12        3        +1.0        3

List of initial states: (range comprehension enabled) 1
--------------------Considered states---------------------

From which state should the excitation originate
(for computation of excitation energies and oscillator strength)?
Lower state for excitation? [1] <Enter>

########################Full input########################

ninit                     50
natom                     3
repr                      MCH
eref                      -548.4091068304
eharm                     0.0
states                    [3, 0, 3]
initf                     <_io.TextIOWrapper name='initconds' mode='r' encoding='UTF-8'>
gen_list                  2
read_QMout                True
make_list                 False
iconddir                  /global/lorenz/sharc_tutorial/prepare
ncond                     51
diag                      False
ion                       False
excite                    2
erange                    [0.0, 0.3674932217565499]
diabatize                 False
allowed                   1
initstate                 0

Do you want to continue? [True]

Reading initial condition file ...
  Progress: [==================================================] 100%
Number of initial conditions in file:          50

Reading QM.out data ...
  Progress: [==================================================] 100%
Number of initial conditions with QM.out:      50

Selecting initial states ...
  Progress: [==================================================] 100%
Number of initial states:                      50

Number of initial conditions excited:
State   Selected   InRange   Total
    1         50        50      50
    2          0        50      50
    3          0        50      50
    4          0        50      50
    5          0        50      50
    6          0        50      50
    7          0        50      50
    8          0        50      50
    9          0        50      50
   10          0        50      50
   11          0        50      50
   12          0        50      50
Writing output to initconds.excited ...
```

Now, you can setup the frozen-nuclei dynamics:

```
user@host> $SHARC/setup_traj.py
```

using

```
  ============================================================================
||                                                                            ||
||                     Setup trajectories for SHARC dynamics                  ||
||                                                                            ||
||          Authors: Sebastian Mai, Philipp Marquetand, Severin Polonius      ||
||                                                                            ||
||                                  Version: 4.0                              ||
||                                 Date: 01.04.25                             ||
||                                                                            ||
  ============================================================================

This script automatizes the setup of the input files for SHARC dynamics.

  ============================================================================
||                              Initial conditions                            ||
  ============================================================================

This script reads the initial conditions (geometries, velocities, initial excited state)
from the initconds.excited files as provided by excite.py.

Please enter the filename of the initial conditions file.
Initial conditions filename: [initconds.excited] (autocomplete enabled) ../prepare/initconds.excited

File ../prepare/initconds.excited contains 50 initial conditions.
Number of atoms is 3
Reference energy -548.409106830400 a.u.
Excited states are in MCH representation.


Please enter the number of states as a list of integers
e.g. 3 0 3 for three singlets, zero doublets and three triplets.
Number of states: [3 0 3] <Enter>


Please enter the molecular charge for each chosen multiplicity
e.g. 0 +1 0 for neutral singlets and triplets and cationic doublets.
Molecular charges per multiplicity: [0 1 0] 0 0 0
Number of states: [3, 0, 3]
Total number of states: 12

Do you want all states to be active? [True] <Enter>

Do you want to see the content of the initconds file? [False] <Enter>
Reading initconds file
  Progress: [================================================================================] 100%
Number of initial conditions in file:          50
Number of excited states and selections:
State    #InitCalc       #Selected
    1           50              50
    2           50               0
    3           50               0
```

```
    4           50              0
    5           50              0
    6           50              0
    7           50              0
    8           50              0
    9           50              0
   10           50              0
   11           50              0
   12           50              0

Please enter a list specifying for which excited states trajectories should be set-up
e.g. 3 6 7 to select states 3, 6, and 7.
States to setup the dynamics: [1] (range comprehension enabled) <Enter>

There can be 50 trajectories set up.

Please enter the index of the first initial condition in the initconds file to be setup.
Starting index: [1] <Enter>

There can be 50 trajectories set up, starting in 1 states.

Please enter the total number of trajectories to setup.
Number of trajectories: [50] <Enter>

Please enter a random number generator seed (type "!" to initialize the RNG from the system time).
RNG Seed:  [!] 1201211801

  ================================================================================
||                          Quantum chemistry interface                           ||
  ================================================================================

Loading interface collection from /user/lorenz/bin/sharc/develop/bin ...

-----------Choose the quantum chemistry interface-----------

Please specify the quantum chemistry interface (enter any of the following numbers):
  1 SHARC_ADAPTIVE          HYBRID interface for adaptive sampling
  2 SHARC_AMS_ADF                  (Not Available! Use SHARC_LEGACY to work with this interface)
  3 SHARC_ANALYTICAL          FAST interface for analytical model Hamiltonians with sympy
  4 SHARC_ASE_DB                      (Not Available!)
  5 SHARC_BAGEL                    (Not Available! Use SHARC_LEGACY to work with this interface)
  6 SHARC_COLUMBUS                 (Not Available! Use SHARC_LEGACY to work with this interface)
  7 SHARC_DO_NOTHING          FAST interface for testing (zero E/grad/NAC/DM, unity overlaps/phases)
  8 SHARC_ECI               HYBRID interface for excitonic HF/CI with multiple fragments
  9 SHARC_FALLBACK          HYBRID interface for calling a fallback interface if primary interface fails
 10 SHARC_GAUSSIAN       AB INITIO interface for GAUSSIAN16 for single-reference methods (CIS, TDDFT)
 11 SHARC_LEGACY            BASIC interface for running legacy interfaces via file I/O (AMS-ADF, ...
 12 SHARC_LVC                FAST interface for linear/quadratic vibronic coupling models (LVC, ...
 13 SHARC_MNDO           AB INITIO interface for the MNDO program (OM2-MRCI)
 14 SHARC_MOLCAS         AB INITIO interface for OpenMolcas (>v23) for multireference ...
 15 SHARC_MOLPRO                  (Not Available! Use SHARC_LEGACY to work with this interface)
 16 SHARC_MOPACPI        AB INITIO interface for the MOPAC-PI program
 17 SHARC_NUMDIFF          HYBRID interface for numerical derivatives (grad, NACdr, SOCdr, DMdr)
 18 SHARC_NWCHEM         AB INITIO interface for NWChem (TDDFT)
 19 SHARC_OPENMM                  (Not Available!)
 20 SHARC_ORCA           AB INITIO interface for ORCA v5-6 (CIS/TDDFT)
 21 SHARC_PYSCF                   (Not Available! Use SHARC_LEGACY to work with this interface)
 22 SHARC_QMMM             HYBRID interface for QM/MM (electrostatic embedding, link atom scheme)
 23 SHARC_QMOUT              FAST interface for frozen-nuclei dynamics (constant E/SOC/DM, unit ...
 24 SHARC_SCHNARC                 (Not Available!)
```

```
 25 SHARC_SPAINN                  (Not Available!)
 26 SHARC_TURBOMOLE      AB INITIO interface for TURBOMOLE (RICC2/ADC2)
 27 SHARC_UMBRELLA          HYBRID interface for adding umbrella-sampling-style restraints ...

Interface number: 23

The following interface was selected:
 23 SHARC_QMOUT              FAST interface for frozen-nuclei dynamics (constant E/SOC/DM ...


The following features are available from this interface:
'multipolar_fit', 'socdr', 'grad', 'dmdr', 'soc', 'h', 'dm', 'phases', 'nacdr', 'overlap'

  ================================================================================
||                      Surface Hopping dynamics settings                       ||
  ================================================================================


----------------Nonadiabatic dynamics method---------------

Please choose the dynamics method you want to employ.
1        Trajectory surface hopping dynamics using single surface potential
2        Semi-classical Ehrenfest dynamics using self-consistent potential
Method: [1] <Enter>
----------------------Simulation time----------------------

Please enter the total simulation time.
Simulation time (fs): [1000.0] 10

Please enter the simulation timestep (0.5 fs recommended).
Simulation timestep (fs): [0.5] 0.05

Simulation will have 200 timesteps.

Please choose the integrator you want to use
1        adaptive timestep Velocity-Verlet integrator
2        fixed timestep Velocity-Verlet integrator
Integrator: [2] <Enter>

Please enter the number of substeps for propagation (25 recommended).
Nsubsteps: [25] 1

The trajectories can be prematurely terminated after they run for a certain time in the lowest state.
Do you want to prematurely terminate trajectories? [False] <Enter>

---------------------Dynamics settings---------------------

Do you want to perform the dynamics in the diagonal representation (SHARC dynamics) or ...
SHARC dynamics? [True] <Enter>
Do you want to include spin-orbit couplings in the dynamics?
Spin-Orbit calculation? [True] <Enter>
Will calculate spin-orbit matrix.

Please choose the quantities to describe non-adiabatic effects between the states:
1        DDT     =  < a|d/dt|b >           Hammes-Schiffer-Tully scheme (not available)
2        DDR     =  < a|d/dR|b >           Original Tully scheme
3        ktdc    = sqrt(D2(dV)/dt2/(dV))/2  Curvature Driven TDC scheme
4        overlap = < a(t0)|b(t) >          Local Diabatization scheme
Coupling number: [4] <Enter>
```

```
1        mixed gradients are calculated as linear combination of MCH gradients only
2        mixed gradients are calculated by correction of MCH gradients with non-adiabatic coupling vector
3        mixed gradients are calculated by rescaling of the MCH gradients according to time derivatives ...

Please choose the gradient mixing scheme for the gradients:
Gradient mixing scheme: [1] <Enter>


During a surface hop, the kinetic energy has to be modified in order to conserve total energy. ...
1        Do not conserve total energy. Hops are never frustrated.
2        Adjust kinetic energy by rescaling the velocity vectors. Often sufficient.
3        Adjust kinetic energy only with the component of the velocity vector along the ...
4        Adjust kinetic energy only with the component of the velocity vector along the ...
5        Adjust kinetic energy only with the component of the velocity vector along the ...
6        Adjust kinetic energy only with the component of the velocity vector along the ...
7        Adjust kinetic energy only with the component of the velocity vector along the ...
8        Adjust kinetic energy only with the component of the velocity vector along the ...
EkinCorrect: [2] <Enter>

If a surface hop is refused (frustrated) due to insufficient energy, the velocity ...
1        Do not reflect at a frustrated hop.
2        Reflect the full velocity vector.
3        Reflect the vibrational velocity vector.
4        Reflect only the component of the velocity vector along the non-adiabatic coupling vector.
5        Reflect only the component of the velocity vector along the gradient difference vector.
6        Reflect only the component of the velocity vector along the projected ...
7        Reflect only the component of the velocity vector along the effective ...
8        Reflect only the component of the velocity vector along the projected effective ...
Reflect frustrated: [1] <Enter>

Please choose a decoherence correction for the diagonal states:
1        No decoherence correction.
2        Energy-based decoherence scheme (Granucci, Persico, Zoccante).
3        Augmented fewest-switching surface hopping (Jain, Alguire, Subotnik).
Decoherence scheme: [2] <Enter>

Please choose a surface hopping scheme for the diagonal states:
1        Surface hops off.
2        Standard SHARC surface hopping probabilities (Mai, Marquetand, Gonzalez).
3        Global flux surface hopping probabilities (Wang, Trivedi, Prezhdo).
Hopping scheme: [2] <Enter>

Do you want to perform forced hops to the lowest state based on a energy gap criterion?
(Note that this ignores spin multiplicity)
Forced hops to ground state? [False] <Enter>

Do you want to scale the energies and gradients?
Scaling? [False] <Enter>

Do you want to damp the dynamics (Kinetic energy is reduced at each timestep by a factor)?
Damping? [False] <Enter>

Do you want to use an atom mask for velocity rescaling or decoherence?
Atom masking? [False] <Enter>

--------------Selection of Gradients and NACs---------------

In order to speed up calculations, SHARC is able to select which gradients and NAC vectors it has to ...

Select gradients? [False] <Enter>
```

```
-----------------Settings for large systems-----------------

Do you want to constrain some bond lengths (via a RATTLE)? [False] <Enter>
Do you want to use a thermostat? [False] <Enter>

------------------------Laser file------------------------

Do you want to include a laser field in the simulation? [False] y
Please specify the file containing the complete laser field. The timestep in the file and the length of ...

Laser files can be created using $SHARC/laser.x

Laser filename: (autocomplete enabled) ../prepare/laser
Laser file must have 201 steps and a time step of 0.050000 fs.

  ================================================================================
||                                Interface setup                                 ||
  ================================================================================

Please provide path to QM.out file or to folder containing ICOND folders [QM.out] (autocomplete enabled)
 ../prepare/
Sym-link the file? (no = copy)? [False] <Enter>

  ================================================================================
||                                    PYSHARC                                     ||
  ================================================================================

The chosen interface can be run very efficiently with PYSHARC.
PYSHARC runs the SHARC dynamics directly within Python (with C and Fortran extension)
with minimal file I/O for maximum performance.
Setup for PYSHARC? [True] <Enter>

  ================================================================================
||                           Content of output.dat files                          ||
  ================================================================================

SHARC or PYSHARC can produce output in ASCII format (all features supported currently)
or in NetCDF format (more efficient file I/O, trajectory restart currently not supported).
Write output in NetCDF format? [True]
Write nuclear and electronic date to separate NetCDF files? [False] <Enter>

Do you want to write the overlap matrix to the output.dat file ?
Write overlap matrix? [True] <Enter>

Do you want to modify the output.dat writing stride?
Modify stride? [False] <Enter>

  ================================================================================
||                                Run mode setup                                  ||
  ================================================================================

-------------------------Run script-------------------------

This script can generate the run scripts for each initial condition in two modes:

  - In mode 1, the calculation is run in subdirectories of the current directory.

  - In mode 2, the input files are transferred to another directory (e.g. a local scratch directory), ...
```

```
Note that in any case this script will create the input subdirectories in the current working directory.

In case of mode 1, the calculations will be run in:
/global/lorenz/sharc_tutorial/traj

Use mode 1 (i.e., calculate here)? [True] <Enter>

---------------------Submission script----------------------

During the setup, a script for running all initial conditions sequentially in batch mode is generated. ...

