MOMAP

User Guide

Version 2021A

Version 2021A

April, 2021

MOMAP User Guide

Version 1.0 edited by:

Dr. Yingli Niu, Dr. Linjun Wang, Dr. Wenqiang Li, Mr. Shuo Zhang, Dr. Hua Geng, Dr. Qian Peng, and Dr. Jianming Chen

Version 2019B edited by:

Dr. Qikai Li, Dr. Yingli Niu, Dr. Linjun Wang, Dr. Qian Peng, and Ms. Lihui Yan

Versions 2020A, 2020B, 2021A edited by:

Dr. Qikai Li

Released by Hongzhiwei Technology (Shanghai) Co., Ltd

and Z.G. Shuai Group

The information in this document applies to MOMAP 2021A

Preface

This MOMAP User Guide is composed of two parts, that is, the part on photophysical properties and the part on transport properties, which reflects the functionalities of current MOMAP version 2021A.

The photophysical property part introduces the quantum chemistry calculations, vibrational analysis with electron-vibration coupling (EVC), and calculations on absorption spectrum, emission spectrum, radiative rate, internal conversion, intersystem crossing *etc*. by examples of the fluorescence spectrum and phosphorescence spectrum calculations. In addition, the Guide also touches upon the sum-over-states approach and Herzberg-Teller effect in calculations.

The transport property part introduces the procedure of charge carrier transport calculations by using naphthalene molecule as an example, with detailed descriptions on calculations of transfer integral, reorganization energy, random walk *etc*. In the meantime, the Guide also gives descriptions on data analysis *etc*. which facilitates the users to incorporate the calculations into their research work.

Many people are engaged in the compilation of this Guide, the main participants are: Prof. Qikai Li from Tsinghua University, Dr Yinli Niu from Beijing Jiaotong University, Prof. Linjin Wang from Zhejiang University, Prof Qian Peng from Institute of Chemistry, Chinese Academy of Sciences, Ms Lihui Yan from Hongzhiwei Technology (Shanghai) Co. Ltd., Ms Lu Wang from Tsinghua University, and more.

Qikai Li
Department of Chemistry
Tsinghua University
April 13, 2021

Contents

Part one: Photophysical Properties

Part two: Transport Properties

Part one: Photophysical Properties

1. Introduction to MOMAP

1.1 Overview

MOMAP (Molecular Materials Property Prediction Package) is a suite of programs for predicting the properties of polyatomic molecules, which is jointly developed by the Key Laboratory of Organic Solids (Institute of Chemistry Chinese Academy of Sciences) and Prof. Zhigang Shuai's research group (Department of Chemistry, Tsinghua University).

MOMAP allows users to study the photophysical properties and charge transport properties of a given molecule. The first version of this user guide focuses on introducing theoretical calculations on photophysical properties, however, this new version will also include calculations on charge transport properties.

1.2 On photophysical properties

The excitation, radiative decay and non-radiative decay are the basic photophysical and photochemical processes. They all have direct impact on the optical and electrical properties of a molecule.

The key photophysical processes, which can be described by Jablonski energy diagram, as shown in Fig. 1. MOMAP can simulate the spectrum of a given molecule, including the absorption, fluorescence and phosphorescence spectrum. In addition, MOMAP is also able to calculate transition rate constants between two electronic states as well, including the radiative and non-radiative decay rate constants for the internal conversion (IC) process and intersystem crossing (ISC) process.

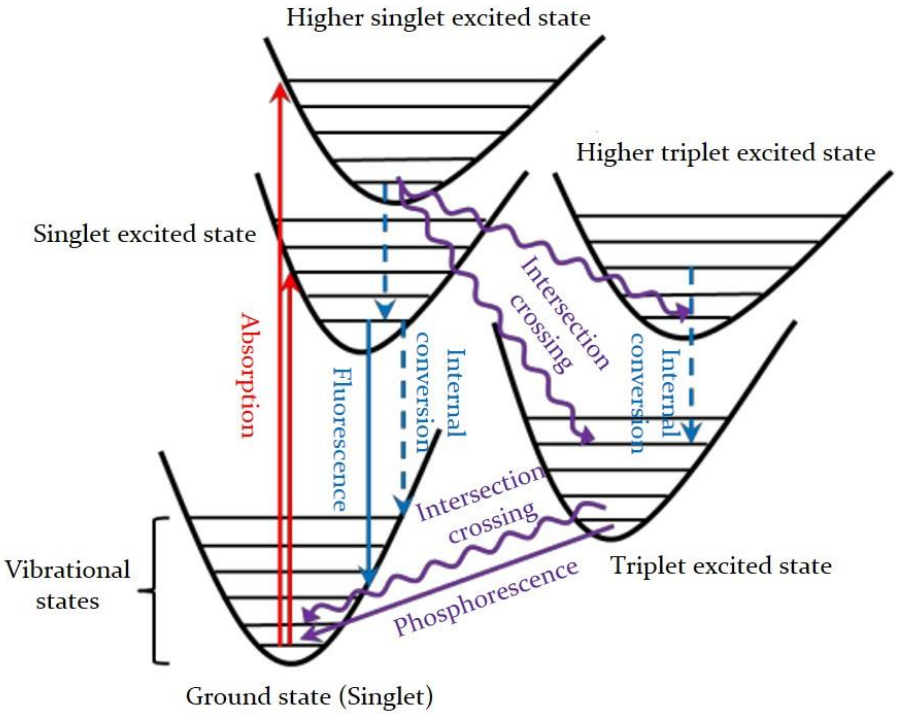


Fig. 1 Jablonski energy diagram

As we know, the photophysical properties are tightly related to the various properties of ground and excited states, which are the basis to carry out calculations in MOMAP. The various module dependencies on common quantum chemistry (QC) calculations are shown in Fig. 2.

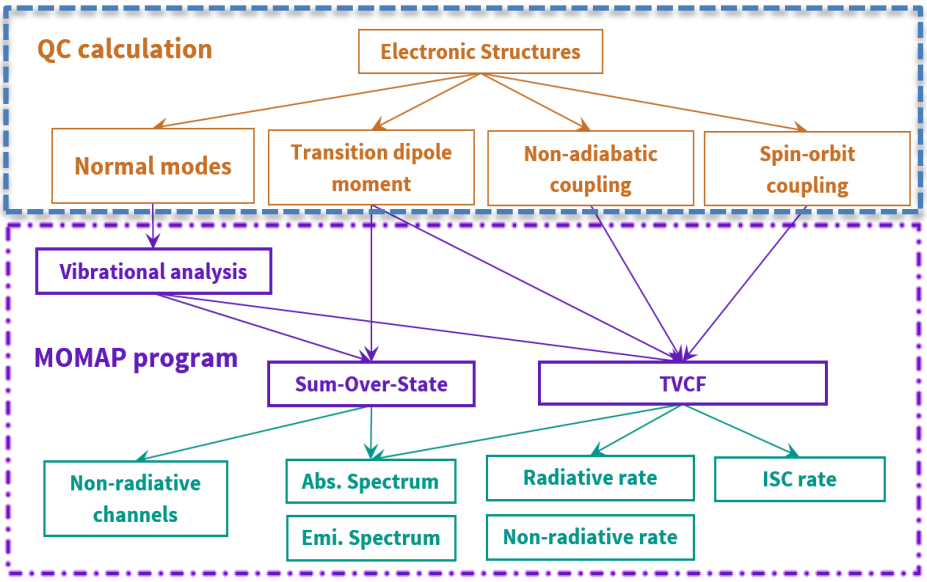


Fig. 2 Dependencies and relationships between MOMAP and QC calculations

In order to perform the calculation, including geometries, single point energy and vibrational modes of ground and excited states, as well as transition dipole moment, non-adiabatic coupling and spin-orbit coupling constant between ground and excited states, some key information should be provided to MOMAP. This information can be obtained from many other sophisticated quantum chemistry calculation packages, such as Gaussian, Q-Chem, TURBOMOLE, NWChem, ORCA, Dalton, ADF and BDF *etc*. The calculation details will be given in the following chapters.

1.3 How to cite MOMAP

Users who publish papers based on MOMAP calculations are required to cite：
MOMAP – Molecular Material Property Prediction Package. http://www.momap.net.cn

Optionally, you may cite the following literatures：

1) Peng, Q.; Yi, Y.; Shuai, Z.; Shao, J., *The Journal of Chemical Physics*, **2007**, *126*, 114302.
2) Peng, Q.; Yi, Y.; Shuai, Z.; Shao, J., *Journal of the American Chemical Society*, **2007**, *129*, 9333-9339.
3) Niu, Y.; Peng, Q.; Shuai, Z., *Science in China Series B: Chemistry*, **2008**, *51*, 1153-1158.
4) Niu, Y.; Peng, Q.; Deng, C.; Gao, X.; Shuai, Z., *The Journal of Physical Chemistry A*, **2010**, *114*, 7817-7831.
5) Peng, Q.; Niu, Y.; Shi, Q.; Gao, X.; Shuai, Z., *Journal of Chemical Theory and Computation*, **2013**, *9*, 1132-1143.
6) Shuai, Z., *Chinese Journal of Chemistry*, **2020**, 38, 1223-1232.

2. MOMAP installation

2.1 Download MOMAP

MOMAP can be downloaded from the following site:

http://www.momap.net.cn/index.php/download

2.2 Install MOMAP

(1) Operating System

MOMAP is mainly targeted to be running in a UNIX-like environment. However, it can also be installed on Windows and MacOS.

(2) Install MOMAP package

The MOMAP package for Linux is downloaded as a single zipped installable run file, e.g., `momap-2021A-linux-mpich2.run.gz`

Unzip the file and add executable attribute to the file, and install MOMAP package by running:

```
$ gunzip momap-2021A-linux-mpich2.run.gz
$ chmod a+x momap-2021A-linux-mpich2.run
$ ./momap-2021A-linux-mpich2.run
```

Or more generally, install MOMAP package by running:

```
$ sh momap-2021A-linux-mpich2.run
```

If MOMAP is to be run under the Ubuntu Linux system, before we start to install MOMAP, we need to promote the user rights and make the user to be an administrator. Here, we will show you how to make a user an administrator in Ubuntu through the command line:

```
$ sudo usermod -aG sudo sampleuser
```

You can verify that the user is now in the “sudo” group by checking the groups a user belongs to, through the following command:

```
$ groups sampleuser
```

Log out and log back in again for it to take effect, then we can proceed to have a normal MOMAP installation.

Before using the MOMAP, one should first set up the running environment, add the following line to `~/.bashrc` if Bash is used:

```
. installed_momap_folder/env.sh
```

Log out and log back in again for it to take effect, then we can proceed to run MOMAP.

If Bash shell is used, one can add the following line:

```
export MOMAP_LICENSE_DEBUG=TRUE
```

in file `~/.bashrc` to activate output of license debug information.

In some supercomputing centers, the SSH port may not be the default 22, in that case, we need to setup the SSH environment variable, for example:

```
export MOMAP_SSH_PORT=5577
```

As to the specific port used, please contact the supercomputing center's system administrator for information.

If the `Environment Modules` is installed, one can use the command `module` to manage the MOMAP running environment, please refer to the installation guide for more details.

For more information, please refer to the **MOMAP Installation Guide**.

3. Duschinsky Rotation Matrix and Vibrational Analysis

3.1 Overview

MOMAP is able to analyze Duschinsky rotation and normal mode vibrations, which is based on `evc_int` and `evc_cart` subprograms[1].

In the following part of this guide, this kind of calculation will be termed as `evc` calculation.

The `evc` calculation can use outputs from other QC programs, such as Gaussian, Q-Chem, TURBOMOLE, ChemShell, Dalton, MOLPRO, DFTB and MOPAC *etc*. It can also read data from the output files, including vibrational frequencies and force constant matrix, and calculate normal mode displacement, Huang-Rhys factor, reorganization energy and Duschinsky rotation matrix between initial and final electronic states under both internal coordinate and Cartesian coordinate.

3.2 Start a calculation

The `evc` calculation requires the basic information on initial and final electronic states. Thus, to begin an `evc` calculation, you need to designate the related file names in the MOMAP input file (i.e., `momap.inp`).

Here is an example of the simplest `evc` input file. For the Gaussian output files, you have to provide the corresponding `.fchk` files as well.