Generate submission script? [False] <Enter>

########################Full input#########################

select_directly          True
ninit                    50
natom                    3
repr                     MCH
diag                     False
eref                     -548.4091068304
eharm                    0.0
initf                    <_io.TextIOWrapper name='../prepare/initconds.excited' mode='r' encoding='UTF-8'>
states                   [3, 0, 3]
nstates                  12
charge                   [0, 0, 0]
statemap                 {1: [1, 1, 0.0], 2: [1, 2, 0.0], 3: [1, 3, 0.0], 4: [3, 1, -1.0],
5: [3, 2, -1.0], 6: [3, 3, -1.0], 7: [3, 1, 0.0], 8: [3, 2, 0.0], 9: [3, 3, 0.0],
10: [3, 1, 1.0], 11: [3, 2, 1.0], 12: [3, 3, 1.0]}
actstates                [3, 0, 3]
isactive                 [True, True, True, True, True, True, True, True, True, True, True, True]
show_content             False
n_issel                  [50, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
setupstates              1
firstindex               1
ntraj                    50
needed_requests          'grad', 'soc', 'h', 'dm', 'overlap'
method                   tsh
tmax                     10.0
dtstep                   0.05
integrator               2
nsubstep                 1
kill                     False
surf                     diagonal
soc                      True
coupling                 4
phases_from_interface    False
gradcorrect              1
ekincorrect              2
reflect                  1
decoherence              ['edc', '0.1']
hopping                  sharc
force_hops               False
force_hops_dE            9999.0
scaling_for_sharc        False
damping                  False
atommaskarray            None
sel_g                    False
sel_t                    False
```

```
rattle                    False
use_thermostat            False
laser                     True
laserfile                 ../prepare/laser
dipolegrad                False
ion                       False
pysharc                   True
netcdf                    True
netcdf_separate           False
write_grad                False
write_NAC                 False
write_property2d          False
write_property1d          False
write_overlap             True
stride                    [1]
log.infolevel             2
cwd                       /global/lorenz/sharc_tutorial/traj
here                      True
copydir                   /global/lorenz/sharc_tutorial/traj
qsub                      False

Do you want to setup the specified calculations? [True] <Enter>

Do you want to link the interface files? [False] <Enter>

  ================================================================================
||                          Setting up directories...                           ||
  ================================================================================

Progress: [================================================] 100%

Do you want to see the input for the first trajectory? [False] <Enter>
Do you want to add keywords to the input of all trajectories? [False] <Enter>
Progress: [================================================] 100%

50 trajectories setup, last initial condition was 50 in state 1.
```

Running these trajectories, one would expect some population transfer to the S_1 state, purely driven by the laser. These are laser-induced hops that do not conserve total energy. In rare cases, if a triplet is energetically near the S_1, one can also see a small amount of triplet population.

3.10 Adaptive training with SPaiNN

Do optimization plus frequency calculation with ORCA. The following is already the optimized structure for convenience.

```
! wB97X-D3 def2-SV(P) rijcosx OPT FREQ

* xyz 0 1
  C          -0.05633993616824      0.01746212853377     -0.08387132777497
  N           1.38829832245032      0.00921160318234     -0.04854009300850
  C           2.06773412673979     -1.18255210437956     -0.00094495182775
  C           1.38189100735409     -2.42121627506315      0.01758006652841
  C           2.07245405099966     -3.62158202455212      0.06606964982132
  C           3.47345576322132     -3.64818659135451      0.09840930589575
  C           4.16402510961392     -2.42857560445056      0.08014279745560
  C           3.48285368870638     -1.22296834726043      0.03149548886752
  H           4.06539766553870     -0.30099845202566      0.01902773227089
  H           5.25722702166292     -2.42744863629714      0.10482681888155
  C           4.18517948925330     -4.89567990558505      0.14939376373846
  N           4.75861965808903     -5.90091929967083      0.19039633395203
  H           1.51464380198197     -4.56201707687994      0.07908767616201
  H           0.29192207669245     -2.45316705117500     -0.00636954098950
  C           2.11805644386287      1.25650449034238     -0.06906079415493
  H           2.77517968627856      1.33696031680251     -0.95599934797307
  H           1.40644090077805      2.09349349204734     -0.10412681574928
  H           2.74312882902302      1.38480398753668      0.83502716177946
  H          -0.45246304801208     -0.50389984611449     -0.97611659232793
  H          -0.49564407632700     -0.45603916170237      0.81479622149453
  H          -0.41225258173906      1.05688735806579     -0.11982855304159
*
```

Load necessary ORCA environment.

```
user@host> orca orca.inp > orca.log
```

Generate molden file including frequencies.

```
user@host> $SHARC/ORCA_hess_freq.py orca.hess
```

Prepare ORCA resources and template files ORCA.template

See Section 6.10.4 (p. 98) in the manual.

```
basis def2-SV(P)
functional wB97X-D3
ri rijcosx
maxiter 700
```

ORCA.resources (replace orcadir and/or scratchdir to existing paths on your machine)

```
orcadir $ORCADIR
scratchdir $TMPDIR
ncpu 1
memory 4000
wfoverlap $SHARC/wfoverlap.x
wfthres 2.0
```

Alternatively, during the setup script the resources can be entered as well.

Generate initial conditions.

`user@host> $SHARC/wigner.py -n 50 orca.hess.molden`

Set up the initial computations

`user@host> $SHARC/setup_init.py`

using

```
Script for setup of initial conditions started...

  ================================================================================
||                                                                                ||
||                    Setup trajectories for SHARC dynamics                       ||
||                                                                                ||
||                   Authors: Sebastian Mai, Severin Polonius                     ||
||                                                                                ||
||                                Version: 4.0                                    ||
||                               Date: 01.04.25                                   ||
||                                                                                ||
  ================================================================================

This script automatizes the setup of excited-state calculations for initial conditions
for SHARC dynamics.

------------------Initial conditions file-------------------

Initial conditions file "initconds" detected. Do you want to use this?
Use file "initconds"? [True] <ENTER>

File "initconds" contains 50 initial conditions.
Number of atoms is 21

----------------Range of initial conditions-----------------

Please enter the range of initial conditions for which an excited-state calculation
should be performed as two integers separated by space.
Initial condition range: [1 50] <ENTER>

Script will use initial conditions 1 to 50 (50 in total).

----------------Number of states and charge-----------------

Please enter the number of states as a list of integers
e.g. 3 0 3 for three singlets, zero doublets and three triplets.
Number of states: 3

Please enter the molecular charge for each chosen multiplicity
```

```
e.g. 0 +1 0 for neutral singlets and triplets and cationic doublets.
Molecular charges per multiplicity: [0] <ENTER>
Number of states: [3]
Total number of states: 3

Loading interface collection from /user/sharc/sharc_main/bin ...

-----------Choose the quantum chemistry interface-----------

Please specify the quantum chemistry interface (enter any of the following numbers):
  1 SHARC_ADAPTIVE          HYBRID interface for adaptive sampling
  2 SHARC_AMS_ADF                  (Not Available! Use SHARC_LEGACY to work with this interface)
  3 SHARC_ANALYTICAL          FAST interface for analytical model Hamiltonians with sympy
  4 SHARC_ASE_DB            HYBRID interface for saveing data to ASE db
  5 SHARC_BAGEL                    (Not Available! Use SHARC_LEGACY to work with this interface)
  6 SHARC_COLUMBUS                 (Not Available! Use SHARC_LEGACY to work with this interface)
  7 SHARC_DO_NOTHING          FAST interface for testing (zero E/grad/NAC/DM, unity overlaps/phases)
  8 SHARC_ECI               HYBRID interface for excitonic HF/CI with multiple fragments
  9 SHARC_FALLBACK          HYBRID interface for calling a fallback interface if primary interface fails
 10 SHARC_GAUSSIAN       AB INITIO interface for GAUSSIAN16 for single-reference methods (CIS, TDDFT)
 11 SHARC_LEGACY             BASIC interface for running legacy interfaces via file I/O (AMS-ADF, ...
 12 SHARC_LVC                 FAST interface for linear/quadratic vibronic coupling models (LVC, QVC ...
 13 SHARC_MNDO           AB INITIO interface for the MNDO program (OM2-MRCI)
 14 SHARC_MOLCAS         AB INITIO interface for OpenMolcas (>v23) for multireference calculations ...
 15 SHARC_MOLPRO                  (Not Available! Use SHARC_LEGACY to work with this interface)
 16 SHARC_MOPACPI        AB INITIO interface for the MOPAC-PI program
 17 SHARC_NUMDIFF           HYBRID interface for numerical derivatives (grad, NACdr, SOCdr, DMdr)
 18 SHARC_NWCHEM         AB INITIO interface for NWChem (TDDFT)
 19 SHARC_OPENMM                  (Not Available!)
 20 SHARC_ORCA           AB INITIO interface for ORCA v5-6 (CIS/TDDFT)
 21 SHARC_PYSCF                   (Not Available! Use SHARC_LEGACY to work with this interface)
 22 SHARC_QMMM              HYBRID interface for QM/MM (electrostatic embedding, link atom scheme)
 23 SHARC_QMOUT               FAST interface for frozen-nuclei dynamics (constant E/SOC/DM ...
 24 SHARC_SCHNARC                 (Not Available!)
 25 SHARC_SPAINN              FAST interface for SPaiNN
 26 SHARC_TURBOMOLE      AB INITIO interface for TURBOMOLE (RICC2/ADC2)
 27 SHARC_UMBRELLA          HYBRID interface for adding umbrella-sampling-style restraints (harmonic ...

Interface number: 20

The following interface was selected:
 20 SHARC_ORCA           AB INITIO interface for ORCA v5-6 (CIS/TDDFT)

The following features are available from this interface:
'grad_pc', 'overlap', 'dm', 'ion', 'h', 'point_charges', 'theodore', 'molden', 'phases', 'grad', 'soc'

----------------Spin-orbit couplings (SOCs)-----------------

Only singlets specified: not calculating spin-orbit matrix.


----------------Overlaps to reference states----------------

Do you want to compute the overlaps between the states at the equilibrium geometry and the states at ...
Reference overlaps? [False] <ENTER>

--------------TheoDORE wave function analysis---------------

Do you want to run TheoDORE to obtain one-electron descriptors for the electronic wave functions?
TheoDORE? [False] <ENTER>
```

```
================================================================================
||                                                                            ||
||                          ORCA interface setup                              ||
||                                                                            ||
================================================================================

Please specify path to ORCA directory (SHELL variables and ~ can be used, will be expanded ...

Path to ORCA: (autocomplete enabled) $ORCADIR

--------------------Scratch directory---------------------

Please specify an appropriate scratch directory. This will be used to run the ORCA calculations.
The scratch directory will be deleted after the calculation. Remember that this script cannot check ...
Path to scratch directory: (autocomplete enabled) $TMPDIR

-----------------ORCA input template file-----------------

Template filename: (autocomplete enabled) ORCA.template

Do you have a 'ORCA.resources' file? [False] yes
Specify the path: [ORCA.resources] (autocomplete enabled) ORCA.resources

  ================================================================================
||                                Run mode setup                                ||
  ================================================================================

------------------------Run script------------------------

This script can generate the run scripts for each initial condition in two modes:
  - In mode 1, the calculation is run in subdirectories of the current directory.
  - In mode 2, the input files are transferred to another directory (e.g. a local scratch directory) ...

Note that in any case this script will create the input subdirectories in the current working directory.

In case of mode 1, the calculations will be run in:
/user/sharc/sharc_tutorial

Use mode 1 (i.e., calculate here)? [True]  <ENTER>

--------------------Submission script---------------------

During the setup, a script for running all initial conditions sequentially in batch mode is generated. ...

Generate submission script? [False] <ENTER>

########################Full input#########################

cwd                     /user/sharc/sharc_tutorial
ninit                   50
natom                   21
initf                   <_io.TextIOWrapper name='initconds' mode='r' encoding='UTF-8'>
irange                  [1, 50]
states                  [3]
nstates                 3
charge                  [0]
needed_requests         'h', 'dm'
soc                     False
refov                   False
```

```
theodore                  False
here                      True
qsub                      False

Do you want to setup the specified calculations? [True] <ENTER>

Do you want to link the interface files? [False]  <ENTER>

  ================================================================================
||                          Setting up directories...                           ||
  ================================================================================


Progress: [================================================] 100%
```

Important: QM.in does not contain gradient request which is needed to train the machine learning model. Add them with

```
user@host> for i in ICOND*; do echo "grad" >> $i/QM.in; done
```

Launch initial conditions

Create db

```
user@host> mkdir spainn && cd spainn
user@host> mkdir data
user@host> spainn-db generate ../ data/init.db
```

Create config files for training SPaiNN models

```
user@host> mkdir experiment
```

generate file data/dmabn.yaml

```
_target_: spainn.SPAINN

datapath: ${run.data_dir}/init.db
n_states: 3
n_nacs: 3
data_workdir: null
batch_size: 5
num_train: 41
num_val: 5
num_test: null
num_workers: 16
num_val_workers: null
num_test_workers: null
```

generate file experiment/dmabn.yaml

```
# @package _global_
defaults:
  - override /model: nnp
  - override /data: dmabn
run:
  experiment: dmabn
globals:
```

```
  cutoff: 10.
  lr: 1.0e-5
  energy_key: energy
  forces_key: forces
  equivariant: True
data:
  distance_unit: Bohr
  property_units:
    energy: Hartree
    forces: Hartree/Bohr
  transforms:
    - _target_: schnetpack.transform.RemoveOffsets
      property: energy
      remove_mean: True
    - _target_: schnetpack.transform.MatScipyNeighborList
      cutoff: ${globals.cutoff}
    - _target_: schnetpack.transform.CastTo64
model:
  output_modules:
    - _target_: spainn.Atomwise
      output_key: ${globals.energy_key}
      n_out: 3
      n_layers: 3
      n_in: ${model.representation.n_atom_basis}
      aggregation_mode: sum
    - _target_: spainn.Forces
      energy_key: ${globals.energy_key}
      force_key: ${globals.forces_key}
  postprocessors:
    - _target_: schnetpack.transform.AddOffsets
      property: energy
      add_mean: True
task:
  outputs:
    - _target_: schnetpack.task.ModelOutput
      name: ${globals.energy_key}
      loss_fn:
        _target_: torch.nn.MSELoss
      metrics:
        mae:
          _target_: torchmetrics.regression.MeanAbsoluteError
        mse:
          _target_: torchmetrics.regression.MeanSquaredError
      loss_weight: 1
    - _target_: schnetpack.task.ModelOutput
      name: ${globals.forces_key}
      loss_fn:
        _target_: torch.nn.MSELoss
      metrics:
        mae:
          _target_: torchmetrics.regression.MeanAbsoluteError
        mse:
          _target_: torchmetrics.regression.MeanSquaredError
      loss_weight: 0.5
```

train 2 models

```
user@host> spainn-train experiment=dmabn run.id=init1
user@host> spainn-train experiment=dmabn run.id=init2
```

the trained models will be stored in runs/init1/best_model and runs/init2/best_model

Setup trajectories, first excite

```
user@host> cd ..
user@host> $SHARC/excite.py
```

```
  ============================================================================
||                                                                            ||
||                     Excite initial conditions for SHARC                    ||
||                                                                            ||
||                            Author: Sebastian Mai                           ||
||                                                                            ||
||                                 Version:4.0                                ||
||                                   01.04.25                                 ||
||                                                                            ||
  ============================================================================


This script automatizes to read-out the results of initial excited-state calculations for SHARC.
It calculates oscillator strength (in MCH and diagonal basis) and stochastically
determines initial states for trajectories.

------------------Initial conditions file-------------------

Initial conditions file "initconds" detected. Do you want to use this?
Use file "initconds"? [True] <ENTER>

File "initconds" contains 50 initial conditions.
Number of atoms is 21

----------------Generate excited state lists----------------

Using the following options, excited state lists can be added to the initial conditions:

1       Generate a list of dummy states
2       Read excited-state information from ab initio calculations (from setup_init.py)

How should the excited-state lists be generated? [2] <ENTER>
Please enter the path to the directory containing the ICOND subdirectories.
Path to ICOND directories: (autocomplete enabled) .

/user/sharc/sharc_tutorial
Directory contains 51 subdirectories.

----------------Excited-state representation----------------

This script can calculate the excited-state energies and oscillator strengths in two representations.
These representations are:
- MCH representation: Only the diagonal elements of the Hamiltonian are taken into account.
  The states arethe spin-free states as calculated in the quantum chemistry code.
  This option should be used if the ground state is spin-pure.
- diagonal representation: The Hamiltonian including spin-orbit coupling is diagonalized.
  The states are spin-corrected, fully adiabatic.
  Note that for this the excited-state calculations have to include spin-or
bit couplings.
  This option should be used if the ground state is spin-mixed.

Do you want to use the diagonal representation (True=diag, False=MCH)? [False] <ENTER>
```

```
----------------------Reference energy----------------------

Reference energy read from file
/user/sharc/sharc_tutorial/ICOND_00000/QM.out
E_ref= -457.979048610500


------------------Excited-state selection-------------------

Using the following options, the excited states can be flagged as valid initial states for dynamics:

1       Unselect all initial states
2       Provide a list of desired initial states
3       Simulate delta-pulse excitation based on excitation energies and oscillator strengths

How should the excited states be flagged? [3] 2


---------------------Excitation window----------------------

Enter the energy window for exciting the trajectories.
Range (eV): [0.0 10.0] <ENTER>

Script will allow excitations only between 0.000000 eV and 10.000000 eV.


--------------------Considered states----------------------

Please give a list of all states which should be
flagged as valid initial states for the dynamics.
Note that this is applied to all initial conditions.
# State map for MCH states:
#State  Mult    M_s     Quant
1       1       +0.0    1
2       1       +0.0    2
3       1       +0.0    3

List of initial states: (range comprehension enabled) 3
--------------------Considered states----------------------

From which state should the excitation originate (for computation of excitation energies and oscillator
strength)?
Lower state for excitation? [1] <ENTER>


#########################Full input#########################

ninit                   50
natom                   21
repr                    MCH
eref                    -457.9790486105
eharm                   0.0
states                  [3]
initf                   <_io.TextIOWrapper name='initconds' mode='r' encoding='UTF-8'>
gen_list                2
read_QMout              True
make_list               False
iconddir                /user/sharc/sharc_tutorial
ncond                   51
diag                    False
```

```
ion                         False
excite                      2
erange                      [0.0, 0.3674932217565499]
diabatize                   False
allowed                     3
initstate                   0

Do you want to continue? [True] <ENTER>

Reading initial condition file ...
  Progress: [==================================================] 100%
Number of initial conditions in file:          50

Reading QM.out data ...
  Progress: [==================================================] 100%
Number of initial conditions with QM.out:      50

Selecting initial states ...
  Progress: [==================================================] 100%
Number of initial states:                      50

Number of initial conditions excited:
State   Selected   InRange   Total
    1          0        50      50
    2          0        50      50
    3         50        50      50
Writing output to initconds.excited ...
```

Before setting up trajectories, prepare template and resources files For the active learning workflow we need FALLBACK.template

```
trial_interface: {"interface": "ADAPTIVE", "args": [], "kwargs": {"fast_queue": True}}
fallback_interface: {"interface": "ASE_DB", "args": [], "kwargs": {}}
```

ADAPTIVE.template

```
---
thresholds: "h": 2e-7
interfaces:
    - {"label": "init1", "interface": "SPAINN", "args": [], "kwargs": {}}
    - {"label": "init2", "interface": "SPAINN", "args": [], "kwargs": {}}
error_function: "mae"
```

ADAPTIVE.resources

```
ncpu 1
memory 5000
```

and ASE_DB.template

```
reference: {"interface": "ORCA", "args": [], "kwargs": {}}
props_to_save: ["h", "grad"]
ase_file: "adaptive_run1.db"
```

then

```
user@host> touch SPAINN.template
```

```
user@host> mkdir traj && cd traj
user@host> $SHARC/setup_traj.py
```

```
Script for setup of SHARC trajectories started...


  ================================================================================
||                                                                              ||
||                    Setup trajectories for SHARC dynamics                     ||
||                                                                              ||
||          Authors: Sebastian Mai, Philipp Marquetand, Severin Polonius        ||
||                                                                              ||
||                                  Version: 4.0                                ||
||                                 Date: 01.04.25                               ||
||                                                                              ||
  ================================================================================


This script automatizes the setup of the input files for SHARC dynamics.


  ================================================================================
||                               Initial conditions                             ||
  ================================================================================



This script reads the initial conditions (geometries, velocities, initial excited state)
from the initconds.excited files as provided by excite.py.

Please enter the filename of the initial conditions file.
Initial conditions filename: [initconds.excited] (autocomplete enabled) ../initconds.excited

File ../initconds.excited contains 50 initial conditions.
Number of atoms is 21
Reference energy -457.979048610500 a.u.
Excited states are in MCH representation.


Please enter the number of states as a list of integers
e.g. 3 0 3 for three singlets, zero doublets and three triplets.
Number of states: [3] <ENTER>


Please enter the molecular charge for each chosen multiplicity
e.g. 0 +1 0 for neutral singlets and triplets and cationic doublets.
Molecular charges per multiplicity: [0] <ENTER>
Number of states: [3]
Total number of states: 3

Do you want all states to be active? [True] <ENTER>

Do you want to see the content of the initconds file? [False] <ENTER>
Reading initconds file
  Progress: [================================================================================] 100%
Number of initial conditions in file:          50
Number of excited states and selections:
```

```
State    #InitCalc       #Selected
    1           50               0
    2           50               0
    3           50              50

Please enter a list specifying for which excited states trajectories should be set-up
e.g. 1 2 3 to select states 1, 2, and 3.
States to setup the dynamics: [3] (range comprehension enabled) <ENTER>

There can be 50 trajectories set up.

Please enter the index of the first initial condition in the initconds file to be setup.
Starting index: [1] <ENTER>

There can be 50 trajectories set up, starting in 1 states.

Please enter the total number of trajectories to setup.
Number of trajectories: [50] 10

Please enter a random number generator seed (type "!" to initialize the RNG from the system time).
RNG Seed:  [!] <ENTER>


  ================================================================================
||                          Quantum chemistry interface                           ||
  ================================================================================

Loading interface collection from /user/sharc/sharc_main/bin ...

-----------Choose the quantum chemistry interface-----------

Please specify the quantum chemistry interface (enter any of the following numbers):
  1 SHARC_ADAPTIVE          HYBRID interface for adaptive sampling
  2 SHARC_AMS_ADF                  (Not Available! Use SHARC_LEGACY to work with this interface)
  3 SHARC_ANALYTICAL          FAST interface for analytical model Hamiltonians with sympy
  4 SHARC_ASE_DB            HYBRID interface for saveing data to ASE db
  5 SHARC_BAGEL                    (Not Available! Use SHARC_LEGACY to work with this interface)
  6 SHARC_COLUMBUS                 (Not Available! Use SHARC_LEGACY to work with this interface)
  7 SHARC_DO_NOTHING          FAST interface for testing (zero E/grad/NAC/DM, unity overlaps/phases)
  8 SHARC_ECI               HYBRID interface for excitonic HF/CI with multiple fragments
  9 SHARC_FALLBACK          HYBRID interface for calling a fallback interface if primary interface fails
 10 SHARC_GAUSSIAN       AB INITIO interface for GAUSSIAN16 for single-reference methods (CIS, TDDFT)
 11 SHARC_LEGACY             BASIC interface for running legacy interfaces via file I/O (AMS-ADF, ...
 12 SHARC_LVC                 FAST interface for linear/quadratic vibronic coupling models (LVC, QVC, ...
 13 SHARC_MNDO           AB INITIO interface for the MNDO program (OM2-MRCI)
 14 SHARC_MOLCAS         AB INITIO interface for OpenMolcas (>v23) for multireference calculations ...
 15 SHARC_MOLPRO                  (Not Available! Use SHARC_LEGACY to work with this interface)
 16 SHARC_MOPACPI        AB INITIO interface for the MOPAC-PI program
 17 SHARC_NUMDIFF           HYBRID interface for numerical derivatives (grad, NACdr, SOCdr, DMdr)
 18 SHARC_NWCHEM         AB INITIO interface for NWChem (TDDFT)
 19 SHARC_OPENMM                  (Not Available!)
 20 SHARC_ORCA           AB INITIO interface for ORCA v5-6 (CIS/TDDFT)
 21 SHARC_PYSCF                   (Not Available! Use SHARC_LEGACY to work with this interface)
 22 SHARC_QMMM              HYBRID interface for QM/MM (electrostatic embedding, link atom scheme)
 23 SHARC_QMOUT               FAST interface for frozen-nuclei dynamics (constant E/SOC/DM, ...
 24 SHARC_SCHNARC                 (Not Available!)
 25 SHARC_SPAINN              FAST interface for SPaiNN
 26 SHARC_TURBOMOLE      AB INITIO interface for TURBOMOLE (RICC2/ADC2)
 27 SHARC_UMBRELLA          HYBRID interface for adding umbrella-sampling-style restraints (harmonic ...
```

```
Interface number: 9

The following interface was selected:
  9 SHARC_FALLBACK          HYBRID interface for calling a fallback interface if primary interface fails

Please specify the path to your FALLBACK.template file [FALLBACK.template] (autocomplete enabled) ../FALLBACK.template
Please specify the path to your ADAPTIVE.template file [ADAPTIVE.template] (autocomplete enabled) ../ADAPTIVE.template
Please specify the path to your ASE_DB.template file [ASE_DB.template] (autocomplete enabled) ../ASE_DB.template

The following features are available from this interface:
'dm', 'h', 'grad'

  ================================================================================
||                      Surface Hopping dynamics settings                       ||
  ================================================================================


----------------Nonadiabatic dynamics method---------------

Please choose the dynamics method you want to employ.
1       Trajectory surface hopping dynamics using single surface potential
2       Semi-classical Ehrenfest dynamics using self-consistent potential
Method: [1] <ENTER>
----------------------Simulation time----------------------

Please enter the total simulation time.
Simulation time (fs): [1000.0] <ENTER>

Please enter the simulation timestep (0.5 fs recommended).
Simulation timestep (fs): [0.5] <ENTER>

Simulation will have 2001 timesteps.

Please choose the integrator you want to use
1       adaptive timestep Velocity-Verlet integrator
2       fixed timestep Velocity-Verlet integrator
Integrator: [2] <ENTER>

Please enter the number of substeps for propagation (25 recommended).
Nsubsteps: [25] <ENTER>

The trajectories can be prematurely terminated after they run for a certain time in the lowest state.
Do you want to prematurely terminate trajectories? [False] <ENTER>


--------------------Dynamics settings---------------------

Do you want to perform the dynamics in the diagonal representation (SHARC dynamics) or in the MCH
representation (regular TSH/SCP)?
SHARC dynamics? [True] <ENTER>
Only singlets specified: not calculating spin-orbit matrix.


Please choose the quantities to describe non-adiabatic effects between the states:
1       DDT     =  < a|d/dt|b >            Hammes-Schiffer-Tully scheme (not available)
2       DDR     =  < a|d/dR|b >            Original Tully scheme        (not available)
3       ktdc    = sqrt(D2(dV)/dt2/(dV))/2  Curvature Driven TDC scheme
4       overlap = < a(t0)|b(t) >           Local Diabatization scheme   (not available)
Coupling number: [3] <ENTER>
```

```
1       mixed gradients are calculated as ...
2       mixed gradients are calculated by ...
3       mixed gradients are calculated by ...