Take `tests/azulene/evc` as an example, edit `momap.inp`, and add the following in the file:

```
[evc]$ cat momap.inp
do_evc      = 1                         # toggle dushin rotation effect, 1 or 0

&evc
  ffreq(1)  = "azulene-s0.log"          # log file of ground state (GS)
  ffreq(2)  = "azulene-s1.log"          # log file of excited state (ES)
/
```

When the `evc` input file is ready, users can do the calculation by running:

```
$ momap.py -i momap.inp -n 4
```

The example also contains a `run` script file, users may modify the file, for example, by changing

[1] dushin_int and dushin_cart programs are modified from dushin program, which is written by Prof. Jeffrey R. Reimers (J. Chem. Phys. vol. 115, 9103-9109, 2001)

the `np` option from 4 to 8, and perform calculation by running the script file:

```
$ ./run
```

Except for `ffreq(1)` and `ffreq(2)` parameters, `evc` program also allows user to project reorganization energy onto the internal coordinate, to take account of isotope effect, and to configure many other advanced settings *etc*. Please refer to Appendix for more parameter settings.

3.3 Program outputs

The main results obtained from this calculation are the properties between initial and final electronic states - normal mode displacement, Huang-Rhys factor, reorganization energy and Duschinsky rotation matrix. The information is kept in `evc.cart.dat` and `evc.dint.dat`.

(1) `evc.cart.dat`

Use Cartesian coordinate to calculate the above properties.

(2) `evc.dint.dat`

Use internal coordinate to calculate normal mode displacement, Huang-Rhys factor and reorganization energy, while using cartesian coordinate to calculate Duschinsky rotation matrix.

Please check the reorganization energy results between `evc.cart.dat` and `evc.dint.dat`. If the energy difference is small (< 1000 cm^{-1}), then use the results in `evc.cart.dat` to do the next calculations. However, if the energy difference is large, then use `evc.dint.dat` to do the next calculations.

4. Fluorescence Spectrum Calculation

4.1 Overview

MOMAP is able to simulate fluorescence spectrum and calculate the corresponding radiative decay rate constant based on the `TVCORF_SPEC`[2] and `TVSPEC_SPEC` subprograms. The `TVCORF_SPEC` subprogram is used to calculate the thermal vibration correlation function (TVCF), while the `TVSPEC_SPEC` subprogram is used to simulate the fluorescence spectrum.

In the following part of this guide, this kind of calculation will be termed as `rad_FL` calculation.

4.2 Start a calculation

To start a `rad_FL` calculation, you need a `*.dat` file, an MOMAP control file, and an optional parallel control file. The `*.dat` file is obtained from the previous mentioned `evc` calculation. A MOMAP control file is used to control how `TVCORF_SPEC` and `TVSPEC_SPEC` subprograms behavior. An optional parallel control file is used to control how many computing processes will be used.

An example for performing `rad_FL` calculation with azulene is put in directory `tests/azulene/kr`. The files needed for this calculation are as follows:

(1) `evc.cart.dat`

`Evc` calculation result file.

(2) `momap.inp`

MOMAP control file.

(3) `nodefile` (optional, it may be automatically generated by running options `-n #`)

Parallel machinefile to control how many computing processes will be used.

4.3 Modifying control file

Here is an example of the MOMAP control file for a `rad_FL` calculation, it shows the various parameters and their meanings. Users may need to change the parameters accordingly before performing a `rad_FL` calculation.

[2] TVCORF_SPEC_para subprogram is available for parallel computation.

```
[kr]$ cat momap.inp
do_spec_tvcf_ft     = 1                     # toggle correlation function calculation, 1 or 0
do_spec_tvcf_spec   = 1                     # toggle fluorescence spectrum calcluation, 1 or 0

&spec_tvcf
  DUSHIN          = .t.                     # toggle Duschinsky rotation effect, .t. or .f.
  Temp            = 300 K                   # temperature
  tmax            = 1000 fs                 # integration time
  dt              = 1   fs                  # integration timestep
  Ead             = 0.075092 au             # adiabatic excitation energy
  EDMA            = 0.92694 debye           # electronic dipole moment of absorption (GS)
  EDME            = 0.64751 debye           # electronic dipole moment of emission (ES)
  FreqScale       = 1.0                     # frequency scaling factor
  DSFile          = "evc.cart.dat"          # input dushin file
  Emax            = 0.3 au                  # upper bound of spectrum frequency
  dE              = 0.00001 au              # output energy interval
  logFile         = "spec.tvcf.log"         # output file for logging
  FtFile          = "spec.tvcf.ft.dat"      # output file for correlation function info
  FoFile          = "spec.tvcf.fo.dat"      # output file for spectrum function info
  FoSFile         = "spec.tvcf.spec.dat"    # output file for spectrum info
/
```

If the users want to use the *sum-over-states* approach to calculate the spectrum at absolute zero (0 K), please refer to directory `examples/azulene/sumstat` for more details.

4.4 Verify convergence of correlation function and obtain results

Correlation function must be converged before obtaining any calculation results. To verify, plot a graph using the first 2 columns in the `spec.tvcf.ft.dat`, which are time and real part of the correlation function (TVCF_RE). The TVCF_RE should be very close to zero and stop oscillating before it reaches the integration time limit. Fig. 3 shows a plot of a converged correlation function.

The radiative decay rate constant can be found at the end of `spec.tvcf.log` file, while the fluorescence spectrum information can be obtained from `spec.tvcf.spec.dat`.

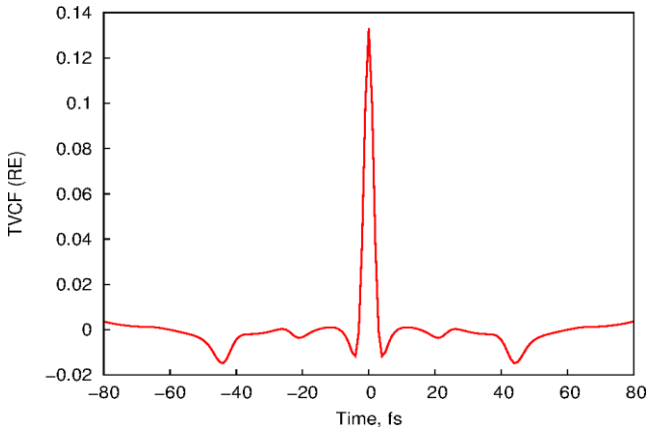


Fig. 3 Distribution of time *vs* real part of a converged correlation function

5. Internal Conversion (IC) Rate Constant

5.1 Overview

MOMAP is also able to calculate internal conversion (IC) rate constant based on the `TVCORF_IC`[3] and `TVSPEC_IC` subprograms. The `TVCORF_IC` subprogram is used to calculate thermal vibration correlation function (TVCF), while the `TVSPEC_IC` subprogram is used to determine the relationship between IC rate constant and energy gap.

In the following part of this guide, this kind of calculation will be termed as `nonrad` calculation.

5.2 Calculate non-adiabatic coupling matrix element (NACME)

Unlike the `rad_FL` calculation, The NACME should be obtained before performing a `nonrad` calculation.

In MOMAP, the `get-nacme` function is used to read the transition electric field and vibration information from Gaussian output and calculate electronic coupling term. This process is integrated into `evc_int` and `evc_cart` subprogram. Users need to provide the Gaussian electric field calculation result and toggle on the `NACME` calculation in MOMAP control file `momap.inp`. The other steps are the same as for a normal `evc` calculation. After finishing the calculation, a `*.nac` file will be generated, and it is used in the `nonrad` calculation.

5.3 Start a calculation

To start a `nonrad` calculation, you need a `*.dat` file, a `*.nac` file, a MOMAP control file, and an optional parallel control file. The `*.dat` file is the `evc` result file. The `*.nac` is the `NACME` calculation result file. As for all MOMAP calculations, a MOMAP control file is needed for `TVCORF_IC` and `TVSPEC_IC` subprograms. A parallel control file is used to control how many computing processes will be used.

An example for performing `nonrad` calculation with azulene are put in directory `tests/azulene/kic/kic`. The files needed in this example are as follows:

`(1) evc.cart.dat`

`Evc` calculation result file.

[3] TVCORF_IC_para subprogram is available for parallel computation.

(2) `evc.cart.nac`

`NACME` result file.

(3) `momap.inp`

MOMAP control file.

(4) `nodefile` (optional, it may be automatically generated by running options `-n #`)

Parallel machinefile to control how many computing processes will be used.

5.4 Modify control file

Here is an example of the MOMAP control file for a `nonrad` calculation, it shows the various parameters and their meanings. Users may need to change the parameters accordingly before performing a `nonrad` calculation.

```
[kic/kic]$ cat momap.inp
do_ic_tvcf_ft   = 1                  # toggle internal conversion correlation function, 1 or 0
do_ic_tvcf_spec = 1                  # toggle internal conversion spectrum, 1 or 0

&ic_tvcf
  DUSHIN         = .t.               # toggle Duschinsky rotation effect, .t. or .f.
  Temp           = 300 K             # temperature
  tmax           = 1000 fs           # integral interval of correlation function
  dt             = 1   fs            # integration timestep of correlation function
  Ead            = 0.075092 au       # adiabatic excitation energy difference between two states
  DSFile         = "evc.cart.dat"    # input dushin file
  CoulFile       = "evc.cart.nac"    # input nacme info file
  Emax           = 0.3 au            # upper bound of spectrum frequency
  logFile        = "ic.tvcf.log"     # output file for logging
  FtFile         = "ic.tvcf.ft.dat"  # output file for correlation function info
  FoFile         = "ic.tvcf.fo.dat"  # output file for spectrum function info
/
```

5.5 Verify convergence of correlation function and obtain results

Make sure the correlation function is converged, this is very important. The verification process can be found in section 4.4.

The internal conversion (IC) rate constant can be found at the end of `ic.tvcf.log` file. The relationship between IC rate constant and energy gap can also be obtained from the `ic.tvcf.log` file.

6. Phosphorescence Spectrum Calculation

6.1 Overview

MOMAP is also able to simulate the phosphorescence spectrum and calculate the corresponding radiative rate constant and intersystem crossing (ISC) rate constant based on the `TVCORF_SPEC`[4] and `TVSPEC_SPEC` subprograms. The `TVCORF_SPEC` subprogram is used to calculate thermal vibration correlation function (TVCF), while the `TVSPEC_SPEC` subprogram is used to simulate phosphorescence spectrum.

In the following part of this guide, this kind of calculation will be termed as `rad_PL` calculation.

6.2 Start a calculation

To start a `rad_PL` calculation, you need a `*.dat` file, a MOMAP control file and an optional parallel control file. The `*.dat` file is obtained from the previous `evc` calculation. The MOMAP control file is used to control the calculations of `TVCORF_SPEC` and `TVSPEC_SPEC` subprograms. The parallel control file is used to control now many computing processes will be used.

An example for performing a `rad_PL` calculation on $Ir(ppy)_3$ are put under directory `tests/Irppy3`. The files needed for this calculation are as follows:

(1) `evc.cart.dat`

`Evc` calculation result file.

(2) `momap.inp`

MOMAP control file.

(3) `nodefile` (optional, it may be automatically generated by running options `-n #`)

Parallel machinefile to control how many computing processes will be used.

6.3 Modify control file

Here is an example of the MOMAP control file for performing a `rad_PL` calculation, it shows the various parameters and their functions. Change the parameters accordingly before carrying out the calculation.

[4] TVCORF_SPEC_para subprogram is available for parallel computation.

```
[kr]$ cat momap.inp
do_spec_tvcf_ft     = 1                    # toggle correlation function calculation, 1 or 0
do_spec_tvcf_spec   = 1                    # toggle fluorescence spectrum calcluation, 1 or 0

&spec_tvcf
  DUSHIN         = .t.                     # toggle Duschinsky rotation effect, .t. or .f.
  Temp           = 298 K                   # temperature
  tmax           = 1500 fs                 # integration time
  dt             = 1   fs                  # integration timestep
  Ead            = 0.0941289 au            # adiabatic excitation energy
  EDMA           = 1.0 debye               # electronic dipole moment of absorption (GS)
  EDME           = 0.306909 debye          # electronic dipole moment of emission (ES)
  DSFile         = "evc.cart.dat"          # input dushin file
  Emax           = 0.3 au                  # upper bound of spectrum frequency
  dE             = 0.00001 au              # output energy interval
  logFile        = "spec.tvcf.log"         # output file for logging
  FtFile         = "spec.tvcf.ft.dat"      # output file for correlation function info
  FoFile         = "spec.tvcf.fo.dat"      # output file for spectrum function info
  FoSFile        = "spec.tvcf.spec.dat"    # output file for spectrum info
/
```

6.4 Verify convergence of correlation function and obtain results

Make sure the correlation function is converged. The verification process can be found in section 4.4 of this guide.

The radiative decay rate constant can be found at the end of `spec.tvcf.log.dat` file. The phosphorescence spectrum information can be obtained from the `spec.tvcf.spec.dat` file.

6.5 Intersystem crossing

Here is an example of the MOMAP control file for performing an ISC calculation:

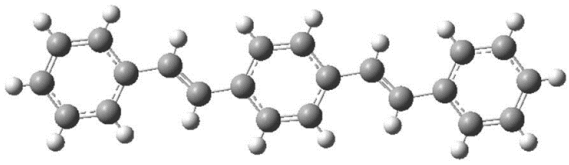
```
[kisc]$ cat momap.inp
do_isc_tvcf_ft      = 1                    # toggle isc correlation function calculation, 1 or 0
do_isc_tvcf_spec    = 1                    # toggle isc spectrum calculation, 1 or 0

&isc_tvcf
  DUSHIN         = .t.                     # toggle Duschinsky rotation effect, .t. or .f.
  Temp           = 298 K                   # temperature
  tmax           = 1500 fs                 # integration time
  dt             = 1 fs                    # integration timestep
  Ead            = 0.0941289 au            # adiabatic excitation energy
  Hso            = 116.877376 cm-1         # Spin-orbit coupling constant
  DSFile         = "evc.cart.dat"          # input dushin file
  Emax           = 0.3 au                  # upper bound of spectrum frequency
  logFile        = "isc.tvcf.log"          # output file for logging
  FtFile         = "isc.tvcf.ft.dat"       # output file for correlation function info
  FoFile         = "isc.tvcf.fo.dat"       # output file for spectrum function info
/
```

Again, the intersystem crossing (ISC) rate constant can be found at the beginning of `isc.tvcf.fo.dat` file.

7. Obtain Information from QC packages

In the following part of this guide, we will show you how to obtain the useful calculation results from various QC packages. The 1,4-distyrylbenzene molecule (DSB for short) and Gaussian 09 package are used to demonstrate the process.



The Gaussian 09 is used to do optimization and frequency calculations on the ground state (S_0) and the lowest singlet excited state (S_1), the transition dipole moment and the transition electric field between S_0 and S_1 states.

7.1 Optimization calculation on ground state (S_0)

Once the initial geometry is constructed, we have to find the optimized S_0 geometry. The route section is set as `#p opt b3lyp/6-31g*`, which indicates an optimization at `B3LYP/6-31G*` level.

When the calculation is completed, locate the **last line with "`SCF Done`"** in the output `*.log` file in order to find the single point energy at the optimized S_0 geometry. In this example, the last line with "`SCF Done`" is like the following:

```
SCF Done:  E(RB3LYP) =  -849.172438992    A.U.
```

The complete results can be found in directory `tests/DSB/opt_and_frequency`.

7.2 Frequency calculation at the optimized S_0 geometry

After finding the optimized S_0 geometry, we need to verify the optimization result and calculate its force constant matrix *via* frequency calculation. The route section is set as `#p freq b3lyp/6-31g*`, which runs a frequency calculation at the `B3LYP/6-31G*` level. You have to define the location of `*.chk` file in Link 0 Commands as well.

Use the Gaussian built-in command `formchk` to generate a `*.fchk` file based on the output `*.chk`. The `*.fchk` file contains readable force constant matrix information that is needed in dushin

calculation.

The complete results can be found in directory `tests/DSB/opt_and_frequency`.

In this example, the route section is set as `#p opt freq b3lyp/6-31g*`, which means we run optimization and frequency calculations at the same time. However, we recommend separating them into two types of calculation in order to avoid any possible mistakes.

7.3 Transition dipole moment (absorption) at the optimized S_0 geometry

After finding the optimized S_0 geometry, we can calculate transition dipole moment (absorption) and vertical excitation energy at this geometry. The route section is set as `#p td b3lyp/6-31g*`, which runs a calculation at `B3LYP/6-31G*` level by using the TDDFT method.

When the calculation is completed, find the related information about "`Excited State 1`" in the output `*.log` file in order to find the vertical excitation energy and transition dipole moment (absorption) at the optimized S_0 geometry. In this example, the information is listed below:

```
Ground to excited state transition electric dipole moments (Au):
state          X           Y           Z        Dip. S.      Osc.
  1       -4.6693     -0.0118      0.0112     21.8029    1.7826


Excited State   1: Singlet-A 3.3372 eV  371.52 nm  f=1.7826  <S**2>=0.000  75 -> 76  0.70728
This state for optimization and/or second-order correction.
Total Energy, E(TD-HF/TD-KS) =  -848.655200149
```

Hence, the vertical excitation energy at the optimized S_0 geometry is 3.3372 eV, and the transition dipole moment (absorption) can be obtained using Dip. S.:

$$\sqrt{21.8029} \times 2.54 \text{ Debye} = 11.86 \text{ Debye}$$

7.4 Optimization calculation on lowest singlet excited state (S_1)

With the optimized S_0 geometry at hand, we can start optimizing S_1 geometry using the optimized S_0 geometry as the initial structure. The route section is set as `#p td opt b3lyp/6-31g*`, which indicates an optimization at the `B3LYP/6-31G*` level using TDDFT method.

When the calculation is completed, locate the **last line with "`SCF Done`"** in the output *.log file in order to find single point energy at the optimized S_1 geometry. In this example, the last line with

"SCF Done" is the following:

```
SCF Done:  E(RB3LYP) =  -849.165742659     A.U.
```

Complete results can be found in directory `tests/DSB/opt_and_frequency`.

7.5 Frequency calculation at the optimized S_1 geometry

After finding the optimized S_1 geometry, we need to verify the optimization result and calculate its force constant matrix *via* frequency calculation. The route section is set as `#p td freq b3lyp/6-31g*`, which runs a frequency calculation at the `B3LYP/6-31G*` level using TDDFT method. You have to define the location of *.chk file in Link 0 Commands as well.

Use Gaussian built-in command `formchk` to generate a *.fchk file based on output *.chk. The *.fchk file contains readable force constant matrix information that is needed in dushin calculation.

The complete results can be found in directory `tests/DSB/opt_and_frequency`.

7.6 Transition dipole moment (emission) at the optimized S_1 geometry

Transition dipole moment (emission) and vertical excitation energy at the optimized S_1 geometry are also given when the calculation in section 7.4 is done. Find the relative information about "Excited State 1" in the output *.log file in order to locate the vertical excitation energy and transition dipole moment (emission) at the optimized S_1 geometry. In this example, the information is listed below:

```
Ground to excited state transition electric dipole moments (Au):
state          X           Y           Z        Dip. S.      Osc.
  1       -5.3165     -0.0242      0.0000     28.2653      1.9597


Excited State   1:   Singlet-?Sym 2.8300 eV   438.11 nm   f=1.9597   <S**2>=0.000   75 -> 76   0.71066
This state for optimization and/or second-order correction.
Total Energy, E(TD-HF/TD-KS) =  -849.061743778
```

Hence, vertical excitation energy at the optimized S_1 geometry is 2.8300 eV, and transition dipole moment (emission) can be obtained using Dip. S.:

$$\sqrt{28.2653} \times 2.54 \text{ Debye} = 13.50 \text{ Debye}$$

The complete results can be found in directory `tests/DSB/opt_and_frequency`.

7.7 Adiabatic energy difference between S_0 and S_1 states

The adiabatic energy difference between S_0 and S_1 states can be calculated using single point energy results from sections 7.1 and 7.4.

In this example, the adiabatic energy difference is:

$$(-849.06174378+849.17423899)\times 27.2114\ \text{eV}=3.0122\ \text{eV}$$

7.8 Transition electric field and NACME at the optimized S_1 geometry

After finding the optimized S_1 geometry, we can calculate transition electric field at this geometry. Then it's possible to run a dushin calculation with NACME option toggled on.

The route section is set as the following line:

```
#p td b3lyp/6-31g(d) prop=(fitcharge,field) iop(6/22=-4, 6/29=1, 6/30=0, 6/17=2)
```

When the calculation is completed, copy two output *.log files into a new directory. One is transition electric field *.log file, which is obtained in this section. The other is frequency calculation at the optimized S_0 geometry *.log file, which is obtained in section 7.2. Then use `get-nacme` to start calculating `NACME`.

The complete results can be found in directory `tests/DSB/nacme`.

Part two: Transport Properties

8. Transport: Background

8.1 Charge Transfer Rate

Due to the weak coupling between molecules, the charge transport in most organics is dominated by the hopping mechanism, which implies that the transport dynamics can be decomposed into elementary charge transfer processes between different pairs of molecules. [1] The charge transfer between two molecules, M_i and M_j, is a charge exchange reaction. The initial and final states are M'_iM_j and $M_iM'_j$, where M' denotes the charge on molecule M. The widely used charge transfer rate from the classical Marcus theory reads[2]

$$k_{ij} = \frac{V_{ij}^2}{\hbar}\sqrt{\frac{\pi}{\lambda k_B T}}\exp\left[-\frac{\left(\lambda+\Delta G_{ij}^0\right)^2}{4\lambda k_B T}\right]$$

Here, V_{ij} is the transfer integral between the initial and final states, λ is the reorganization energy which is defined as the energy change associated with the geometry relaxation during the charge transfer, and ΔG_{ij}^0 is relevant change of total Gibbs free energy. In molecular semiconductors with only one kind of molecules, ΔG_{ij}^0 equals to zero, and then the Marcus rate becomes

$$k_{ij} = \frac{V_{ij}^2}{\hbar}\sqrt{\frac{\pi}{\lambda k_B T}}\exp\left(-\frac{\lambda}{4k_B T}\right)$$

Thereby, the charge transfer is actually a thermal activation process over a barrier of $\lambda/4$. The Marcus rate is most appropriate when (1) the temperature is high, (2) the molecules are in equilibrium in both the initial and final states, and (3) the intermolecular diabatic couplings are weak. [3]

8.2 Pauli Master Equation

When the charge transfer rates are defined, the charge transport dynamics can be characterized by the Pauli master equation (PME) with a Markov assumption. [4] It describes the time evolution of the electron population on each state with a set of purely classical kinetic equations,

$$\dot{P}_i = \sum_{j\neq i}\left(k_{ji}P_j - k_{ij}P_i\right)$$

where P_i is the occupation number of the charge to be on molecule i, and k_{ij} is the charge transfer rate from molecule i to j. Namely, the population change on state i is the difference between the total

population transfer from other states to state *i* and that from state *i* to other states. The PME can be solved directly through an iterative numerical scheme. One needs to set the initial populations on all states and solve the differential equations to get the charge distribution at any time *t*. Alternatively, one may use a kinetic Monte Carlo (KMC) algorithm with a series random walk trajectories. [5] In comparison, the KMC approach is numerically more efficient for large systems as converged results can be generally obtained with a sampling of acceptable number of trajectories.

8.3 Monte Carlo Simulation

The concept of Monte Carlo was first proposed by Stanislaw Ulam in the 1940s. Ulam was a mathematician who worked on the Manhattan Project. Initially, the method was derived to solve the problem of determining the average distance neutrons would travel through various materials. The method was named after the Monte Carlo Casino in Monaco since the randomness of the outcomes that is crucial to games such as roulette or dices is essential for Monte Carlo simulations.

The main ideas behind the Monte Carlo simulation are the repeated random sampling of inputs of the random variable and the aggregation of the results. The variable with a probabilistic nature is assigned a random value. The model is then calculated based on the random value. The result of the model is recorded, and the process is repeated. Usually, the process is repeated hundreds or thousand times. When the simulation is complete, the results can be averaged to determine the estimated value.

8.4 Lattice Random Walk

Random walk is a typical kind of Monte Carlo process. Using a regular lattice as an illustration, at each step the location jumps to another site according to some probability distribution. In a *simple random walk*, the location can only jump to neighboring sites of the lattice, forming a *lattice path*, as show in Figure 1.