Please choose the gradient mixing scheme for the gradients:
Gradient mixing scheme: [1] <ENTER>


During a surface hop, the kinetic energy has to be modified in order to conserve total energy.
There are several options to that:
1       Do not conserve total energy. Hops are never frustrated.
2       Adjust kinetic energy by rescaling the velocity vectors. Often sufficient.
3       Adjust kinetic energy ... the vibrational velocity vector.
4       Adjust kinetic energy ... the non-adiabatic coupling vector. (not possible)
5       Adjust kinetic energy ... the gradient difference vector.
6       Adjust kinetic energy ... the projected non-adiabatic coupling vector. (not possible)
7       Adjust kinetic energy ... the effective non-adiabatic coupling vector.
8       Adjust kinetic energy ... the projected effective non-adiabatic coupling vector.
EkinCorrect: [2] <ENTER>

If a surface hop is refused (frustrated) due to insufficient energy,
the velocity can either be left unchanged or reflected:
1       Do not reflect at a frustrated hop.
2       Reflect the full velocity vector.
3       Reflect the vibrational velocity vector.
4       Reflect only the component of the velocity vector along the non-adiabatic coupling vector.
        (not possible)
5       Reflect only the component of the velocity vector along the gradient difference vector.
6       Reflect only the component of the velocity vector along the projected non-adiabatic coupling vector.
        (not possible)
7       Reflect only the component of the velocity vector along the effective non-adiabatic coupling vector.
8       Reflect only the component ... projected effective non-adiabatic coupling vector.
Reflect frustrated: [1] <ENTER>

Please choose a decoherence correction for the diagonal states:
1       No decoherence correction.
2       Energy-based decoherence scheme (Granucci, Persico, Zoccante).
3       Augmented fewest-switching surface hopping (Jain, Alguire, Subotnik).
Decoherence scheme: [2] <ENTER>

Please choose a surface hopping scheme for the diagonal states:
1       Surface hops off.
2       Standard SHARC surface hopping probabilities (Mai, Marquetand, Gonzalez).
3       Global flux surface hopping probabilities (Wang, Trivedi, Prezhdo).
Hopping scheme: [2] <ENTER>

Do you want to perform forced hops to the lowest state based on a energy gap criterion?
(Note that this ignores spin multiplicity)
Forced hops to ground state? [False] <ENTER>

Do you want to scale the energies and gradients?
Scaling? [False] <ENTER>

Do you want to damp the dynamics (Kinetic energy is reduced at each timestep by a factor)?
Damping? [False] <ENTER>

Do you want to use an atom mask for velocity rescaling or decoherence?
Atom masking? [False] <ENTER>

--------------Selection of Gradients and NACs---------------
```

```
In order to speed up calculations, SHARC is able to select which gradients and NAC vectors
it has to calculate at a certain timestep. The selection is based on the energy difference
between the state under consideration and the classical occupied state.


SHARC dynamics without SOC and NAC: setting minimal selection threshold.


-----------------Settings for large systems-----------------

Do you want to constrain some bond lengths (via a RATTLE)? [False] <ENTER>
Do you want to use a thermostat? [False] <ENTER>



------------------------Laser file------------------------

Do you want to include a laser field in the simulation? [False] <ENTER>

  ================================================================================
||                                Interface setup                                 ||
  ================================================================================


================================================================================
||                                                                            ||
||                          FALLBACK interface setup                          ||
||                                                                            ||
================================================================================


========================== Setting up Trial interface ==========================

================================================================================
||                                                                            ||
||                          ADAPTIVE interface setup                          ||
||                                                                            ||
================================================================================


Do you have an ADAPTIVE.resources file? [False] yes
Specify path to ADAPTIVE.resources [ADAPTIVE.resources] (autocomplete enabled) ../ADAPTIVE.resources
Setting up interface init1
================================================================================
||                                                                            ||
||                           SPAINN interface setup                           ||
||                                                                            ||
================================================================================


Specify a path to a SPAINN template file.
Template path: (autocomplete enabled) ../SPAINN.template
Do you have a SPAINN.resources file? [False] <ENTER>
-------------------SPAINN ressource usage-------------------

Specify path to SPaiNN model:  (autocomplete enabled) ../spainn/runs/init1/best_model
Setting up interface init2
================================================================================
||                                                                            ||
```

```
||                          SPAINN interface setup                          ||
||                                                                          ||
==============================================================================


Specify a path to a SPAINN template file.
Template path: (autocomplete enabled) ../SPAINN.template
Do you have a SPAINN.resources file? [False] <ENTER>
------------------SPAINN ressource usage------------------

Specify path to SPaiNN model:  (autocomplete enabled) ../spainn/runs/init2/best_model
======================== Setting up Fallback interface =======================

==============================================================================
||                                                                          ||
||============================ASE_DB interface setup============================||
||                                                                          ||
==============================================================================



========================== Setting up child interface ==========================

==============================================================================
||                                                                          ||
||                           ORCA interface setup                           ||
||                                                                          ||
==============================================================================



Please specify path to ORCA directory
(SHELL variables and ~ can be used, will be expanded when interface is started).

Path to ORCA: (autocomplete enabled) $ORCADIR

---------------------Scratch directory----------------------

Please specify an appropriate scratch directory.
This will be used to run the ORCA calculations.
The scratch directory will be deleted after the calculation.
Remember that this script cannot check whether the path is valid,
since you may run the calculations on a different machine.
The path will not be expanded by this script.
Path to scratch directory: (autocomplete enabled) $TMPDIR

------------------ORCA input template file-----------------

Template filename: (autocomplete enabled) ../ORCA.template

Do you have a 'ORCA.resources' file? [False] yes
Specify the path: [ORCA.resources] (autocomplete enabled) ../ORCA.resources

  ============================================================================
||                                  PYSHARC                                   ||
  ============================================================================

The chosen interface can be run very efficiently with PYSHARC.
PYSHARC runs the SHARC dynamics directly within Python (with C and Fortran extension)
with minimal file I/O for maximum performance.
```

```
Setup for PYSHARC? [True] <ENTER>

  ================================================================================
||                          Content of output.dat files                           ||
  ================================================================================

SHARC or PYSHARC can produce output in ASCII format (all features supported currently)
or in NetCDF format (more efficient file I/O, trajectory restart currently not supported).
Write output in NetCDF format? [True] no

Do you want to write the gradients to the output.dat file ?
Write gradients? [False] <ENTER>

Do you want to write the non-adiabatic couplings (NACs) to the output.dat file ?
Write NACs? [False] <ENTER>

Do you want to write property matrices to the output.dat file  (e.g., Dyson norms)?
Write property matrices? [False] <ENTER>

Do you want to write property vectors to the output.dat file  (e.g., TheoDORE results)?
Write property vectors? [False] <ENTER>

Do you want to write the overlap matrix to the output.dat file ?
Write overlap matrix? [False] <ENTER>

Do you want to modify the output.dat writing stride?
Modify stride? [False] <ENTER>


  ================================================================================
||                                 Run mode setup                                 ||
  ================================================================================


-------------------------Run script------------------------

This script can generate the run scripts for each initial condition in two modes:

  - In mode 1, the calculation is run in subdirectories of the current directory.

  - In mode 2, the input files are transferred to another directory (e.g. a local scratch directory),
    the calculation is run there, results are copied back and the temporary directory is deleted.
    Note that this temporary directory is not the same as the "scratchdir" employed by the interfaces.

Note that in any case this script will create the input subdirectories in the current working directory.

In case of mode 1, the calculations will be run in:
/gpfs/data/sascha/sharc_tutorial/traj

Use mode 1 (i.e., calculate here)? [True] <ENTER>

--------------------Submission script---------------------

During the setup, a script for running all initial conditions sequentially in batch mode is generated.
Additionally, a queue submission script can be generated for all initial conditions.

Generate submission script? [False] <ENTER>


########################Full input#########################
```

```
select_directly          True
ninit                    50
natom                    21
repr                     MCH
diag                     False
eref                     -457.9790486105
eharm                    0.0
initf                    <_io.TextIOWrapper name='../initconds.excited' mode='r' encoding='UTF-8'>
states                   [3]
nstates                  3
charge                   [0]
statemap                 1: [1, 1, 0.0], 2: [1, 2, 0.0], 3: [1, 3, 0.0]
actstates                [3]
isactive                 [True, True, True]
show_content             False
n_issel                  [0, 0, 50]
setupstates              3
firstindex               1
ntraj                    10
needed_requests          'dm', 'h', 'grad'
method                   tsh
tmax                     1000.0
dtstep                   0.5
integrator               2
nsubstep                 25
kill                     False
surf                     diagonal
soc                      False
coupling                 3
phases_from_interface    False
gradcorrect              1
ekincorrect              2
reflect                  1
decoherence              ['edc', '0.1']
hopping                  sharc
force_hops               False
force_hops_dE            9999.0
scaling_for_sharc        False
damping                  False
atommaskarray            None
sel_g                    True
sel_t                    False
eselect                  0.001
rattle                   False
use_thermostat           False
laser                    False
dipolegrad               False
ion                      False
pysharc                  True
netcdf                   False
netcdf_separate          False
write_grad               False
write_NAC                False
write_property2d         False
write_property1d         False
write_overlap            False
stride                   [1]
log.infolevel            2
cwd                      /user/sharc/sharc_tutorial/traj
```

```
here                    True
copydir                 /user/sharc/sharc_tutorial/traj
qsub                    False

Do you want to setup the specified calculations? [True] <ENTER>

Do you want to link the interface files? [False] <ENTER>

  ================================================================================
||                          Setting up directories...                           ||
  ================================================================================


Progress: [=====                                              ]  10%

Do you want to see the input for the first trajectory? [False]
Do you want to add keywords to the input of all trajectories? [False]
Progress: [==================================================] 100%

10 trajectories setup, last initial condition was 10 in state 3.
```

After running, each trajectory should now have a file <PATH TO TRAJ>/QM/fallback_interface/adaptive_run1.db containing data points. Those databases can then be concatenated to the initial database. Train new models with the new database and repeat.

3.11 Setting up initial conditions for collisions

To set up an initial condition for collision dynamics, one requires one initial condition file for molecule + atom collision dynamics, and two initial condition files for molecule + molecule collision dynamics.

We start by an example of doing molecule + atom collision dynamics. Assume we have the following initial condition file, which has one sampled geometry for HI molecule:

```
SHARC Initial conditions file, version 2.0
Ninit     1
Natom     2
Repr      None
Temp            0.0000000000
Eref            0.0000000000
Eharm           0.0056103551

Equilibrium
 H   1.0   0.00000000   0.00000000   0.00000000   1.00782500   0.00000000   0.00000000   0.00000000
 I  53.0   0.00000000   0.00000000   3.36371300  126.90446800   0.00000000   0.00000000   0.00000000


Index     1
Atoms
 H   1.0   0.00000000   0.00000000   0.20953564   1.00782500   0.00000000   0.00000000  -0.00070329
 I  53.0   0.00000000   0.00000000   3.36204895  126.90446800   0.00000000   0.00000000   0.00000559
States
Ekin        0.000460810987 a.u.    0.012539310308 eV
Epot_harm   0.005149544152 a.u.    0.140126285822 eV
Epot        0.005149544152 a.u.    0.140126285822 eV
Etot_harm   0.005610355139 a.u.    0.152665596130 eV
Etot        0.005610355139 a.u.    0.152665596130 eV
```

One can construct a molecule + atom collision dynamics initial condition by the following example commend:

```
user@host> $SHARC/bimolecular_collision.py -n 1 --atom H initconds1
```

where **initconds1** is the initial condition file shown in the above panel. This commend uses the default settings of initial separation between the atom and the molecule, which is 10 Å; impact parameter is sampled according to a default setting, which is between 0 to 5 Å; and initial relative collision energy is set to 1 eV. Because only one initial condition file is used, the script automatically recognize that this is a molecule + atom initial condition. And the corresponding atom is H atom, as indicated by **--atom H**. One would see the following printing information:

See Section 7.3 (p. 182) in the manual.

```
Only one initial condition file is used, set up molecule + atom collision dynamics
==================================================
Initial Condition Sampling for MOLECULE + ATOM
==================================================
Involved atom is: H
Reading initconds1...
center of mass of MOLECULE 1 is moved to (0, 0, 0)
place atom at (impact_parameter, 0, initial_separation)
Deleted file: KEYSTROKES.bimolecular_collision
Random orient MOLECULE 1
Computed uniform velocity for molecule 2 atoms: 0.270052 bohr/(atomic_time)
Collision velocity added to molecule 2 atoms.
Molecule data combined successfully.
Combined data written to initconds.
```

And the resulting initial condition is saved in file **initconds** with the following contents:

```
Ninit     1.0
Natom     3.0
Repr      None
Temp      0.0000000000
Eref      0.0000000000
Eharm     0.0000000000

Equilibrium
H   1.0   0.00000000   0.00000000   0.00000000   1.00782500   0.00000000   0.00000000   0.00000000
I   53.0   0.00000000   0.00000000   3.36371300   126.90446800   0.00000000   0.00000000   0.00000000
H   1.0   3.72861171   0.00000000   18.89726125   1.00782500   0.00000000   0.00000000   0.00000000

Index     1
Atoms
H   1.0   -1.50619778   1.88074654   -1.99411856   1.00782500   0.00000000   0.00000000   -0.00070329
I   53.0   0.01196163   -0.01493614   0.01583650   126.90446800   0.00000000   0.00000000   0.00000559
H   1.0   3.72861171   0.00000000   18.89726125   1.00782500   0.00000000   0.00000000   -0.27005182
States
Ekin         0.037210133163 a.u.    1.012539305754 eV
Epot         0.005149544152 a.u.    0.140126234911 eV
Etot         0.042359677315 a.u.    1.152665540664 eV
```

One can, of course, uses non-default set up, for example:

```
user@host> $SHARC/bimolecular_collision.py -n 1 --separation 5.0 --bmin 0.0 --bmax 1.0 --atom H initconds1
```

which says the initial separation is 5.0 Å, and impact parameter is sampled between 0.0 to 1.0 Å.
And the resulting initial condition file may look:

```
Ninit     1.0
Natom     3.0
Repr      None
Temp      0.0000000000
Eref      0.0000000000
Eharm     0.0000000000

Equilibrium
H   1.0   0.00000000   0.00000000   0.00000000   1.00782500   0.00000000   0.00000000   0.00000000
I   53.0   0.00000000   0.00000000   3.36371300   126.90446800   0.00000000   0.00000000   0.00000000
H   1.0   1.45366204   0.00000000   9.44863062   1.00782500   0.00000000   0.00000000   0.00000000