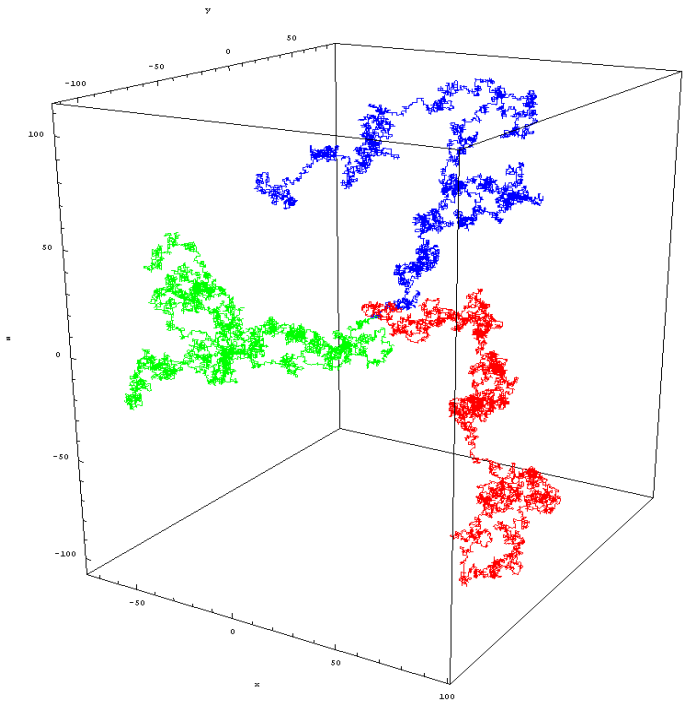


Figure 8-1. Three representative random walks in three dimensions.

For a particle in a known fixed position at $t = 0$, the central limit theorem tells us that after a large number of independent steps in the random walk, the walker's position is distributed according to a normal distribution of total *variance*:

$$\sigma^2 = \frac{t}{\delta t}\varepsilon^2,$$

where t is the time elapsed since the start of the random walk, ε is the size of a step of the random walk, and δt is the time elapsed between two successive steps. This corresponds to the Green function of the diffusion equation that controls the Wiener process, which suggests that, after a large number of steps, the random walk converges toward a Wiener process (a stochastic process with similar behavior to Brownian motion, sometimes the Wiener process is called "Brownian motion"). In three dimensions, the *variance* corresponding to the Green's function of the diffusion equation is:

$$\sigma^2 = 6\mathrm{D}t\,.$$

By equalizing this quantity with the variance associated to the position of the random walker, one obtains the equivalent diffusion coefficient to be considered for the asymptotic Wiener process toward which the random walk converges after a large number of steps:

$$\mathrm{D} = \frac{\sigma^2}{2n\mathrm{t}} = \frac{\varepsilon^2}{2n\delta t},$$

where $n = 1$, 2, or 3 is the dimensionality of the system under investigation.

8.5 Charge Carrier Mobility

The mobility of a charge carrier is related to the diffusion coefficient by the *Einstein* relationship:

$$\mathrm{D} = \frac{\mathrm{k}_B \mathrm{T}}{q} \mu \,.$$

Rearrange the above equation, we have:

$$\mu = \frac{q}{\mathrm{k}_B \mathrm{T}} \mathrm{D} \,.$$

Thus, the task is to first find the diffusion coefficient of a charge carrier, then the mobility of charge carrier is obtained by using the *Einstein* relationship. [1] To fulfil the task, we can resort to Monte Carlo simulations. In detail, an arbitrary molecular site in the bulk system is initially chosen as the starting position for the charge. The charge then has a probability of p_i to hop to the i-th neighbor (see Figure 2). In practice, in order to determine the next site of the charge in a statistical sense, a random number ξ uniformly distributed between 0 and 1 is generated. If $\sum_{i=1}^{b-1} p_i < \xi < \sum_{i=1}^{b} p_i$, the charge hops to the b-th neighbor with a hopping time $1/k_b$, which assumes no correlation between the hopping events along different pathways. The simulation continues until the diffusion distance exceeds the lattice constant by at least 2 – 3 orders of magnitude. This process is repeated for thousands of times and averaged to get a linear relationship between the mean-square displacement (MSD) and the simulation time.

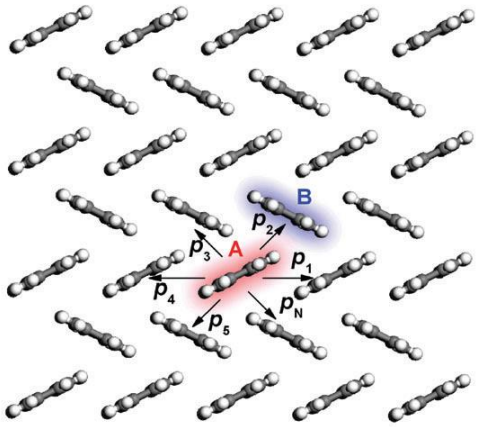


Figure 8-2. Schematic representation of the charge hopping pathways from molecule A to its neighbors with probabilities p_1, p_2, …, and p_N.

The diffusion coefficient D is calculated through

$$\mathrm{D} = \frac{1}{2n} \frac{dMSD}{dt} \,.$$

where n = 1, 2, or 3 is the dimensionality of the system under investigation.

In MOMAP Transport package, we setup a lot of initial different (random) seeds, record the tracks, and then average over the tracks. Even though an individual track looks rather jittery, however, when a certain number of (say 2,000) tracks are averaged, we can get a pretty linear averaged line (thick red line), as shown in the following figure:

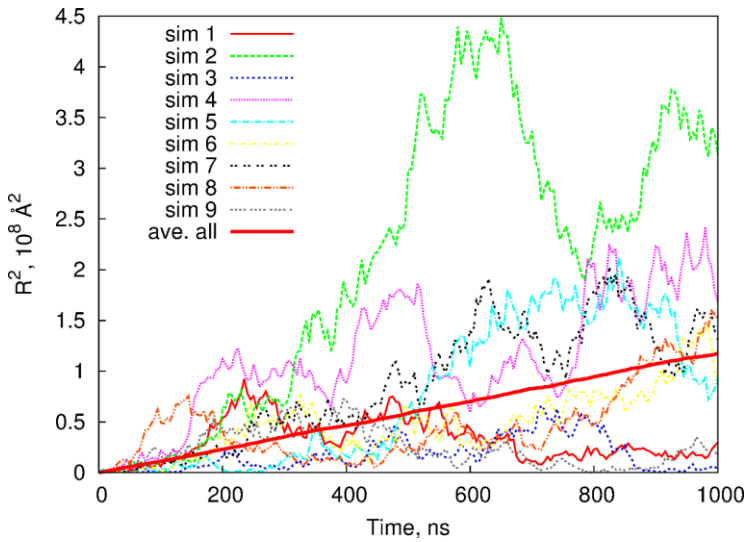


Figure 8-3. Mobility calculation by using Monte Carlo simulations.

With the averaged track, we can fit to a linear line, obtain the diffusion coefficient, and then the charge carrier mobility.

References

1. L. J. Wang, G. J. Nan, X. D. Yang, Q. Peng, Q. K. Li, and Z. Shuai, Chem. Soc. Rev. 39, 423-434 (2010).
2. R. A. Marcus, Rev. Mod. Phys. 65, 599-610 (1993).
3. S. H. Lin, C. H. Chang, K. K. Liang, R. Chang, Y. J. Shiu, J. M. Zhang, T. S. Yang, M. Hayashi, and F. C. Hsu, Adv. Chem. Phys. 121, 1 (2002).
4. H. J. Kreuzer, Nonequilibrium Thermodynamics and Its Statistical Foundations (Oxford University Press, New York, 1981).
5. X. D. Yang, L. J. Wang, C. L. Wang, W. Long, and Z. Shuai, Chem. Mater. 20, 3205-3211 (2008).

9. Transport: Quick Start

We assume that MOMAP is properly installed and configured for your site, you can edit your `.bashrc` if you are using Bash in Linux.

This chapter is intended for the novice user in using the program package, especially the MOMAP Transport sub-package. We will use the example of Naphthalene to show which steps are necessary to prepare a calculation and run the MOMAP transport calculations. In the meantime, we will give all important information in a consistent way.

9.1 Naming conversions

Before we begin with our first introductory example, we describe the naming conventions, to which we will abide by throughout this user's guide.

On UNIX-like systems the files are case-sensitive, when running MOMAP it is required that all files reside in a subdirectory `./test1`. Thus, if we want to remove the calculations, we can easily delete the whole directory, or run the script `transport_clear.sh` under the directory.

9.2 Create a new calculation

As mentioned above, we first create a sub-directory `test1`, and copy the file `naphthalene.cif` under `test1`. The naphthalene unit cell is shown in Figure 9-1.

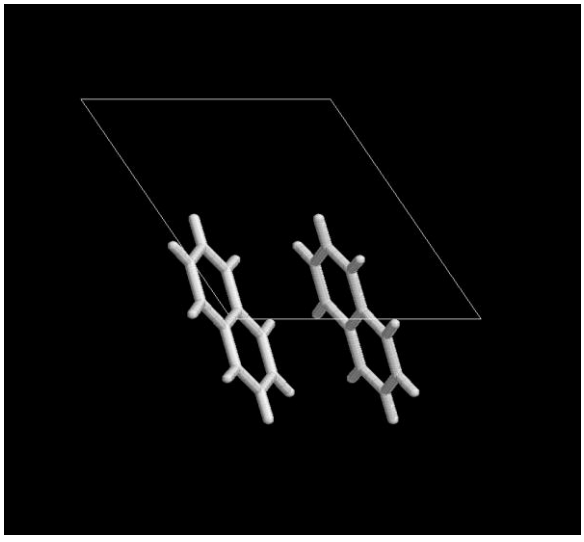


Figure 9-1 Naphthalene unit cell

Use an editor or any handy command, we can see the crystal structure parameters in the first few lines of naphthalene.cif, some things like the following:

```
[test1]$ cat  naphthalene.cif
data_naphthalene
_audit_creation_date              2017-03-25
_audit_creation_method            'Materials Studio'
_symmetry_space_group_name_H-M   'P1'
_symmetry_Int_Tables_number      1
_symmetry_cell_setting           triclinic
loop_
_symmetry_equiv_pos_as_xyz
  x,y,z
_cell_length_a                   8.0980
_cell_length_b                   5.9530
_cell_length_c                   8.6520
_cell_angle_alpha               90.0000
_cell_angle_beta               124.4000
_cell_angle_gamma               90.0000
…
```

It is clear that the `cif` file is an output of Materials Studio with space group of '`P1`', currently MOMAP Transport program can support all space groups, not restricted only to space group '`P1`'. With a `cif` file, we have all the crystal structure parameters in hand, as shown above.

Now we have prepared our first molecular file, however, before we can run the MOMAP Transport calculations, we have to prepare the `momap.inp` file, a control file for MOMAP package.

There exist quite a few of control parameters for `momap.inp`, however, all the parameters have their default values if we do not set them. To make life easy, we have written a program called `transport_geninp.exe` to generate the `momap.inp` for the MOMAP transport calculations.
Before we begin to generate the `momap.inp`, we would better setup our environment settings, as these settings rarely change in a specific computing cluster environment. The typical environment settings are as follows (you may put them in `~/.bashrc`):

```
export MOMAP_JOB_SCHED=slurm
export MOMAP_JOB_QUEUE=X12C
export MOMAP_QC_EXE=g09
export MOMAP_QC_PPN=12
export MOMAP_MODULE_QC=gaussian/g09.e01
```

These are the initial values that will be entered into our `momap.inp` if we run the `transport_geninp.exe`, we can change them later on with an editor. The currently supported scheduling systems include PBS, SLURM, LSF and LOCAL, these are the scheduling systems in frequent use. The LOCAL means the jobs are run in a local machine, it can be of great help for a Linux box without job scheduling system. If needed, more scheduling systems can be added. If the computing cluster is installed with environment module, the last two lines can be added, but we can change the contents according to our specific situation.

If our quantum calculation (QC) engine is of the Gaussian g16, we need to change the third line to g16, and the fifth line to g16.b01.

Once the MOMAP environments are set, we can use our `transport_geninp.exe` to generate `momap.inp`, all the MOMAP Transport programs have a help option, either `-h` or `--help`, for example, in case `transport_geninp.exe`, we have:

```
[test1]$ transport_geninp.exe --help
*******************************************************************************
*  MOMAP Transport Calculation Utility, Version 2021A (2.3.1) build on Mar 31 2021   *
*  Zhigang Shuai Group, Department of Chemistry, Tsinghua University, Beijing       *
*******************************************************************************
Transport Momap Input Generation

Usage: transport_geninp.exe [opts]
  -config s : set config file, default to momap.inp
  -cif      : use cif file as molecule input (default)
  -mol      : use mol file as molecule input
  -module   : set to use environment module flag
  -terse    : generate terse momap.inp (default)
  -verbose  : generate verbose momap.inp
  -oldconfig: generate old config.inp
  -orca     : use ORCA compute engine
  -qchem    : use Q-Chem compute engine

e.g.: transport_geninp.exe
  transport_geninp.exe -config momap.inp
  transport_geninp.exe -verbose
  transport_geninp.exe -cif
  transport_geninp.exe -mol
  transport_geninp.exe -module
  transport_geninp.exe -config momap.inp -verbose
  transport_geninp.exe -oldconfig
  transport_geninp.exe -orca
```

If option `-config` is used, then we can designate our output control file, the default file is `momap.inp` if this option is not specified.

If option `-cif` is used, then cif crystal parameter will be used in `momap.inp`, it will automatically search for the first found cif file in the current directory if it exists.

If option `-mol` is used, then mol molecule parameter will be used in `momap.inp`, again it will automatically search for the mol files in the current directory.

If option `-module` is used, then it will activate module parameter output.

If option `-terse` is used, then it will generate a *terse* `momap.inp`, while the other parameters using the default values.

If option `-verbose` is used, then it will generate a *verbose* `momap.inp`, almost all the parameters will be entered into the `momap.inp`. Thus, we can tune the parameters as needed.

These options can be used in combination, as shown in the last line.

If we run the `transport_geninp.exe` without any options, it will generate a `momap.inp` like the following:

```
[test1]$ cat momap.inp
&transport
  do_transport_prepare                = 1
  do_transport_submit_HL_job          = 1
  do_transport_get_transferintegral   = 1
  do_transport_submit_RE_job          = 1
  do_transport_get_re_evc             = 1
  do_transport_run_MC                 = 1
  do_transport_get_mob_MC             = 1
  do_transport_run_MC_temp            = 0
  do_transport_get_mob_MC_temp        = 0
  do_transport_run_ME                 = 0
  do_transport_get_mob_ME             = 0
  do_transport_run_ME_temp            = 0
  do_transport_get_mob_ME_temp        = 0
  do_transport_gatherdata             = 1

  # Job Scheduling
  sched_type        = pbs       ! pbs, slurm, lsf, or local
  queue_name        = blade

  compute_engine    = 1         ! 1 = Gaussian, 2 = ORCA, 3 = Q-CHEM
  qc_exe            = g09       ! g09/g16 or fullpath/orca or qchem

#  module_qc        = gaussian/g09.e01

  qc_method         = b3lyp
  qc_basis          = b3lyp STO-3g
  qc_basis_re       = b3lyp STO-3g
  qc_memory         = 4096  ! MB
  qc_nodes          = 1
  qc_ppn            = 8

  # Temperature Dependence
  temp              = 300
  start_temp        = 200
  end_temp          = 300
  delta_temp        = 50
  ratetype          = marcus  ! macus or quantum
  lat_cutoff        = 4       ! for neighbor list construction

  nsimu             = 2000
  tsimu             = 1000    ! in ns
  tsnap             = 5

  crystal           = naphthalene.cif
/
```

Other control parameters can be entered into `momap.inp`, users are advised to consult the `data/config.inp` for details.

For example, users may add a line like the following to `momap.inp`:

```
nodelist = "c01,c02"
```

to designate the computing nodes to be used in the running.

In addition, the `transport_prepare.exe` can also be used to prepare oniom input files, for example, if we set the 3rd bit up for control parameter `data_mol_output_level`, that is,

```
data_mol_output_level = 8
```

Then, after running `transport_prepare.exe`, we have 4 extra files in `data` directory:

```
mol1_oniom.com
mol1_oniom.xyz
mol2_oniom.com
mol2_oniom.xyz
```

The MOMAP Transport package uses the following control block:

```
&transport
…
/
```

It starts with `&transport`, and ends with `/`.
The lines beginning with '#' are comments, the '!' is also used for comment as in the case of Fortran coding.

The initial `do_` lines are control flags, can be either 1 (enabled) or 0 (disabled). If we need to do MC calculations, we simply set them to 1 accordingly.

With the generated `momap.inp`, we can do some fine tunings for our specific case, for example, we may change `queue_name` value from `blade` to `X12C`, `gaussian_ppn` from 16 to 12 *etc*.

In the meantime, a file called `run.sh` is also generated, it contains:

```
[test1]$ cat run.sh
#!/bin/sh

python $MOMAP_ROOT/bin/momap.py &> momap.log &
```

Once we have carefully checked the `momap.inp`, we can simply run the `run.sh` by issuing:

```
[test1]$ sh run.sh
```

Or

```
[test1]$ python $MOMAP_ROOT/bin/momap.py &> momap.log &
```

In the meantime, we can check the running processes by looking into the `momap.log` file, or the `RUN` directory where the running locks are located. We may also use the job scheduling commands to check the running processes.

If everything is okay, at the end of the log file, we can see somethings like the following:

```
…
**********************************
*     All successfully done.       *
**********************************
Duration: 0 days 4 hours 53 minutes 57 seconds.
```

will appear, which means the job is done successfully.

Please carefully check the `momap.log` file for any abnormalities.

Finally, the output results are gathered and put in file `momap.dat`.

10. Transport: Looking into the details

The `momap.py` is the job manager for MOMAP package, we can see the various running steps in the `momap.log` file. Now we have a look into the detailed processes of MOMAP Transport calculations.

10.1 Transport preparation

The first step of job manager `momap.py` is to run `transport_prepare.exe`. When the `transport_prepare.exe` is run, it will generate quite a few of directories and files.

To demonstrate how the data and directories are arranged for MOMAP transport calculations, we set both control parameters `HL_unique_mol` and `RE_unique_mol` to 0 in the `momap.inp`.

By running the `transport_prepare.exe`, the screen output is as follows:

```
$ transport_prepare.exe
****** Perform Transport Preparation...
Reading config file "momap.inp"...
Reading crystal file "naphthalene.cif"...
  Identifier:
  Spacegroup name: 'P1'
  Spacegroup operations:
    x,y,z
  Cell lattice a = 8.098
  Cell lattice b = 5.953
  Cell lattice c = 8.652
  Cell lattice alpha = 90
  Cell lattice beta  = 124.4
  Cell lattice gamma = 90
  natoms_cif = 36
  First atom: C1 C 0.082321 0.018562 0.328357
  ...
  Last  atom: H36 H 0.466698 0.795196 0.331298
  Unit cell nmols  = 2
  Unit cell natoms = 18 18
Crystal file naphthalene.cif parsing done.
Make whole molecules...
  molecule 1 COM =  0.000000  0.000000 -0.000000
  molecule 2 COM =  0.500000  0.500000  1.000000
Writing config file "data/config.inp"...

**** MOMAP Build Neighbor List ****
  Neighbor rcutoff distance: 4
  Neighbor search cell (-/+): 3 3 3
**** End of MOMAP Build Neighbor List ****

****** MOMAP Transport Preparation Succsessfully Done.
```

Then, all the necessary data and directories for MOMAP Transport calculations are prepared. The full directory and file tree is shown in the following pages (in Linux case, as follows, by simply run the `tree` command):

```
[test1]$ tree ./
./
├── data
│   ├── cellvec.dat
│   ├── config.inp
│   ├── mol1_bonds.dat
│   ├── mol1.mol
│   ├── mol1_neighbors.cif
│   ├── mol1_neighbors_mid.cif
│   ├── mol2_bonds.dat
│   ├── mol2.mol
│   ├── mol2_neighbors.cif
│   ├── mol2_neighbors_mid.cif
│   ├── neighbor.dat
│   ├── re_lambda_4P_files-e.dat
│   ├── re_lambda_4P_files-h.dat
│   ├── trans_int_files.dat
│   ├── uc_H.inp
│   └── uc_L.inp
├── evc
│   ├── mol1
│   │   ├── elec
│   │   │   └── job_get_NM.local
│   │   └── hole
│   │       └── job_get_NM.local
│   └── mol2
│       ├── elec
│       │   └── job_get_NM.local
│       └── hole
│           └── job_get_NM.local
├── HL
│   ├── 2mol-1-10.com
│   ├── 2mol-1-11.com
│   ├── 2mol-1-12.com
│   ├── 2mol-1-13.com
│   ├── 2mol-1-14.com
│   ├── 2mol-1-1.com
│   ├── 2mol-1-2.com
│   ├── 2mol-1-3.com
│   ├── 2mol-1-4.com
│   ├── 2mol-1-5.com
│   ├── 2mol-1-6.com
│   ├── 2mol-1-7.com
│   ├── 2mol-1-8.com
│   ├── 2mol-1-9.com
│   ├── 2mol-2-10.com
│   ├── 2mol-2-11.com
│   ├── 2mol-2-12.com
│   ├── 2mol-2-13.com
│   ├── 2mol-2-14.com
│   ├── 2mol-2-1.com
│   ├── 2mol-2-2.com
│   ├── 2mol-2-3.com
│   ├── 2mol-2-4.com
│   ├── 2mol-2-5.com
│   ├── 2mol-2-6.com
│   ├── 2mol-2-7.com
│   ├── 2mol-2-8.com
│   ├── 2mol-2-9.com
```

Cont.

```
│   ├── nei_mol-1-10.com
│   ├── nei_mol-1-11.com
│   ├── nei_mol-1-12.com
│   ├── nei_mol-1-13.com
│   ├── nei_mol-1-14.com
│   ├── nei_mol-1-1.com
│   ├── nei_mol-1-2.com
│   ├── nei_mol-1-3.com
│   ├── nei_mol-1-4.com
│   ├── nei_mol-1-5.com
│   ├── nei_mol-1-6.com
│   ├── nei_mol-1-7.com
│   ├── nei_mol-1-8.com
│   ├── nei_mol-1-9.com
│   ├── nei_mol-2-10.com
│   ├── nei_mol-2-11.com
│   ├── nei_mol-2-12.com
│   ├── nei_mol-2-13.com
│   ├── nei_mol-2-14.com
│   ├── nei_mol-2-1.com
│   ├── nei_mol-2-2.com
│   ├── nei_mol-2-3.com
│   ├── nei_mol-2-4.com
│   ├── nei_mol-2-5.com
│   ├── nei_mol-2-6.com
│   ├── nei_mol-2-7.com
│   ├── nei_mol-2-8.com
│   ├── nei_mol-2-9.com
│   ├── uc_mol-1.com
│   └── uc_mol-2.com
├── jobs
│   ├── job_2mol-1.pbs
│   ├── job_2mol-2.pbs
│   ├── job_nei_mol-1.pbs
│   ├── job_nei_mol-2.pbs
│   ├── job_re-mol1-anion.pbs
│   ├── job_re-mol1-cation.pbs
│   ├── job_re-mol1-neutral.pbs
│   ├── job_re-mol2-anion.pbs
│   ├── job_re-mol2-cation.pbs
│   ├── job_re-mol2-neutral.pbs
│   ├── job_transint.pbs
│   └── job_uc_mol.pbs
├── MC-marcus
│   ├── elec
│   │   ├── get_mob.pbs
│   │   ├── get_mob.py
│   │   ├── prepare-mc.py
│   │   ├── run_mc_batch.py
│   │   └── run_mc.pbs
│   └── hole
│       ├── get_mob.pbs
│       ├── get_mob.py
│       ├── prepare-mc.py
│       ├── run_mc_batch.py
│       └── run_mc.pbs
├── mol1.mol
├── mol2.mol
├── momap.inp
```

Cont.

```
├── RE
│   ├── evc-e-mol1.inp
│   ├── evc-e-mol2.inp
│   ├── evc-h-mol1.inp
│   ├── evc-h-mol2.inp
│   ├── re-mol1-anion.com
│   ├── re-mol1-anion_nC.com
│   ├── re-mol1-cation.com
│   ├── re-mol1-cation_nC.com
│   ├── re-mol1-neutral.com
│   ├── re-mol1-neutral_eN.com
│   ├── re-mol1-neutral_hN.com
│   ├── re-mol2-anion.com
│   ├── re-mol2-anion_nC.com
│   ├── re-mol2-cation.com
│   ├── re-mol2-cation_nC.com
│   ├── re-mol2-neutral.com
│   ├── re-mol2-neutral_eN.com
│   └── re-mol2-neutral_hN.com
├── RUN
├── run.sh
└── scr
    ├── copy_neutral_chk_files.py
    ├── copy_RE_lambda_4P_files.py
    ├── gaussian_log_check.py
    ├── get_RE_lambda_4P.py
    ├── get_RE_lambda_evc.py
    ├── get_RE.py
    ├── get_transint.py
    ├── merge_evc_NM_output.py
    ├── mol_one_nei.py
    ├── mol_one_uc.py
    ├── mol_RE_anion_cation.py
    ├── mol_RE_lambda_4P.py
    ├── mol_RE_neutral.py
    ├── mol_two.py
    ├── momap.sh
    ├── prepare_RE.py
    └── run_RE.py

16 directories, 141 files
```

If control parameter `sched_job_array`=0 is set in `momap.inp`, more job scripts will be generated. However, do not be frightened by the sheer number of files, as they are all well-organized.