Index     1
Atoms
H   1.0   -1.35931945   1.31049861   2.49342980   1.00782500   0.00000000   0.00000000   -0.00070329
I   53.0   0.01079518   -0.01040746   -0.01980183   126.90446800   0.00000000   0.00000000   0.00000559
H   1.0   1.45366204   0.00000000   9.44863062   1.00782500   0.00000000   0.00000000   -0.27005182
States
Ekin         0.037210133163 a.u.    1.012539305754 eV
Epot         0.005149544152 a.u.    0.140126234911 eV
Etot         0.042359677315 a.u.    1.152665540664 eV
```

Next, we provide an example of performing molecule + molecule initila condition sampling, which requires two initial condition files. We use one initial condition file of HI, which is already shown. And here is another initial condition file which corresponds to water moelcule:

```
SHARC Initial conditions file, version 2.0
Ninit     1
Natom     3
Repr      None
Temp            0.0000000000
Eref            0.0000000000
Eharm           0.0204394495

Equilibrium
 O   8.0   0.00000000   0.00000000   0.00000000  15.99491500   0.00000000   0.00000000   0.00000000
 H   1.0   0.00000000   0.00000000   1.79524000   1.00782500   0.00000000   0.00000000   0.00000000
 H   1.0   1.69257100   0.00000000  -0.59840700   1.00782500   0.00000000   0.00000000   0.00000000


Index     1
Atoms
 O   8.0   0.00000000   0.00515533  -0.00029786  15.99491500  -0.00003636   0.00000406  -0.00006761
 H   1.0   0.00000000   0.01467661   1.76189274   1.00782500   0.00073318  -0.00003900   0.00025911
 H   1.0   1.69257100  -0.09649539  -0.56033246   1.00782500  -0.00015612  -0.00002543   0.00081385
States
Ekin        0.008141409474 a.u.    0.221539118269 eV
Epot_harm   0.002691230986 a.u.    0.073232152444 eV
Epot        0.002691230986 a.u.    0.073232152444 eV
Etot_harm   0.010832640460 a.u.    0.294771270714 eV
Etot        0.010832640460 a.u.    0.294771270714 eV
```

The commend line of generating molecule + molecule initial condition file can be, for example:

```
user@host> $SHARC/bimolecular_collision.py -n 1 --separation 10.0 --bmin 0.0 --bmax 1.0 initconds1 initconds2
```

where **initconds2** is shown above. And this commend requires initial separation between HI and water molecule is 10.0 Å. One would observe the following printing information:

```
Two initial condition files are used, set up molecule + molecule collision dynamics
==================================================
Initial Condition Sampling for MOLECULE + MOLECULE
==================================================
Reading initconds1...
Reading initconds2...
center of mass of MOLECULE 1 is moved to (0, 0, 0)
center of mass of MOLECULE 2 is moved to (impact_parameter, 0, initial_separation)
Deleted file: KEYSTROKES.bimolecular_collision
Random orient MOLECULE 1
Computed uniform velocity for molecule 2 atoms: 0.063882 bohr/(atomic_time)
Collision velocity added to molecule 2 atoms.
Molecule data combined successfully.
Combined data written to initconds.
```

The resulting initial condition file looks like:

```
Ninit     1.0
Natom     5.0
Repr      None
Temp      0.0000000000
Eref      0.0000000000
Eharm     0.0000000000
```

```
Equilibrium
H   1.0   0.00000000   0.00000000   0.00000000   1.00782500   0.00000000   0.00000000   0.00000000
I   53.0   0.00000000   0.00000000   3.36371300   126.90446800   0.00000000   0.00000000   0.00000000
O   8.0   0.00000000   0.00000000   0.00000000   15.99491500   0.00000000   0.00000000   0.00000000
H   1.0   0.00000000   0.00000000   1.79524000   1.00782500   0.00000000   0.00000000   0.00000000
H   1.0   1.69257100   0.00000000   -0.59840700   1.00782500   0.00000000   0.00000000   0.00000000

Index     1
Atoms
H   1.0   2.65131719   -1.65877288   -0.03657637   1.00782500   0.00000000   0.00000000   -0.00070329
I   53.0   -0.02105571   0.01317332   0.00029048   126.90446800   0.00000000   0.00000000   0.00000559
O   8.0   0.92405522   0.00515533   9.38136106   15.99491500   -0.00003636   0.00000406   -0.06394924
H   1.0   0.92405522   0.01467661   11.14355166   1.00782500   0.00073318   -0.00003900   -0.06362252
H   1.0   2.61662622   -0.09649539   8.82132646   1.00782500   -0.00015612   -0.00002543   -0.06306778
States
Ekin         0.045351542637 a.u.    1.234078343537 eV
Epot         0.007840775138 a.u.    0.213358360748 eV
Etot         0.053192317775 a.u.    1.447436704285 eV
```

3.12 Exciting with the EOE or PDA schemes

In this section, we show how the advanced schemes to pick the initial electronic state are used. We use the SO_2 molecule, with frequencies from DFT in ORCA (for simplicity) and single points with CASSCF(6,4) in OpenMolcas (in order to also compute magnetic transition dipoles and electric transition quadrupoles).

3.12.1 Initial condition generation

To do a frequency calculation with ORCA, make a folder **freq** and create the input file **orca.inp**:

```
! bp86 def2-svp opt freq
* xyz 0 1
S 0 0 0
O 0 1.3 0.7
O 0 -1.3 0.7
*
```

Load the necessary ORCA environment, then run

```
user@host> orca orca.inp > orca.log
```

From the Hessian file, create the Molden file:

```
user@host> python $SHARC/ORCA_hess_freq.py orca.hess
```

From the Molden file, sample 20 Wigner snapshots:

```
user@host> python $SHARC/wigner.py orca.hess.molden -n 20
```

We need the **initconds** file for the next step.

See Section 6.10.4 (p. 98) in the manual.

3.12.2 Single point calculations with MOLCAS

In the subfolder **init**, create the **MOLCAS.template**:

```
basis cc-pvdz
ras2 4
nactel 6
inactive 13
roots 4  0  3
no-douglas-kroll
cholesky
method CASSCF
```

See Section 7.1 (p. 177) in the manual.

Set up the single point calculations as usual with:

```
user@host> python $SHARC/setup_init.py
```

- Load the **initconds** file, include all 20 initial conditions,
- Use **4 0 3** for states,
- Load the OpenMolcas interface (**SHARC_MOLCAS.py**),
- Request spin–orbit couplings as well as magnetic dipoles and electric quadrupoles,
- Set up OpenMolcas and run settings as usual.

See Section 7.9 (p. 188) in the manual.

Then, run all 21 folders (reference plus 20 initial conditions) via **all_run_init.sh**. Subsequently, each folder should contain a **QM.out** file.

Inspect the excitation energy for the S_1 state for the reference geometry:

```
user@host> python $SHARC/QMout_print.py ICOND_00000/QM.out
```

which should give:

```
Number of states: [4, 0, 3]
State        Label          E (E_h)      dE (eV)        f_osc     Spin
    1          S00  -547.1912068181   0.00000000   0.00000000   1.0000 #initial state
    2          S01  -547.0672249966   3.37371723   0.00388334   1.0000
    3          S02  -547.0600541145   3.56884687   0.00000000   1.0000
    4          S03  -546.9628004540   6.21525379   0.00001628   1.0000
    5          T01  -547.1015065819   2.44086777   0.00000000   3.0000
    6          T02  -547.0793024522   3.04507292   0.00000000   3.0000
    7          T03  -547.0669708801   3.38063209   0.00000000   3.0000
```

Hence, the desired excitation energy is 3.37 eV (0.1238 Hartree).

Also run

```
user@host> python $SHARC/excite.py
```

to collect the excited-state information and write the file **initconds.excited**. This is needed for the PDA method and can also be used as input for the EOE scheme.

3.12.3 Prepare the laser file

To use the EOE method, you need to create a file with the laser's electric (and magnetic) fields for each time step.

To plan the laser file, one should first decide on the time step that one plans to use in the eventual surface hopping simulations. It is very advantageous if the EOE simulations use the same time step, and that means that the time steps in the laser file need to use an integer fraction of the time step. In this example, we plan for surface hopping with 0.5 fs time steps. The laser (with energy of 3.37 eV) should use a smaller time step on the order of 0.01 fs (to avoid undersampling the oscillations of the field), so we will use 25 substeps, for a step of 0.02 fs. The laser should have a a 5 fs FWHM and a $\sin^2$ envelope, so we set the EOE simulations for 20 fs, as shown below. The $\sin^2$ envelope is not strictly realistic, but is advantageous because the field strength becomes actually zero.

Start **laser.x** and create the laser file:

```
user@host> $SHARC/laser.x
```

See Section 7.12 (p. 196) in the manual.

Use the following settings:

- Use 1 laser,
- Write out electric and magnetic field,
- Use real fields,
- Set the simulation from 0 fs to 20 fs with 1001 steps (yields 0.02 fs steps),
- Don't write debug files,
- Use **0 0 1** for the electric polarization (the symmetry axis of SO_2 above),
- Use **0 1 0** for the magnetic polarization (the O–O direction),
- Use a sinusoidal envelope,
- Field strength of 1 GV/m (the field strength does not really matter for EOE due to renormalization),
- Set the envelope as **0 5 10 10 15** (field is zero until 5 fs, then grows to maximum at 10 fs, then goes to zero at 15 fs).
- Use 3.37 eV as the frequency,
- Set the phase to 0,
- Do not use any of the chirps (set all values to 0).

This creates a file called **laser**.

3.12.4 Setting up, running, and analyzing the EOE simulations

Make another folder called **eoe/** and start the setup up script:

```
user@host> $SHARC/setup_laser_excitation.py
```

See Section 7.13 (p. 197) in the manual.

Use the following inputs:

- Give the path to the **init/** folder, choose either linking or copying,
- Give any seed,
- Give the path to the **initconds.excited** file (the **initconds** file works, but does not contain the reference energy),
- Setup the simulations from state 1,
- Give the path to the **laser** file,
- Give 0.5 fs as the desired dynamics step, then accept the suggested number of substeps (25),
- Select isotropic orientation distribution,

- Choose the MCH representation (select False),
- Set up the remaining options as needed.

This will produce a folder **Singlet_0/** with the input for 20 EOE simulations.

You can run them all with

```
user@host> sh all_run_traj.sh
```

These trajectories should be very fast (about 1 second per simulation).

Finally, you can perform the actual EOE initial state selection.

```
user@host> $SHARC/excite_laser_excitation.py
```

Use the following inputs:

- Give any seed,
- Use renormalization of 1.0,
- Do not use smoothing,
- Choose the default number of allowed hops (99999, effectively infinite).

This will run the EOE algorithm and write the file **initconds_Singlet_0.excited**. This file can be used as any other **initconds.excited** file to set up trajectories. Note that this file contains start times for each selected initial state, which **setup_traj.py** will write into each trajectory's folder as file **start.time**. This file is considered by some of the analysis tools. You can use **shift_output_lis.py** to create a properly shifted **output.lis**. **population.py**, **geo.py**, **geo_NM.py**, **data_extractor.x**, and **data_extractor_NetCDF.x** automatically detect the **start.time** file and shift the created results accordingly.

See Section 7.14 (p. 198) in the manual.

See Section 7.30 (p. 218) in the manual.

3.12.5 Exciting with the PDA approach

Instead of using the EOE scheme, you can also use the PDA scheme, if **promdens** is installed.

Prepare the **initconds.excited** file as described above, then

```
user@host> $SHARC/excite_with_promdens.py initconds.excited --dt 0.5
                                         --omega 0.1238 --fwhm 5. --env sin2 --t0 20. -neg ignore -np 10
```

This creates the file **initconds.excited.excited**. As with EOE, the initconds file produced by PDA contains start times.

See Section 7.15 (p. 199) in the manual.

3.13 Preparing initial conditions with VASP

In this section we show how to generate initial conditions (**initconds** file) out of VASP calculations to setup SHARC-VASP surface hopping dynamics.

Initial conditions sampling can be performed either by using Wigner sampling or by sampling snapshots from a ground state molecular dynamics trajectory. In both examples below, the sampling is performed targeting a minimal silicon unit cell in thermal equilibrium at 300 K. Note that such small cell size is considered only for exemplary purposes. To obtain meaningful results on real target systems convergence with respect to (super)cell size should be always considered.

See Section 6.13 (p. 102) in the manual.

3.13.1 Wigner sampling with VASP

See Section 6.28 (p. 142) in the manual.

In order to perform Wigner sampling with VASP first a frequency (phonon) calculation must be performed. To do so generate a directory called **phonons/** and create there the following input files:

POSCAR:

```
Si8
   1.00
     5.4437023729394527    0.0000000000000000    0.0000000000000003
    -0.0000000000000003    5.4437023729394527    0.0000000000000003
     0.0000000000000000    0.0000000000000000    5.4437023729394527
  Si
     8
Direct
     0.7500000000000000    0.7500000000000000    0.2500000000000000
     0.0000000000000000    0.5000000000000000    0.5000000000000000
     0.7500000000000000    0.2500000000000000    0.7500000000000000
     0.0000000000000000    0.0000000000000000    0.0000000000000000
     0.2500000000000000    0.7500000000000000    0.7500000000000000
     0.5000000000000000    0.5000000000000000    0.0000000000000000
     0.2500000000000000    0.2500000000000000    0.2500000000000000
     0.5000000000000000    0.0000000000000000    0.5000000000000000
```

INCAR:

```
SYSTEM = bulk Si
ISPIN =  1
ISMEAR=0
SIGMA=0.1
ENCUT=350
GGA = PE
PREC=Accurate
IBRION=5
NFREE=2
EDIFF=1e-7
EDIFFG=-1e-3
POTIM=0.015
```

KPOINTS:

```
Gamma-point only
 1        ! one k-point
rec       ! in units of the reciprocal lattice vector
 0 0 0 1  ! 3 coordinates and weight
```

POTCAR:

The POTCAR file containing the Si PAW pseudopotentials is bound to the user VASP installation directory. If, for instance, we define "vasp_installation_directory" as the path to the VASP installation directory containing the VASP executables in **bin/** and the pseudopotentials in **potcar/**,then issue the following command to get the proper POTCAR file you need