In `data` directory, the `uc_H.inp` and `uc_L.inp`, input files for HOMO and LUMO determinations, are for internal use only, and may be used to check the correctness of the results. The `mol1.mol` and `mol2.mol` are the separated molecular files of a `cif` file, for example, also can be used to check the correctness of molecule separation.

The file `reorg_einternal_files.dat` is used for reorganization internal energy calculations for molecules in the unit cell, which is used for the onsite energy calculation if the control parameter `lat_site_energy` (default to 0) is set to 1 in the `momap.inp` control file.

The `trans_int_files.dat` is for transfer integral calculations. The other files have the meaning as the name suggests.

Finally, the `config.inp` is the configuration file that the system actually uses. In the default settings, we use scheduling job array (`sched_job_array` = 1), do not include onsite energy (`lat_site_energy` = 0), output only base mobility plus angular resolved mobilities information (`mob_output_level` = 2).

For example, if we do not want to output angular resolved mobilities, we can unset the 2nd bit (`mob_output_level` is a bit-wise setting flag), that is, `mob_output_level` = 0. The parameter `data_mol_output_level` is used to control the output of molecular information in `data` directory, default to 2. It is a bit-wise control parameter, the 1st bit corresponding to output for the 1st molecule, 2nd bit to output for all molecules, 3rd bit to output for the supercell cif file, and 4th bit to output for the Gaussian oniom input files. All the bit setting can be combined, for example, to output all information, we can set the parameter to 1+2+4+8 = 15, that is , `data_mol_output_level = 15`.

The `evc` directory is a work directory for reorganization energy related calculations, which uses data from the `RE` directory to do the calculations.

The `HL` directory is for transfer integral calculations. The naming convention is as follows:

- The "`uc_mol-`" prefix is for single molecule in the central unit cell, thus the `uc_mol-1.com` and `uc_mol-2.com` are two Gaussian input files for the central unit cell.
- The "`nei_mol-`" prefix is for single molecule in the neighbor unit cells, say, `nei_mol-1-6.com` means the 6th neighbor molecule (the specific cell index is specified in the `neighbor.dat` file) of the 1st molecule in the central unit cell.
- The "`2mol-`" prefix is for two-molecule-pair (dipole), say, `2mol-1-12.com`, which means the 1st molecule in the central unit cell and the 12th neighbor molecule of this 1st molecule in central cell are combined to form the Gaussian input file.

The `jobs` directory is used for QC (Gaussian) calculations in directories `HL` and `RE`, we use scheduling job array option as the default option to reduce the number of job scripts, which is controlled by the control parameter `sched_job_array`. These job scripts are called by the python

scripts in `scr` directory.

The `RE` directory is for reorganization calculations. The control files for `evc.exe` are also put in this directory. The normal mode calculations are done with these three directories, that is, `evc`, `RE`, and `scr`.

The `RUN` directory is a directory where the running locks are put, users can check this directory for system progress. If an error occurs, `ERROR` flag will be set up in the `RUN` directory, which can be used to trace where the error occurs.
The `scr` directory is a directory where the python scripts are put, these python scripts are called by the job manager `momap.py`. The sequential executions are logged into the `momap.log`, for example, if the output is redirected to `momap.log`.
The logging information for `transport_prepare.exe` may looks like some things as follows:

```
Do transport preparation ...

>> Run at directory: .

$> transport_prepare.exe
****** Perform Transport Preparation ...
Reading config file "momap.inp" ...
Reading crystal file "150h.cif" ...
  Cell lattice a = 5.9575
  Cell lattice b = 7.4678
  Cell lattice c = 38.764
  Cell lattice alpha = 90
  Cell lattice beta  = 90.067
  Cell lattice gamma = 90
  natoms_cif = 92
  First atom: C1 C 0.09200 0.35760 0.02220
  ...
  Last  atom: H92 H 0.15250 0.78170 0.41950
  Unit cell nmols  = 2
  Unit cell natoms = 46 46
Crystal file 150h.cif parsing done.
Writing config file "data/config.inp" ...

**** MOMAP Build Neighbor List ****
  Neighbor rcutoff distance: 7
  Neighbor search cell (-/+): 3 3 3
**** End of MOMAP Build Neighbor List ****

****** MOMAP Transport Preparation Successfully Done.

--- Normal end for transport preparation.
```

In the logging file, we can see where the job is run and what job is run.

From the above logging information, we know the job is run at the current working directory . (dot) (note the line begins with >>), the job run is `transport_prepare.exe` (note the line begins with $>).

Finally, if the command is run successfully, at the end of logging information for this program, there will appear a line, like, "`--- Normal end for transport preparation`".

There may be other information which may be useful to users in the logging file.

However, to ensure the computing efficiency, in actual calculations, we normally set both control parameters `HL_unique_mol` and `RE_unique_mol` to 1, which is also the default setting.

Note that users may tune the parameter `bond_dis_scale` (default to 1.15) when molecular separation with `cif` file is failed.

Now, by running the `transport_prepare.exe`, the screen output is as follows:

```
[test1]$ transport_prepare.exe
****** Perform Transport Preparation ...
Reading config file "momap.inp" ...
Reading molecular file "mol1.mol" ...
Reading molecular file "mol2.mol" ...
Writing config file "data/config.inp" ...

…

Crystal file naphthalene.cif parsing done.
Make whole molecules...
  molecule 1 COM =  0.000000  0.000000 -0.000000
  molecule 2 COM =  0.500000  0.500000  1.000000
Writing config file "data/config.inp"...

**** MOMAP Build Neighbor List ****
  Neighbor rcutoff distance: 4
  Neighbor search cell (-/+): 3 3 3
**** End of MOMAP Build Neighbor List ****

*** Check Duplicate 2mol pairs ***
  Unique 2mol pairs: 5 out of 28
      1   1   <=>    1
      1   5   <=>    5
      1   7   <=>    7
      1  11   <=>   11
      1  13   <=>   13
*** End of Check Duplicate 2mol pairs ***

**** Check RE Duplicate Molecules in Unit Cell ****
  Unit cell molecule indexes:  1 1
  Unit cell unique molecule indexes: 1
**** End of Check RE Duplicate Molecules in Unit Cell ****

**** Check HL Duplicate Molecules in Unit Cell ****
  Unit cell molecule indexes:  1 1
  Unit cell unique molecule indexes: 1
**** End of Check HL Duplicate Molecules in Unit Cell ****

****** MOMAP Transport Preparation Successfully Done.
```

Here, only the **unique** molecules and molecular pairs (dipoles) are selected for the transfer integral

and reorganization energy calculations, which greatly reduces the computing time for QC job calculations. The full directory and file tree is shown as below:

```
[test1]$ tree ./
./
├── data
│   ├── cellvec.dat
│   ├── config.inp
│   ├── mol1_bonds.dat
│   ├── mol1.mol
│   ├── mol1_neighbors.cif
│   ├── mol1_neighbors_mid.cif
│   ├── mol2.mol
│   ├── mol2_neighbors.cif
│   ├── mol2_neighbors_mid.cif
│   ├── neighbor.dat
│   ├── re_lambda_4P_files-e.dat
│   ├── re_lambda_4P_files-h.dat
│   ├── trans_int_files.dat
│   ├── uc_H.inp
│   ├── uc_L.inp
│   └── unique_id_map.dat
├── evc
│   └── mol1
│       ├── elec
│       │   └── job_get_NM.local
│       └── hole
│           └── job_get_NM.local
├── HL
│   ├── 2mol-11.com
│   ├── 2mol-13.com
│   ├── 2mol-1.com
│   ├── 2mol-5.com
│   ├── 2mol-7.com
│   ├── nei_mol-11.com
│   ├── nei_mol-13.com
│   ├── nei_mol-1.com
│   ├── nei_mol-5.com
│   ├── nei_mol-7.com
│   └── uc_mol-1.com
├── jobs
│   ├── job_2mol.local
│   ├── job_nei_mol.local
│   ├── job_re-mol1-anion.local
│   ├── job_re-mol1-cation.local
│   ├── job_re-mol1_lambda_4P.local
│   ├── job_re-mol1-neutral.local
│   ├── job_transint.local
│   └── job_uc_mol.local
├── MC-marcus
│   ├── elec
│   │   ├── get_mob.local
│   │   ├── get_mob.py
│   │   ├── prepare-mc.py
│   │   ├── run_mc_batch.py
│   │   └── run_mc.local
│   └── hole
│       ├── get_mob.local
│       ├── get_mob.py
│       ├── prepare-mc.py
│       ├── run_mc_batch.py
│       └── run_mc.local
├── momap.inp
├── naphthalene.cif
```

Cont.

```
├── RE
│   ├── evc-e-mol1.inp
│   ├── evc-h-mol1.inp
│   ├── re-mol1-anion.com
│   ├── re-mol1-anion_nC.com
│   ├── re-mol1-cation.com
│   ├── re-mol1-cation_nC.com
│   ├── re-mol1-neutral.com
│   ├── re-mol1-neutral_eN.com
│   └── re-mol1-neutral_hN.com
├── RUN
├── run.sh
└── scr
    ├── copy_RE_lambda_4P_files.py
    ├── gaussian_log_check.py
    ├── get_RE_lambda_4P.py
    ├── get_RE_lambda_evc.py
    ├── get_RE.py
    ├── get_transint.py
    ├── merge_evc_NM_output.py
    ├── mol_one_nei.py
    ├── mol_one_uc.py
    ├── mol_RE_anion_cation.py
    ├── mol_RE_lambda_4P.py
    ├── mol_RE_neutral.py
    ├── mol_two.py
    ├── momap.sh
    ├── prepare_RE.py
    └── run_RE.py

13 directories, 76 files
```

As we only calculate the *unique* molecules and molecular pairs (dipoles), we need to map these unique molecules and molecular pairs to the original molecules and molecular pairs, the mapping information is put in file `unique_id_map.dat` under `data` directory:

```
[test1]$ cat unique_id_map.dat
# Unique ID mapping
2
  14
      1   1    1
      1   2    2
      1   3    3
      1   4    4
      1   5    5
      1   6    6
      1   7    7
      1   8    8
      1   9    9
      1  10   10
      1  11   11
      1  12   12
      1  13   13
      1  14   14
  14
      2   1   15
      2   2   16
      2   3   17
      2   4   18
      2   5   19
      2   6   20
      2   7   21
      2   8   22
      2   9   23
      2  10   24
      2  11   25
      2  12   26
      2  13   27
      2  14   28
```

The first line is comment, the 2nd line is the number of molecules in the central unit cell, then follows number of neighbors for each central unit cell molecule and ID mapping data, which repeats the number of molecules in the central unit cell. For the three-column data in the above table, the first column is the central unit cell molecule ID, the second column is the neighbor ID for the corresponding central unit cell molecule, and the third column is the *uniformly* numbered IDs for the whole central unit cell.