```
user@host> cat vasp_installation_directory/potcar/PBE/Si/POTCAR > POTCAR
```

Once the 4 input files are prepared (**INCAR,POSCAR,KPOINTS,POTCAR**), the phonon calculations can be started as

```
user@host> mpirun -np 2 vasp_gam > vasp.out
```

assuming to use 2 cpus in the example above.

From the output file (**OUTCAR**) of the VASP phonons calculation generate the Molden file:

```
user@host> python $SHARC/vasp_to_molden.py -p POSCAR -o OUTCAR
```

This script generates a **vasp.molden** file that can be used by **wigner.py** to sample structures from the Wigner distribution.

From the Molden file, sample 100 Wigner snapshots at 300 K by doing:

```
user@host> python $SHARC/wigner.py vasp.molden -t 300.0 -T --VASP_masses -n 100
```

This step generates the **initconds** file containing 100 initial conditions (positions and momenta) sampled from the system's Wigner distribution at 300 K. This file can then be processed by **setup_init.py** to setup the corresponding single point calculations for each sampled structure following the standard SHARC workflow.

See Section 6.13.4 (p. 104) in the manual.

See Section 7.1 (p. 177) in the manual.

See Section 7.9 (p. 188) in the manual.

3.13.2 Ground-state ab initio molecular dynamics sampling with VASP

Alternatively to Wigner sampling, ground-state ab initio molecular dynamics can be used to generate an ensemble of structures for preparing initial conditions.

In molecular dynamics sampling, the system is first heated to the target temperature. To do so create a directory called **heating/** and copy there the same files (**POSCAR, KPOINTS, POTCAR**) as above. For the **INCAR**, use the following settings: **INCAR**:

```
# DFT related
GGA = PE
EDIFF = 1e-5
ISMEAR = 0
SIGMA = 0.1
ENCUT=350
# MD related --------------------------
LWAVE = .FALSE.     # do not write the WAVECAR file (saves disk space)
LCHARG = .FALSE.    # do not write the CHGCAR file (saves disk space)
ISIF = 2            # NVT
IBRION = 0          # use molecular dynamics instead of ionic relaxation
POTIM = 1.0         # time step for MD in femtoseconds (fs)
MDALGO = 3          # choose Langevin dynamics algorithm for MD
LANGEVIN_GAMMA = 10.0  # friction coefficient (damping strength) for Langevin
TEBEG = 10           # initial temperature of the MD simulation (K)
TEEND = 300         # final temperature (K);
NSW = 1000        # maximum number of MD steps
ISYM=0
```

This performs a 1 ps ab initio MD run to heat up the silicon cell up to 300 K assuming NVT conditions. Run the calculation as

```
user@host> mpirun -np 4 vasp_gam > vasp.out
```

Once the calculation is finished, an equilibration stage of 3 ps has to be performed at the target temperature of 300 K. To do so create a new directory called **equilibration/** and copy there the **KPOINTS,POTCAR** files used for the heating run. As for the **INCAR** now use:

```
# DFT related
GGA = PE
EDIFF = 1e-5
ISMEAR = 0
SIGMA = 0.1
```

```
ENCUT=350
# MD related --------------------------
LWAVE = .FALSE.     # do not write the WAVECAR file (saves disk space)
LCHARG = .FALSE.    # do not write the CHGCAR file (saves disk space)
ISIF = 2            # NVT
IBRION = 0          # use molecular dynamics instead of ionic relaxation
POTIM = 1.0         # time step for MD in femtoseconds (fs)
MDALGO = 3          # choose Langevin dynamics algorithm for MD
LANGEVIN_GAMMA = 10.0  # friction coefficient (damping strength) for Langevin
TEBEG = 300           # initial temperature of the MD simulation (K)
TEEND = 300         # final temperature (K);
NSW = 3000        # maximum number of MD steps
ISYM=0
```

where **TEBEG=TEEND=300 K** ensures that an equilibration run is performed at the constant temperature of 300 K. The equilibration run must be restarted from the last frame of the heating run. To do so take the last frame of the heating trajectory (**heating/CONTCAR**) and rename it to **POSCAR** inside the **equilibration/** folder,

```
user@host> cp heating/CONTCAR equilibration/POSCAR
```

Then perform the equilibration run inside the folder **equilibration/** as

```
user@host> mpirun -np 4 vasp_gam > vasp.out
```

Once the equilibration run is finished, a final production stage of 5 ps has to be performed at the target temperature of 300 K. This is the final trajectory that will be used to sample initial conditions. To do so create a new directory called **production/** and copy there the **KPOINTS,POTCAR** files used for the equilibration run. Create there the following **INCAR**

```
# DFT related
GGA = PE
EDIFF = 1e-5
ISMEAR = 0
SIGMA = 0.1
ENCUT=350
# MD related --------------------------
LWAVE = .FALSE.     # do not write the WAVECAR file (saves disk space)
LCHARG = .FALSE.    # do not write the CHGCAR file (saves disk space)
ISIF = 2            # NVT
IBRION = 0          # use molecular dynamics instead of ionic relaxation
POTIM = 1.0         # time step for MD in femtoseconds (fs)
MDALGO = 3          # choose Langevin dynamics algorithm for MD
LANGEVIN_GAMMA = 10.0  # friction coefficient (damping strength) for Langevin
TEBEG = 300           # initial temperature of the MD simulation (K)
TEEND = 300         # final temperature (K);
NSW = 5000        # maximum number of MD steps
ISYM=0
VELOCITY=.true.
```

Note that in the new **INCAR** there is now an additional line (**VELOCITY=.true.**). This is necessary to make VASP write out atomic velocities as well, which is required to properly generate initial conditions. Subsequently take the last frame of the equilibration trajectory and copy it into the **production/** folder as starting structure,

```
user@host> cp equilibration/CONTCAR production/POSCAR
```

Then submit the production run as

```
user@host> mpirun -np 4 vasp_gam > vasp.out
```

The production trajectory can be finally used to sample different snapshots to generate initial conditions. To do so create a new directory named **initial_conditions**, go to that directory and then issue the following command:

```
user@host> python $SHARC/vasp_to_initconds.py ../production/ ../heating/POSCAR  -n 10 --every -x
```

This command will read the whole trajectory from the production run, read the equilibrium structure that we used to start the heating step (only to have a meaningful reference structure for ICOND_00000) and will sample 10 snapshots out of the production run,

sampling one snapshot every 5000 fs/10 = 500 fs. Note that **vasp_to_initconds.py** only works if you have installed in your conda's environment both **mdtraj** and **py4vasp** python packages. **vasp_to_initconds.py** will generate the **initconds** file containing 10 initial conditions (positions and momenta) sampled from an NVT molecular dynamics production run at 300 K. This file can then be processed by **setup_init.py** to setup the corresponding single point calculations for each sampled structure following the standard SHARC workflow.

See Section 7.9 (p. 188) in the manual.

4 Usage of the Interfaces

Within SHARC, the quantum chemistry calculations are always performed by the quantum chemistry interfaces, which provide a unified way to carry out the quantum chemistry independent of the used program. Hence, in principal every part of SHARC is compatible with the different quantum chemistry programs (with some exceptions, e.g., GAUSSIAN cannot provide spin-orbit couplings). However, depending on the used program, some details—mostly related to preparation of the frequency calculation and of the template files—of the preparation steps change. In the following, these details are addressed, in the order in which the interfaces are discussed in the Manual.

See Section 6.0.1 (p. 76) in the manual.

4.1 Do Nothing interface

This interface is for testing purposes only. It returns only zeros and cannot be used for actual research projects.

See Section 6.1 (p. 82) in the manual.

4.2 QMout interface

This interface is intended for frozen-nuclei dynamics, see Sections 3.9 and 3.12. Note that this interface cannot be used standalone. It always depends on another (usually ab initio) interface to produce a set of initial conditions and a **`QM.out`** file for each initial condition.

See Section 6.2 (p. 82) in the manual.

4.3 Analytical expressions

For the interface using analytical expressions for the potential energy surfaces, only one input file is needed (**`Analytical.template`**). This file contains the definitions of all analytical expressions. See below for an example of this file.

The remaining procedure with the analytical potential interface is analogous to the usage of the other interfaces. Note that there is no general workflow to setup initial conditions for this interface.

See Section 6.3 (p. 83) in the manual.

4.3.1 One-dimensional case

In the following, we will prepare the input for a single particle moving in one dimension on two states.

The file **`Analytical.template`** consists of a file header and a file body. The header for the one-dimensional case and for two states could look as follows:

```
1
2
H       x       0       0
```

The first line defines the number of atoms to be one, the second line gives the number of states. The third line is a mapping of the cartesian coordinates of the atom to variable names. In this case, the x component of the coordinate of the atom is linked to the actual variable called **`x`**. The variable can then be used in the file body in the definitions of the potentials.

Below the file header (with $n_{\text{atom}} + 2$ lines), different blocks can be put. Variable blocks define can be used to define constants:

```
Variables
k 2.0
D12 10.0
Re 2.0
omega 13.
mu 1.0
End
```

Most importantly, matrix blocks define the potential energies, couplings and gradients. A matrix block could look like:

```
Hamiltonian
0.5*k*x**2+omega;
0;   0.5*k*(x-Re)**2+D12,
```

Note the keyword **Hamiltonian**, which defines the type of matrix given. In SHARC4, derivatives are automatically obtained and do not need to be coded in the template file.

Put together, the complete input for the example might look like:

```
1
2
H       x       0       0

Variables
k 2.0
D12 10.0
Re 2.0
mu 1.0
End

Hamiltonian
0.5*k*x**2,
0,   0.5*k*(x-Re)**2+D12,

Dipole  1
0.0,
0.5*mu, 0.0,
```

Only the **Hamiltonian** block needs to be present. Optional blocks are **SpinOrbit** and **Dipole**.

For more details please refer to the manual.

4.4 LVC Models

The linear-vibronic coupling model interface uses analytical expressions (like the analytical interface). However, the LVC interface allows more complicated, realistic models for many-atom molecules, which can be automatically parametrized from a single point calculation.

The parametrization of LVC models is described in the specialized tutorial in section 3.6. Running LVC SHARC trajectories efficiently with **pysharc** is described in the specialized tutorial in section 3.7.

See Section 6.4 (p. 86) in the manual.

4.5 SPaiNN Models

SPaiNN models are machine learned potentials. This means, that you first need to train a model, which provides all the necessary outputs, such as energies for all the states, gradients/forces for these states and the couplings between these states if necessary. Alternatively, an already trained model can be used if it behaves the same way. An example is shown in Section 3.10.

See Section 6.4.4 (p. 88) in the manual.

4.6 SchNarc Models

SchNarc models are machine learned potentials. This means, that you first need to train a model, which provides all the necessary outputs, such as energies for all the states, gradients/forces for these states and the couplings between these states if necessary. Alternatively, an already trained model can be used if it behaves the same way. The usage of SchNarc models is relatively analogous to the usage of SPaiNN modes as shown in Section 3.10.

See Section 6.5 (p. 90) in the manual.

See Section 6.6 (p. 91) in the

4.7 OpenMM force fields

See Section 6.7 (p. 92) in the manual.

The OpenMM interface allows to use Amber force field files in SHARC. However, since force fields are not made for excited states, there are severe restrictions on what can be done with the OpenMM interface. Typically, it is only used in an QM/MM context. An example of its usage is given in the specialized tutorial in Section 3.8.

General remarks:

- There is no simply way to do a frequency calculation with an Amber force field, so there is no way to obtain a **molden** file and compute a Wigner distribution.
- Initial conditions can be generated with Amber, see Section 3.8.
- Likewise, the necessary **prmtop** files should be generated with Amber.
- The OpenMM interface, if used alone, will only permit to compute a single singlet state. In this case, you have to select wave function overlaps or curvature-driven dynamics in SHARC, even though electronic couplings are actually not relevant.

4.8 GAUSSIAN

See Section 6.9 (p. 94) in the manual.

Using GAUSSIAN in combination with SHARC requires some special attention, because TD-DFT is a single-reference method. This means that TD-DFT is comparably easy to use—there is no active space and no state-averaging (so adding more states to the computation does not affect the lower states). On the contrary, TD-DFT will often fail to converge when the energy gap between the ground state and the excited state becomes small. Another difference is that the ground state and the excited states are not treated equally—excited-state gradients are more expensive than the ground state gradient, and no transition dipole moments between two excited states can be computed. Yet another important difference is that spin-orbit couplings can not be computed with the SHARC-GAUSSIAN interface currently.

These are the main differences when using GAUSSIAN instead of MOLCAS:

- GAUSSIAN calculations—single points, optimizations, frequency calculations—can be setup with GAUSSIAN's GUI program, **gaussview**, or manually.
- In order to convert the result of a GAUSSIAN frequency calculations to MOLDEN format, run the frequency calculation with the **freq=hpmodes** option and use **GAUSSIAN_freq.py**.
- Template files for the SHARC-GAUSSIAN interface should be created by copying and adjusting the documented example in **$SHARC/../examples/SHARC_GAUSSIAN/**.
- GAUSSIAN cannot compute nonadiabatic coupling vectors, so all SHARC-GAUSSIAN simulations rely on **wfoverlap.x** or have to use curvature-driven dynamics.
- The SHARC-GAUSSIAN interface supports QM/MM via the **SHARC_QMMM.py** hybrid interface.

Because there are relevant S_1/S_0 conical intersections, the dynamics of $CH_2NH_2^+$ cannot be properly described with TD-DFT. Users who intend to follow the tutorial with GAUSSIAN instead of MOLCAS should therefore use a different molecule (SO_2 might work well). Alternatively, the **force_hops_to_gs** option can be used to enable an approximate decay from the S_1 to the S_0 of $CH_2NH_2^+$.

4.9 ORCA

See Section 6.10 (p. 97) in the manual.

Using ORCA in combination with SHARC requires some special attention, because TD-DFT is a single-reference method. This means that TD-DFT is comparably easy to use—there is no active space and no state-averaging (so adding more states to the computation does not affect the lower states). On the contrary, TD-DFT will often fail to converge when the energy gap between the ground state and the excited state becomes small. Another difference is that the ground state and the excited states are not treated equally—excited-state gradients are more expensive than the ground state gradient, and no transition dipole moments between two excited states can be computed.

These are the main differences when using ORCA instead of MOLCAS:

- ORCA calculations—single points, optimizations, frequency calculations—can be setup with different GUI programs, e.g., **Gabedit**, or manually.
- The result of ORCA frequency calculations can be converted to MOLDEN format by applying **ORCA_hess_freq.py** to the **.hess** file from the frequency calculation.
- Template files for the SHARC-ORCA interface should be created by copying and adjusting the documented example in **$SHARC/../examples/SHARC_ORCA/**.
- ORCA cannot compute nonadiabatic coupling vectors for TD-DFT, so all SHARC-ORCA simulations rely on **wfoverlap.x** or have to use curvature-driven dynamics.
- The SHARC-ORCA interface supports QM/MM via the **SHARC_QMMM.py** hybrid interface.

For the reasons mentioned above, the dynamics of $CH_2NH_2^+$ cannot be properly described with TD-DFT. Users who intend to follow the tutorial with ORCA instead of MOLCAS should therefore use a different molecule (SO_2 might work well). Alternatively, the **force_hops_to_gs** option can be used to enable an approximate decay from the S_1 to the S_0 of $CH_2NH_2^+$.

4.10 NWCHEM

Using NWCHEM in combination with SHARC requires some special attention, because TD-DFT is a single-reference method. This means that TD-DFT is comparably easy to use—there is no active space and no state-averaging (so adding more states to the computation does not affect the lower states). On the contrary, TD-DFT will often fail to converge when the energy gap between the ground state and the excited state becomes small. Another difference is that the ground state and the excited states are not treated equally—excited-state gradients are more expensive than the ground state gradient, and no transition dipole moments between two excited states can be computed.

See Section 6.12 (p. 100) in the manual.

These are the main differences when using NWCHEM instead of MOLCAS:

- The SHARC package currently does not provide tools to generate input for NWCHEM calculations. Please see the NWCHEM documentation.
- NWCHEM frequency calculations produce a MOLDEN file that can be used with **`wigner.py`**.
- Template files for the SHARC-NWCHEM interface should be created by copying and adjusting the documented example in **`$SHARC/../examples/SHARC_NWCHEM/`**.
- NWCHEM cannot compute nonadiabatic coupling vectors for TD-DFT, so all SHARC-NWCHEM simulations rely on **`wfoverlap.x`** or have to use curvature-driven dynamics.
- The SHARC-NWCHEM interface does not support QM/MM

For the reasons mentioned above, the dynamics of $CH_2NH_2^+$ cannot be properly described with TD-DFT. Users who intend to follow the tutorial with NWCHEM instead of MOLCAS should therefore use a different molecule (SO_2 might work well). Alternatively, the **`force_hops_to_gs`** option can be used to enable an approximate decay from the S_1 to the S_0 of $CH_2NH_2^+$.

4.11 TURBOMOLE

The SHARC-TURBOMOLE interface can carry out ADC(2) and CC2 calculations. Note that the interface does not allow using TURBOMOLE's TD-DFT functionality (for TD-DFT, one can use ADF or GAUSSIAN instead). ADC(2) and CC2 are single-reference methods. This means that they are comparably easy to use—there is no active space and no state-averaging (so adding more states to the computation does not affect the lower states). On the contrary, ADC(2) and CC2 will often fail to converge when the energy gap between the ground state and the excited state becomes small. Another difference is that the ground state and the excited states are not treated equally—excited-state gradients are more expensive than the ground state gradient, and no transition dipole moments between two excited states can be computed.

See Section 6.14 (p. 105) in the manual.

Also note that within SHARC, spin-orbit couplings are only available with ADC(2), but not with CC2.

These are the main differences when using ADC(2) in TURBOMOLE instead of MOLCAS:

- TURBOMOLE calculations—single points, optimizations, frequency calculations—can be setup with TURBOMOLE's input facility **`define`**.
- (For MP2 frequencies with **`ricc2`**, use **`NumForce -ri -level mp2`**, instead of the **`NumForce -level CC2`** claimed in the TURBOMOLE documentation.)
- The result of TURBOMOLE frequency calculations can be converted to MOLDEN format using TURBOMOLE's **`tm2molden`** program.
- Template files for the SHARC-TURBOMOLE interface should be created by copying and adjusting the documented example in **`$SHARC/../examples/SHARC_RICC2/`**.
- TURBOMOLE cannot compute nonadiabatic coupling vectors, so all SHARC-TURBOMOLE simulations rely on **`wfoverlap.x`** or have to use curvature-driven dynamics.
- You can set **`$TURBODIR`** to the main path of TURBOMOLE in your Shell profile. SHARC scripts will recognize this variable, which will simplify the setup.
- The interface supports QM/MM via the **`SHARC_QMMM.py`** hybrid interface.

For the reasons mentioned above, the dynamics of $CH_2NH_2^+$ cannot be properly described with ADC(2) or CC2. Users who intend to follow the tutorial with TURBOMOLE instead of MOLCAS should therefore use a different molecule (SO_2 might work well). Alternatively, the **`force_hops_to_gs`** option can be used to enable an approximate decay from the S_1 to the S_0 of $CH_2NH_2^+$.

4.12 MOLCAS

MOLCAS is the program used in the full tutorial in Chapter 2.

See Section 6.15 (p. 107) in the manual.

4.13 MNDO

The SHARC-MNDO interface can carry out MRCI calculations using the OMx and ODMx, Hamiltonians. MRCI is a multi-reference method. This means that this method is comparably complicated to use since there is an active space. OM2/ODM2 will often fail to

See Section 6.16 (p. 111) in the manual.

converge when the energy gap between the ground state and the excited state becomes small. Transition dipole moments between two excited states can be computed. Non-adibatic couplings and overlaps between two states can be computed, but spin-orbit couplings cannot be computed by MNDO.

These are the main differences when using semiempirical Hamiltonians in MNDO instead of MOLCAS:

- MNDO cannot do frequency calculations, these have to be provided another way.
- Template files for the SHARC-MNDO interface should be created by copying and adjusting the documented example in **`$SHARC/../examples/SHARC_MNDO/`**.
- MNDO cannot compute spin-orbit couplings, so only singlet states can be treated.
- The interface supports QM/MM via the **`SHARC_QMMM.py`** hybrid interface.

4.14 MOPAC-PI

The SHARC-MOPA-PI interface can carry out FOMO-CI calculations using the AM1, PM3, PM6 and MNDO Hamiltonians. FOMO-CI is a multi-reference method. This means that this method is comparably complicated to use since there is an active space. Transition dipole moments between two excited states can be computed. Non-adibatic couplings between two states can be computed but spin-orbit couplings cannot be computed by the SHARC-MOPA-PI interface.

See Section 6.17 (p. 113) in the manual.

These are the main differences when using semiempirical Hamiltonians in MOPAC-PI instead of MOLCAS:

- MOPAC-PI cannot do frequency calculations, these have to be provided another way.
- Template files for the SHARC-MOPAC-PI interface should be created by copying and adjusting the documented example in **`$SHARC/../examples/SHARC_MOPACPI/`**.
- MOPAC-PI cannot compute spin-orbit couplings, so only singlet states can be treated.
- The interface does not support QM/MM via the **`SHARC_QMMM.py`** hybrid interface. MOPAC-PI uses a preimplemented TINKER interface that is part of the MOPAC-PI software package. TINKER-specific files need to be provided.

4.15 Legacy interface

The **`SHARC_LEGACY.py`** interface is a frontend for five other interfaces, so that they can be used consistently in SHARC4. Hence, you do not use **`SHARC_LEGACY.py`** alone, but always to access one of these five interfaces. These are for AMS-ADF, COLUMBUS, BAGEL, MOLPRO, and PYSCF. Please see their interface specific tutorials in the next sections.

See Section 6.19 (p. 118) in the manual.

4.16 AMS ADF

Using AMS ADF in combination with SHARC requires some special attention, because TD-DFT is a single-reference method. This means that TD-DFT is comparably easy to use—there is no active space and no state-averaging (so adding more states to the computation does not affect the lower states). On the contrary, TD-DFT will often fail to converge when the energy gap between the ground state and the excited state becomes small. Another difference is that the ground state and the excited states are not treated equally—excited-state gradients are more expensive than the ground state gradient, and no transition dipole moments between two excited states can be computed. Yet another important difference is that spin-orbit couplings can only be computed between singlets and triplets (unlike in the multi-reference codes, where all multiplicities can be used).

See Section 6.20 (p. 119) in the manual.

These are the main differences when using ADF instead of MOLCAS:

- ADF calculations—single points, optimizations, frequency calculations—can be setup with ADF's suite of GUI programs.
- The result of ADF frequency calculations (standard output or **`TAPE21`**) can be converted to MOLDEN format using **`AMS_ADF_freq.py`**.
- Before starting ADF or any Python script related to ADF, source the relevant **`adfrc.sh`** file which came with ADF.
- Template files for the SHARC-ADF interface should be created by copying and adjusting the documented example in **`$SHARC/../examples/SHARC_ADF/`**.
- ADF cannot compute nonadiabatic coupling vectors, so all SHARC-ADF simulations rely on **`wfoverlap.x`** or have to use curvature-driven dynamics..
- The interface is not compatible with QM/MM calculations.

For the reasons mentioned above, the dynamics of $CH_2NH_2^+$ cannot be properly described with TD-DFT. Users who intend to follow the tutorial with ADF instead of MOLCAS should therefore use a different molecule (SO_2 might work well). Alternatively, the **`force_hops_to_gs`** option can be used to enable an approximate decay from the S_1 to the S_0 of $CH_2NH_2^+$.

4.17 COLUMBUS

COLUMBUS is a very complex suite of many independent programs, each with own input files which are strongly inter-dependent. Hence, SHARC does not come with an input preparation tool like **`molpro_input.py`** or **`molcas_input.py`**. Instead, users should use

See Section 6.21 (p. 123) in the manual.

COLUMBUS' interactive input facility, **colinp**. Users who are new to COLUMBUS should first work through the COLUMBUS main tutorial and the COLUMBUS online documentation before starting to work with the SHARC-COLUMBUS interface.

These are the main differences when using COLUMBUS instead of MOLCAS:

- Input preparation (both for regular COLUMBUS calculations and for the interface template) is relatively complicated. See the following subsections for some general hints.
- COLUMBUS has a number of hard internal limits: a maximum of 255 basis functions and a maximum of 65535 CSFs in MCSCF (CAS(16,12) or larger will not work).
- One can use either of two integral codes: DALTON or MOLCAS. The former can compute nonadiabatic coupling vectors, whereas the latter can compute spin-orbit couplings. For the latter, the COLUMBUS-MOLCAS interface must be installed.
- One can use either of two CASSCF codes: COLUMBUS own code or MOLCAS (faster, but no gradients). For the latter, the COLUMBUS-MOLCAS interface must be installed.
- If planning to run CASSCF-based SHARC dynamics, we recommend to not use COLUMBUS; instead, use MOLCAS, MOLPRO, or BAGEL. COLUMBUS should only be used for MRCI-based dynamics.
- MRCI-based dynamics can be very expensive. Careful setup is required.
- Running the **wfoverlap.x** program with MRCI wave functions can be quite expensive. Hence, users should carefully adjust the related wave function truncation threshold.

4.17.1 General Hints for using COLUMBUS

In order to use COLUMBUS, whether via its driver script **runc** or through the SHARC-COLUMBUS interface, you have to set the environment variable **$COLUMBUS** to the directory containing the COLUMBUS executables (like **runc**, **mcscf.x**, **ciudg.x**, etc.).

In order to prepare a calculation, first convert the standard xyz file containing the molecular geometry to COLUMBUS format:

```
user@host> $COLUMBUS/xyz2col.x < geom.xyz
```

which will create the file **geom**. Then, start **colinp**

```
user@host> $COLUMBUS/colinp
```

Go through the input sections, starting with the integral section, followed by SCF, CASSCF, MRCI and finally the run setup. Advanced users can then manually alter the input files as necessary (e.g., to achieve a RAS-type reference wave function). After the preparation of the input, start COLUMBUS via