Thus, for example, a file `2mol-13.com` has a uniform ID 13, which corresponds to central unit cell molecule ID 1 and neighbor molecule ID 13, as show in the above list. As another example, if we have a file `2mol-24.com`, from the above list, we know it corresponds to the central unit cell molecule ID 2 and neighbor molecule ID 10.

10.2 Transfer Integral Calculations

Once the transport preparation is done, all the basic files and directories are created.

The next step is to do the transfer integral calculations.

The work is done by calling two python scripts in `scr` directory, that is, `mol_one.py` and `mol_two.py`, to do the one-molecule single point energy calculations and two-molecule single point energy calculations. These two python scripts set up running locks and submit jobs for the transfer integral calculations.

The logging information may look like the following:

```
Submit jobs for transfer integral calculations ...

>> Run at directory: .

$> python scr/mol_one.py

9486[].ntestq
9487[].ntestq
9488[].ntestq

$> python scr/mol_two.py

9489[].ntestq
9490[].ntestq

--- Normal end for transfer integral calculations.
```

When a job is finished, it will remove the corresponding lock automatically.

10.3 Reorganization Energy Calculations

When the transfer integral calculations finish, all related locks on transfer integral calculations will be cleaned. The MOMAP manager will soon submit jobs for reorganization energy calculations. Comparing to the transfer integral calculations, this step takes more time to finish.
The logging information is as follows:

```
Submit jobs for reorganization energy calculation ...

>> Run at directory: .

$> python scr/mol_RE.py

9491.ntestq
9492.ntestq

--- Normal end for reorganization energy calculations.
```

Again, when a job is finished, it will remove the corresponding lock automatically.
More options can be added in this step. For example, if one would like to optimize the anion and cation state geometries based on the optimized neutral state geometry, one can set the parameter `RE_use_neutral_chk` to 1, that is, `RE_use_neutral_chk = 1`. Also, if one would like to calculate the reorganization energies by using the Nelson four-point method, one can set the

parameter `RE_calc_lambda_4P` to 1, that is `RE_calc_lambda_4P = 1`. This can be used to check if the reorganization energy in `evc` calculation is reliable.

10.4 Collect Transfer Integrals

In this step, by calling the python script `scr/get_transint.py`, we obtain the transfer integral data, `VH.dat` and `VL.dat`, for the later transfer hopping rate calculations.

The logging information can be some things as follows:

```
Collect transfer integrals ...

>> Run at directory: .

$> python scr/get_transint.py

9493[].ntestq

--- Normal end for collecting transfer integrals.
```

The above calculations actually use information listed in files `trans_int_files.dat` under `data` directory. For example, the contents of `trans_int_files.dat` may look as follows:

```
[test1]$ cat trans_int_files.dat
2
14
../HL/2mol-1.log  ../HL/uc_mol-1.log  ../HL/nei_mol-1.log  1 2
../HL/2mol-1.log  ../HL/uc_mol-1.log  ../HL/nei_mol-1.log  1 2
../HL/2mol-1.log  ../HL/uc_mol-1.log  ../HL/nei_mol-1.log  1 2
../HL/2mol-1.log  ../HL/uc_mol-1.log  ../HL/nei_mol-1.log  1 2
../HL/2mol-5.log  ../HL/uc_mol-1.log  ../HL/nei_mol-5.log  1 1
../HL/2mol-5.log  ../HL/uc_mol-1.log  ../HL/nei_mol-5.log  1 1
../HL/2mol-7.log  ../HL/uc_mol-1.log  ../HL/nei_mol-7.log  1 2
../HL/2mol-7.log  ../HL/uc_mol-1.log  ../HL/nei_mol-7.log  1 2
../HL/2mol-7.log  ../HL/uc_mol-1.log  ../HL/nei_mol-7.log  1 2
../HL/2mol-7.log  ../HL/uc_mol-1.log  ../HL/nei_mol-7.log  1 2
../HL/2mol-11.log  ../HL/uc_mol-1.log  ../HL/nei_mol-11.log  1 1
../HL/2mol-11.log  ../HL/uc_mol-1.log  ../HL/nei_mol-11.log  1 1
../HL/2mol-13.log  ../HL/uc_mol-1.log  ../HL/nei_mol-13.log  1 1
../HL/2mol-13.log  ../HL/uc_mol-1.log  ../HL/nei_mol-13.log  1 1
14
../HL/2mol-1.log  ../HL/uc_mol-1.log  ../HL/nei_mol-1.log  2 1
../HL/2mol-1.log  ../HL/uc_mol-1.log  ../HL/nei_mol-1.log  2 1
../HL/2mol-1.log  ../HL/uc_mol-1.log  ../HL/nei_mol-1.log  2 1
../HL/2mol-1.log  ../HL/uc_mol-1.log  ../HL/nei_mol-1.log  2 1
../HL/2mol-5.log  ../HL/uc_mol-1.log  ../HL/nei_mol-5.log  2 2
../HL/2mol-5.log  ../HL/uc_mol-1.log  ../HL/nei_mol-5.log  2 2
../HL/2mol-7.log  ../HL/uc_mol-1.log  ../HL/nei_mol-7.log  2 1
../HL/2mol-7.log  ../HL/uc_mol-1.log  ../HL/nei_mol-7.log  2 1
../HL/2mol-7.log  ../HL/uc_mol-1.log  ../HL/nei_mol-7.log  2 1
../HL/2mol-7.log  ../HL/uc_mol-1.log  ../HL/nei_mol-7.log  2 1
../HL/2mol-11.log  ../HL/uc_mol-1.log  ../HL/nei_mol-11.log  2 2
../HL/2mol-11.log  ../HL/uc_mol-1.log  ../HL/nei_mol-11.log  2 2
../HL/2mol-13.log  ../HL/uc_mol-1.log  ../HL/nei_mol-13.log  2 2
../HL/2mol-13.log  ../HL/uc_mol-1.log  ../HL/nei_mol-13.log  2 2
```

The first line contains the number of molecules in unit cell. Then follows a number of neighbors for that each molecule in the central unit cell, plus the three-file-group listing and the two molecular IDs. These files are used by the MOMAP command `transport_transferintegral.exe`.

10.5 Analyze Reorganization Energies

In this step, we further split the calculation into three parts: `prepare_RE.py`, `run_RE.py`, and `get_RE.py`. The first part is to prepare input files for the `evc.exe` program to do normal mode calculations, the second part is using the scheduling system to do the actual calculations, while the third part is to collect the calculated results and put the results into places.
The logging information may look as follows:

```
Analyze reorganization energies ...

>> Run at directory: .

$> python scr/prepare_RE.py

Prepare for evc calculations ...
  treating mol01 ...
  treating mol02 ...

$> python scr/run_RE.py

9494.ntestq
9495.ntestq
9496.ntestq
9497.ntestq

$> python scr/get_RE.py

Calculating NM data ...
  treating mol01 ...
  treating mol02 ...

--- Normal end for analyzing reorganization energies.
```

10.6 Monte Carlo (MC) simulations

Once the above preparation work is done, we can do MC simulations to calculate the charge carrier mobilities.

This step is also split into two parts, that is, `prepare-mc.py` and `run_mc_batch.py`. The first part is to copy the obtained related input files (e.g., VH.dat, VL.dat, NM-e.dat NM-h.dat) into the MC working directory, and do the hopping rate calculations. The second part is to submit jobs to the scheduling (batching) system to do the MC simulations. As the MC program runs, the track files are written out into `tracks` directory. Normally, 2000 tracks will generate fairly good mobility results.

In this step, we normally take advantage of the OpenMP parallelization capability, it linearly scales with the number of cores in a node. For example, if the running node has 28 cores, the performance gain is 28 times comparing to the same serial job.

The logging information may look as follows:

```
Run Monte Carlo (MC) simulations ...

>> Run at directory: MC-marcus/elec

$> python prepare-mc.py

*** Perform Transport Hopping Rate Calculation ...
Reading config file "momap.inp" ...
Reading crystal file " naphthalene.cif" ...
*** Successfully done.

$> python run_mc_batch.py

9498.ntestq

>> Run at directory: MC-marcus/hole

$> python prepare-mc.py

*** Perform Transport Hopping Rate Calculation ...
Reading config file "momap.inp" ...
Reading crystal file "  naphthalene.cif" ...
*** Successfully done.

$> python run_mc_batch.py

9499.ntestq

--- Normal end for Monte Carlo (MC) simulations.
```

The ratetype supported are Marcus, quantum, and egelst.

10.7 Calculate Random Walk Mobilities

Once the MC simulations finish, we can calculate the random walk mobilities from the MC track files by using the Einstein relationship.

The logging information is as follows:

```
Calculate Random Walk mobilities ...

>> Run at directory: MC-marcus/elec

$> python get_mob.py

9500.ntestq

>> Run at directory: MC-marcus/hole

$> python get_mob.py

9501.ntestq

--- Normal end for Random Walk mobilities.
```

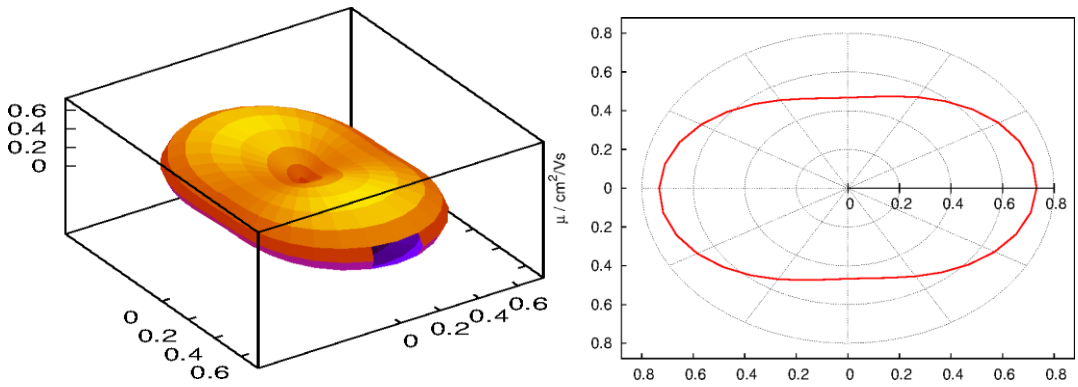
Depends on the `do_` options we selected, there may be temperature dependent MC simulations and mobility calculations, or ME related calculations, but the procedures are similar.

In the MC calculation directories, once the ps2png is properly installed, we can use the following commands to generate and display the 3D and 2D mobility plots:

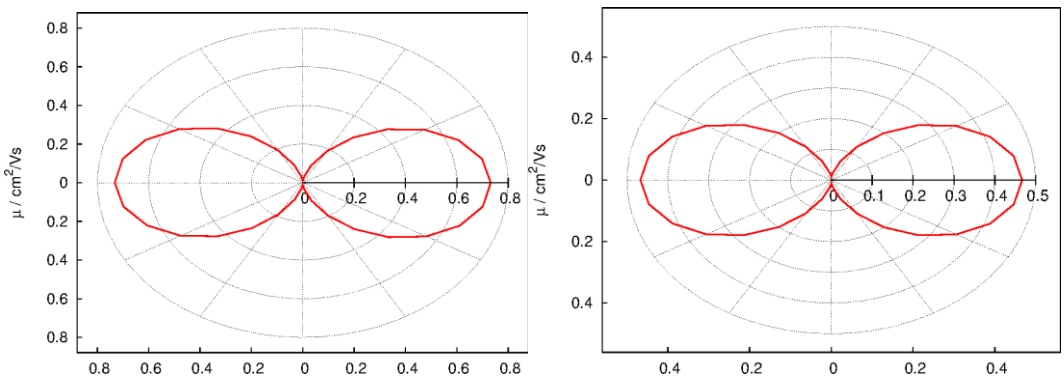
```
$> gnuplot *.gnu
$> ps2png *.eps
$> display *.png
```

If the installed gnuplot supports `term pngcairo`, we can simply run:

```
$> gnuplot *.gnu-png
$> display *.png
```