```
user@host> $COLUMBUS/runc -m [MEMORY in MB] > runls&
```

An optimization can be carried out directly with **runc**—just use the corresponding options in **colinp**.

For a (numerical) frequency calculation, use **colinp** to generate the internal coordinates and the **DISPLACEMENT** directories. Then, a calculation needs to be carried out in each of the **DISPLACEMENT** subdirectories, a task which is accomplished by the **calc.pl** script, or by manually starting **runc** in each of the directories (this approach might be faster because the calculations can be parallelized). Subsequently, the script **forceconst.pl** can be used to collect the results, producing the file **suscalls** which is compatible with the MOLDEN format (and can be used with **wigner.py**).

4.17.2 COLUMBUS input for usage with the interface

For the SHARC-COLUMBUS interface, a template directory is needed. The directory needs to contain one subdirectory with input for each "job". Each job usually contains the input for one multiplicity; the exception is the computation of spin-orbit couplings, where several multiplicities are computed in the same job. In order to prepare the input for the SHARC-COLUMBUS interface with spin-orbit couplings, you can see the tutorial on spin-orbit coupling calculations.

Integral program input The interface is able to either use integrals from DALTON or from MOLCAS. With DALTON, nonadiabatic coupling vectors can be computed but no spin-orbit couplings, with MOLCAS it is vice versa; hence, choose the integral program with care.

This is a step-by-step description for the integral setup:

1. Run the preparation program (prepinp): yes
2. Choose either DALTON or MOLCAS
3. Choose the symmetry (input is different for DALTON—write **c1** for no symmetry—and MOLCAS—press **<ENTER>** for no symmetry)
4. Provide the geometry file in COLUMBUS format
5. Provide the basis set
6. For MOLCAS: include scalar relativity and spin-orbit integrals if desired

SCF input Since the SCF step is actually not carried out if starting orbitals are provided, it is sufficient to setup a closed-shell neutral wave function with default parameters.

MCSCF input In the MCSCF section any desired state-averaging scheme can be defined.

This is a step-by-step description for the MCSCF setup:

1. Do not freeze any orbitals at the MCSCF level.
2. If gradients are desired, always set up for CI gradients (even if doing MCSCF-level dynamics).
3. Enter the number of distinct row tables (DRTs); you will need one DRT for each multiplicity included in the state-averaging.
4. For each DRT, provide the MCSCF settings (number of electrons, multiplicity, symmetry (**1**), RAS settings (**0** and **0**).
5. Provide the number of active orbitals, do not apply group restrictions.
6. Set the convergence to very tight (this improves the computation of gradients), e.g., **knorm [1.e-6] wnorm [1.e-6] DE [1.e-10]**, and the number of states for state-averaging.

CI input There are three possible ways to setup the CI input. If you have only one multiplicity in this job, then use the **one-DRT case**. If you have several multiplicities in this job but do not want to compute spin-orbit couplings, use **independent multiple-DRTs**. If you have several multiplicities in this job and want to compute compute spin-orbit couplings, use the **one-DRT case**.

Make sure that all multiplicities used in the SHARC input are covered with all job directories.

This is a step-by-step description for the CI setup:

1. Press **<ENTER>** to leave the DRT explanation text
2. Choose **one-DRT case** or **independent multiple-DRTs** (as explained above)
3. Choose **y** if you want to compute gradients
4. If you want to compute spin-orbit couplings, enable **Spin-Orbit CI**, give the maximum multiplicity (e.g., **3**), and the spin representation (always **1 1 1**)
5. Complete the DRT input (number of electrons, symmetry (**1**), number of frozen core/virtual orbitals, internal orbitals, doubly-occupied, auxiliary, excitation level (CASSCF: **0**, MRCI: **1** or **2**), reference symmetry (**1**), no group restrictions)
6. Choose **ciudg** (not **pciudg**)
7. Select **CI** as type of calculation
8. Choose the number of roots (will be overridden by the interface, but choose at least **2** because the **colinp** sets up necessary input files) and **RTOL** for the CI procedure (**1e-4** or larger if performance is needed)
9. Select at least one transition moment (will be overridden by the interface)

Set up job control Most of the job control is overridden by the interface, but some options are needed to obtain all necessary input files.

This is a step-by-step description for the job setup:

1. Choose **Job control for single point or gradient calculation**
2. Choose **single point calculation**
3. Select: **SCF**, **MCSCF**, **one-electron properties for all methods**, **transition moments for MR-CISD**, **nonadiabatic couplings (and/or gradients)**
4. Also select either **MR-CISD (serial operation)** (no spin-orbit coupling) or **SO-CI coupled to non-rel. CI** (for spin-orbit coupling)
5. If **SO-CI coupled to non-rel. CI**, choose the number of states per multiplicity, e.g., **4:0:3** (will be overridden)
6. Choose **first moments**
7. Do not use **analysis in internal coordinates**
8. Choose **DCI*(E2-E1) term (interstate coupling)** (or **DCI+DCSF term (non-adiabatic coupling)** when using DALTON)
9. Do not use **intersection analysis (slope)**
10. Exit **colinp**

Repeat these steps for all input directories if you have more than one job.

4.18 BAGEL

The functionality of the SHARC-BAGEL interface is very similar to the SHARC-MOLCAS interface (but can also do dynamics with analytical gradients and nonadiabatic couplings for different flavors of CASPT2). Hence, users who went through the full tutorial in chapter 2 might feel familiar when using BAGEL instead.

See Section 6.22 (p. 126) in the manual.

These are the main differences when using BAGEL instead of MOLCAS:

- Currently, there is no automatized way to setup BAGEL calculations with SHARC tools. Please refer to the BAGEL manual for doing preparatory calculations and frequency jobs.

- BAGEL cannot state-average over different multiplicities.
- BAGEL can compute nonadiabatic coupling vectors. These can be used in **sharc.x** for electronic propagation and for the exact transformation of the gradient vectors. Related options can be set during the **setup_traj.py** dialogue.
- Unlike the MOLCAS interface, the BAGEL interface cannot perform QM/MM calculations.
- Overlap calculations do not require **pyquante** as in SHARC3, but instead only that the **pyscf** Python module is installed.

4.19 MOLPRO

The functionality of the SHARC-MOLPRO interface is very similar to the SHARC-MOLCAS interface. Hence, users who went through the full tutorial in chapter 2 should feel familiar when using MOLPRO instead.

See Section 6.23 (p. 129) in the manual.

These are the main differences when using MOLPRO instead of MOLCAS:

- MOLPRO calculations—single point, optimization, frequencies—can be setup with **molpro_input.py**. This also allows preparing MOLDEN files to be used with **wigner.py**.
- **molpro_input.py** can write **MOLPRO.template** files.
- MOLPRO can state-average over different multiplicities. Therefore, in **MOLPRO.template** one can define multiple independent CASSCF jobs, each with arbitrary state-averaging schemes. See the example files in **$SHARC/../examples/SHARC_MOLPRO/** for such a template file.
- MOLPRO can compute nonadiabatic coupling vectors. These can be used in **sharc.x** for electronic propagation and for the exact transformation of the gradient vectors. Related options can be set during the **setup_traj.py** dialogue. **Note**, however, that MOLPRO sometimes randomly changes the sign of nonadiabatic coupling vectors in a way that cannot be corrected, so propagating with **coupling nacdr** is discouraged (use **coupling overlap**).
- Unlike the MOLCAS interface, the MOLPRO interface cannot perform CASPT2 or QM/MM calculations.
- Only segmented basis sets (e.g., Pople or Ahlrichs) can be used for the SHARC-MOLPRO interface, while generally contracted basis sets (e.g., Dunning or ANO) cannot be used because MOLPRO cannot calculate gradients with them.
- The SHARC-MOLPRO interface almost always requires **wfoverlap.x**, so this should be installed from the SHARC homepage.
- You can set **$MOLPRO** to the main directory of MOLPRO in your Shell profile. SHARC scripts will generally notice that this variable is set, which will simplify the setup.

4.20 PySCF

The functionality of the SHARC-PySCF interface is currently relatively limited (only singlet states). Otherwise, as an interface for multi-reference methods, there are some similarities to the full tutorial in chapter 2 that uses MOLCAS.

See Section 6.24 (p. 134) in the manual.

These are the main differences when using PySCF instead of MOLCAS:

- Currently, there is no automatized way to setup PySCF calculations with SHARC tools. Please refer to the PySCF manual for doing preparatory calculations and frequency jobs.
- PySCF can compute nonadiabatic coupling vectors, but no wave function overlaps.
- Unlike the MOLCAS interface, the PySCF interface cannot perform QM/MM calculations.

4.21 ASE_DB Database interface

This is a hybrid interface, meaning that it calls another interface (its "child") to do the actual quantum-chemical computations. It cannot be used standalone and there are no possibilities to produce initial conditions. See the adaptive training tutorial for one use case of the ASE database interface (Section 3.10).

See Section 6.25 (p. 137) in the manual.

4.22 Classical Path Approximation (CPA) interface

This is a hybrid interface, meaning that it calls another interface (its "child") to do the actual quantum-chemical computations. It cannot be used standalone and there are no possibilities to produce initial conditions.

See Section 6.28 (p. 142) in the manual.

4.23 Umbrella sampling interface

This is a hybrid interface, meaning that it calls another interface (its "child") to do the actual quantum-chemical computations. It cannot be used standalone and there are no possibilities to produce initial conditions.

See Section 6.26 (p. 138) in the manual.

4.24 Numerical differentiation interface

This is a hybrid interface, meaning that it calls another interface (its "child") to do the actual quantum-chemical computations. It cannot be used standalone and there are no possibilities to produce initial conditions.

See Section 6.29 (p. 144) in the manual.

4.25 QM/MM interface

This is a hybrid interface, meaning that it calls other interfaces (its "children") to do the actual quantum-chemical computations. It cannot be used standalone and there are no possibilities to produce initial conditions. See the QM/MM tutorial for one use case of the QM/MM interface (Section 3.8).

See Section 6.30 (p. 147) in the manual.

4.26 ECI interface

This is a hybrid interface, meaning that it calls other interfaces (its "children") to do the actual quantum-chemical computations. It cannot be used standalone and there are no possibilities to produce initial conditions. The SHARC manual contains a comprehensive example in the documentation of the ECI interface.

See Section 6.31 (p. 149) in the manual.

4.27 Adaptive sampling interface

This is a hybrid interface, meaning that it calls other interfaces (its "children") to do the actual quantum-chemical computations. It cannot be used standalone and there are no possibilities to produce initial conditions. See the adaptive training tutorial for one use case of the adaptive sampling interface (Section 3.10).

See Section 6.32 (p. 161) in the manual.

4.28 Fallback interface

This is a hybrid interface, meaning that it calls other interfaces (its "children") to do the actual quantum-chemical computations. It cannot be used standalone and there are no possibilities to produce initial conditions. See the adaptive training tutorial for one use case of the fallback interface (Section 3.10).

See Section 6.33 (p. 163) in the manual.

5 Quick Copy-Paste Tutorial for a single Trajectory

This quick tutorial presents how to setup a single SHARC trajectory, by simply creating all necessary input files without employing the tools of the SHARC suite. In this quick tutorial, we are using the equilibrium geometry as starting geometry and random initial velocities.

The goal is to prepare all input files for SHARC and the MOLCAS interface. The necessary files and directories are presented in figure 5.1. Note that the files **`MOLCAS.template`** and **`MOLCAS.resources`** are only necessary since we are using the SHARC-MOLCAS interface.

The contents of the files are given and explained in the following. Please be aware that the two subdirectories need to exist before SHARC can be started.

5.1 Input File

The SHARC input file ("**`input`**") contains the dynamics settings and names of additional input files (geometry, velocity, coefficients). An example is given below:

```
geomfile "geom"
veloc random 0.1

nstates 4 0 3
state 3 mch
coeff auto

ezero -94.41294549
tmax 25.000000
stepsize 0.5

surf sharc
coupling overlap
decoherence_scheme edc

grad_select
eselect 0.1
select_directly
```

The meaning of these keywords is: The geometry is read from file **`geom`**. The nuclear velocities are picked at random, with 0.1 eV kinetic energy per atom. Four singlet states and three triplet states will be included in the simulation (0 is the number of doublet states). The initial state is the third state (the S_2). The initial coefficients will be set automatically (the initial state will have a coefficient of

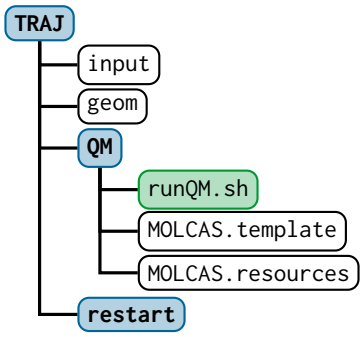

Figure 5.1: Input files for a SHARC dynamics simulation of a single, independent trajectory.

1.0, the remaining states a coefficient of zero). The diagonal elements of the Hamiltonian will be shifted by −94.41294549 Hartree. The simulation will run for 25 fs with a 0.5 fs timestep. The SHARC formalism will be used (propagation on the diagonalized states). Nonadiabatic interactions are described with wavefunction overlaps. SHARC will select which gradients to compute at each time step, with a selection threshold of 0.1 eV. SHARC will select these gradients directly, without doing two quantum chemistry calculations per time step.

5.2 Geometry File

The geometry file "**geom**" contains the chemical symbols, atomic charge, x, y and z coordinates and the relative atomic masses.

```
C 6.0 +0.00000000 +0.00000000 +0.00000000 12.00000000
N 7.0 +0.00000000 +0.00000000 +2.45664494 14.00307401
H 1.0 +1.78220520 +0.00000000 -1.02895628 1.00782504
H 1.0 +1.78220520 +0.00000000 +3.55174167 1.00782504
H 1.0 -1.78220520 +0.00000000 +3.55174167 1.00782504
H 1.0 -1.78220520 +0.00000000 -1.02895628 1.00782504
```

5.3 QM Run Script

At each timestep, SHARC writes the current geometry and different keywords to the file **QM/QM.in** and then calls **runQM.sh**. After this call is finished, SHARC reads the results of the quantum chemistry calculation from **QM.out**.

In most of the cases, in **runQM.sh** simply one of the SHARC-interfaces is called:

```
cd QM
$SHARC/SHARC_MOLCAS.py QM.in >> QM.log 2>> QM.err
```

The interface will do all work necessary to produce the desired file **QM.out**.

5.4 MOLCAS Template

The MOLCAS interface needs as additional input file giving the settings for the electronic structure calculation. The file is called "**MOLCAS.template**". It employs a simple keyword-argument structure and looks like:

```
basis cc-pVDZ
nactel 7
ras2 4
inactive 5
roots 4 0 3
method casscf
```

5.5 MOLCAS Resources

The MOLCAS interface additionally needs some paths and resource settings, both of which are read from the file "**MOLCAS.resources**".

```
molcas /usr/license/molcas/molcas80
scratchdir $TMPDIR/WORK
savedir ../restart/

memory 1000
ncpu 1
```

5 Quick Copy-Paste Tutorial for a single Trajectory

5.6 Running SHARC

With the input files prepared, the trajectory can then be started by simply executing:

```
$SHARC/sharc.x input
```

5.7 Output

SHARC produces four output files, **output.log**, **output.lis**, **output.dat** and **output.xyz**. The file **output.log** contains mainly a listing of the chosen options and the resulting dynamics settings. At higher print levels, the log file contains also information per timestep. **output.lis** contains a table with one line per timestep, giving active states, energies and expectation values. **output.dat** contains a list of all important matrices and vectors at each timestep. This information can be extracted with **data_extractor.x** to yield plottable table files. **output.xyz** contains the geometries of all timesteps, allowing visualization of the trajectory with the appropriate software (e.g., MOLDEN).