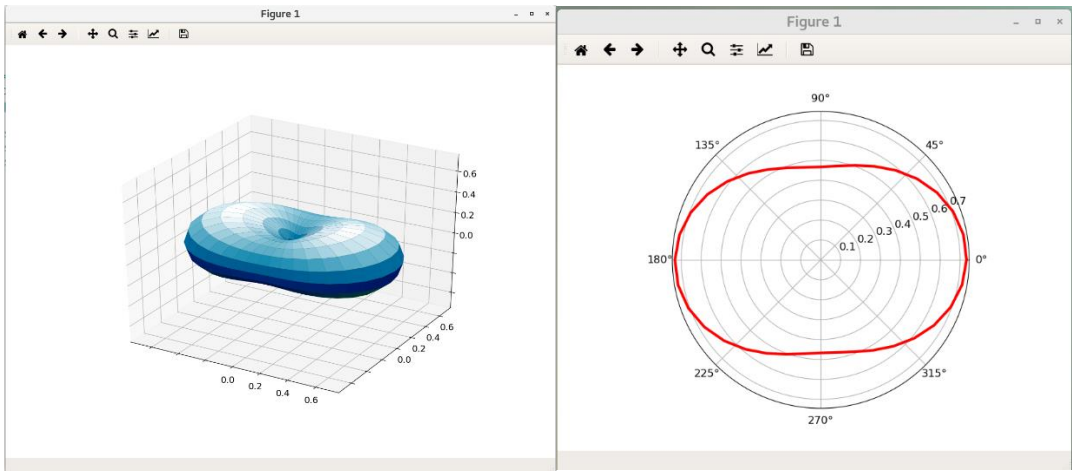
3D plot Plane xy



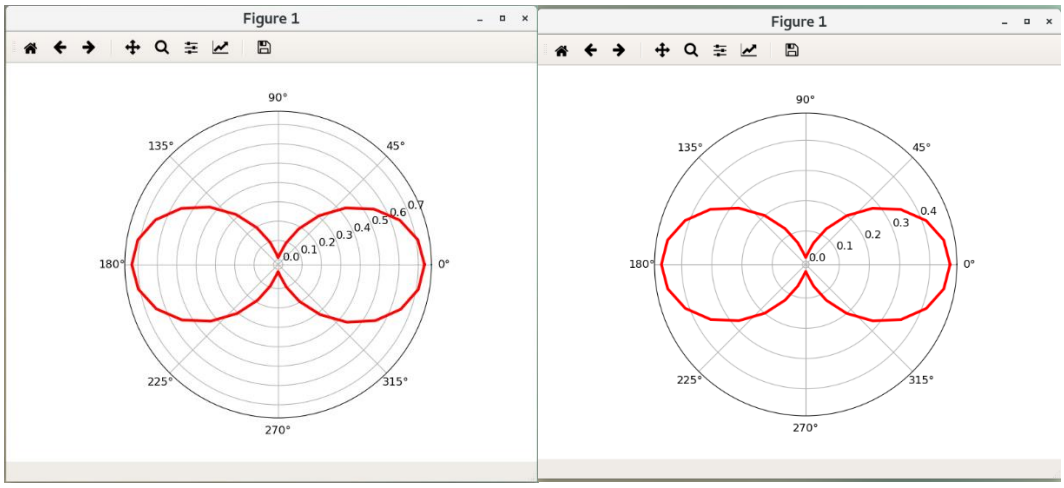
Plane xz Plane yz

Fig. 10-1 The 3D and 2D plots for the electron case by using gnuplot

In addition, if `numpy` and `matplotlib` packages are installed with python, we can also use the generated python scripts to display the mobility plots. The corresponding python scripts in running MC directories are: `mob_direction_all.py`, `mob_plane_xy.py`, `mob_plane_xz.py` and `mob_plane_yz.py`. For example, the 3D and 2D plots for the electron case are shown as follows:



3D plot Plane xy



Plane xz Plane yz

Fig. 10-2 The 3D and 2D plots for the electron case by using matplotlib with python

10.8 Gather data

As all the calculations finish, the results are gathered to the file `momap.dat` as follows.

```
 __  __    ___    __  __     _     ____
|  \/  |  / _ \  |  \/  |   / \   |  _ \
| |\/| | | | | | | |\/| |  / _ \  | |_) |
| |  | | | |_| | | |  | | / ___ \ |  __/
|_|  |_|  \___/  |_|  |_|/_/   \_\|_|      Version 2021A (2.3.1)

 Copyright (c) 2017 Shuaigroup @ Tsinghua University &
       Institute of Chemistry, Chinese Academy of Sciences.
       All Rights Reserved.

Running configuration:
  data/config.inp

Separated molecular information:
  data/mol1.mol
  data/mol2.mol

Neighbor information:
  data/neighbor.dat
  data/mol1_neighbors.cif
  data/mol2_neighbors.cif

Transfer integral information:
  data/VH.dat
  data/VL.dat

Reorganization energy information:
  data/NM-e.dat
  data/NM-h.dat

**** Hopping rates for MC-marcus/elec:
  MC-marcus/elec/hoprates.dat

**** Mobility data for MC-marcus/elec
mob_a  / error [cm**2/Vs]:  6.283116e+00    6.268443e-01
mob_b  / error [cm**2/Vs]:  4.462247e+00    4.881686e-01
mob_c  / error [cm**2/Vs]:  8.591702e-06    7.566622e-07
mob_av / error [cm**2/Vs]: 3.581787e+00    1.879785e-01
Directional mobilities are in file:
  MC-marcus/elec/mob_direction_all.dat
**** End of Mobility data for MC-marcus/elec

**** Hopping rates for MC-marcus/hole:
  MC-marcus/hole/hoprates.dat

**** Mobility data for MC-marcus/hole
mob_a  / error [cm**2/Vs]:  5.832380e+00    5.140441e-01
mob_b  / error [cm**2/Vs]:  5.071254e+00    4.927727e-01
mob_c  / error [cm**2/Vs]:  7.975353e-06    6.495725e-07
mob_av / error [cm**2/Vs]: 3.634544e+00    2.263642e-01
Directional mobilities are in file:
  MC-marcus/hole/mob_direction_all.dat
**** End of Mobility data for MC-marcus/hole

Normal end of MOMAP data gathering.
```

Finally, the job is done!

11. Transport: Parameters

11.1 Transport environment variables

• **MOMAP_ROOT**	Must be set before running MOMAP!
• **MOMAP_LICENSE**	If not set, then $MOMAP_ROOT/license/hzwtec.lic is assumed.
• **MOMAP_SSH_PORT**	(default to 22)
• **MOMAP_LICENSE_DEBUG**	(default to `FALSE`, setting to `TRUE` will write license debug file ‘`momap_lic_log.txt`’ under directory `/tmp`)
• MOMAP_JOB_SCHED	Optional
• MOMAP_JOB_QUEUE	Optional
• MOMAP_QC_EXE	Optional
• MOMAP_QC_PPN	Optional
• MOMAP_QC_NODES	Optional
• MOMAP_MODULE_QC	Optional

11.2 Transport input variables

do_transport_prepare	Default to 1, can be 0 (disable) or 1 (enable)
do_transport_submit_HL_job	Default to 1, can be 0 or 1
do_transport_get_transferintegral	Default to 1, can be 0 or 1
do_transport_submit_RE_job	Default to 1, can be 0 or 1
do_transport_get_re_evc	Default to 1, can be 0 or 1
do_transport_run_MC	Default to 1, can be 0 or 1
do_transport_get_mob_MC	Default to 1, can be 0 or 1
do_transport_run_MC_temp	Default to 0, can be 0 or 1
do_transport_get_mob_MC_temp	Default to 0, can be 0 or 1
do_transport_run_ME	Default to 0, can be 0 or 1
do_transport_get_mob_ME	Default to 0, can be 0 or 1
do_transport_run_ME_temp	Default to 0, can be 0 or 1
do_transport_get_mob_ME_temp	Default to 0, can be 0 or 1
do_transport_gather_momap_data	Default to 1, can be 0 or 1
sched_type	Scheduling type, can be pbs, slurm, lsf, or local.
queue_name	Scheduling queue name
sched_job_array	Default to 1, can be 0 or 1, if use job array in scheduling
nodelist	Default to empty, can be, nodelist = "c01,c02" to designate the computing nodes to be used.
compute_engine	Default to 1, can be 1 (Gaussian), 2 (ORCA), 3 (Q-Chem)
qc_method	Default to "b3lyp"
qc_basis	Default to "b3lyp STO-3g" Zindo calculation is supported by using "int=zindos" as input for qc_basis, can be used for large system calculations.
qc_basis_re	For reorganization calculation if set, otherwise default to qc_basis.
qc_exe	Default to g09
qc_memory	Default to 4096, in MB
qc_nodes	Default to 1
qc_ppn	Default to 8

scratch	Default to /tmp, only valid for ORCA calculation.
module_qc	Use module to setup QC running environment
start_temp end_temp delta_temp	Default to 200 Default to 300 Default to 50
chargetype evc_type use_evc_dint	Default to a, can be e: electron, h: hole or a: all Default to cart, can be cart or int Default to 0, if this variable is set to 1, then use dint to calculate reorganization energy.
vector_a vector_b vector_c lat_cutoff neighbor_scell super_cell	e.g., 8.098 0.0 0.0 e.g., 0.0 5.975 0.0 e.g., -4.888 0.0 7.138 Default to 4.0 angstroms Default to 3 3 3, for neighbor cell search Default to 5 5 5, for ME calculations
lat_site_energy	Default to 0, for onsite energy calculations
momap_dat_file mob_output_level delta_angle data_mol_output_level	Default to momap-{ratetype}.dat Default to 2, can be 0-base, 2-mob_all Default to 10, for angular resolved mobility calculation. Default to 2 (set the 2nd bit up) 1st bit – output only 1st molecular data files, 2nd bit – output all molecular data files, 3rd bit – output supercell cif file, 4th bit – output Gaussian oniom input files. The bit-wise output control setting can be combined.
temp maxt ratetype sta	Default to 300 Default to 10000 Default to marcus, can be marcus, quantum. Default to 1, if use short-time approximation.
nsimu tsimu tsnap	Default to 2000 Default to 1000 ns Default to 5 ns
iterlimit	Default to 100

nevery_out	Default to 10
HL_unique_mol RE_unique_mol	Default to 1, can be 0 or 1 Default to 1, can be 0 or 1
molecule crystal	Default to “2 mol1.mol mol2.mol” with −mol option Default to “naphthalene.cif”
bond_dis_scale HL_unique_ctrl_ratio	Default to 1.15, used to help separating molecules in cif file, users may tune this parameter for abnormal cases. Default to 0.05, used for judging unique dipoles. |eigval[i] – eigval[j]| / max(|eigval[i]|, |eigval[j]|) < 0.05
RE_use_neutral_chk RE_calc_lambda_4P	Calculate anion and cation state reorganization energies by using neutral state chk file, can be 0 or 1, default to 0. If calculate reorganization energies by using the Nelson four-point method, can be 0 or 1, default to 1.
Thinfilm	Format: thinfilm = dir nuc, here dir can be 0(vector_a), 1(vector_b), and 2(vector_c), nuc is the number of repeating unit cell in dir direction, e.g., thinfilm = 0 2
V_dynamic_disorder	Default to 0, used to control if we need to calculate dynamic disorder of transfer integrals, the per-molecular files VH*-dyn.dat or VL*-dyn.dat are to be provided under data directory.
HOMOLUMO_dynamic_disorder	Default to 0, used to control if we need to calculate dynamic disorder of HOMO/LUMO, files HOMO-dyn.dat or LUMO-dyn.dat are to be provided under data directory.

12. PYSOC input variables

do_pysoc	Default to 0, can be 0 or 1, for PYSOC calculations
sched_type queue_name	Scheduling type, can be pbs, slurm, lsf, or local. Scheduling queue name

qc_exe	Default to g09, can be g09 or g16
qc_memory	Default to 4096, in MB
qc_nodes	Default to 1
qc_ppn	Default to 1
module_qc	Use module to setup QC running environment
pysoc_QM_code	Can be gauss_tddft or tddftb
pysoc_QM_input_file	Used only by gauss_tddft
n_excited_singlets	Default to 0, can be set to, say, 4
n_excited_triplets	Default to 0, can be set to, say, 4
soc_scale	Default to 1.0
cicoeff_thresh	Default to 1.0e-5
bIncludeGroundState	Default to 'True', can be 'True' or 'False'

Note: Use equal sign (=) to assign variables, e.g., temp = 300.