

---

# **BDF-Manual**

**索兵兵, 王聪等**

**Mar 09, 2022**



# SOFTWARE INTRODUCTION

<b>1</b>	<b>BDF 软件简介</b>	<b>1</b>
<b>2</b>	<b>Installation and Operation</b>	<b>3</b>
2.1	Installation instructions . . . . .	3
2.2	Cmake compile BDF . . . . .	4
2.3	Program operation . . . . .	7
<b>3</b>	<b>Input and output formats</b>	<b>13</b>
3.1	Input format of BDF . . . . .	13
3.2	Input format of molecular structure . . . . .	20
3.3	BDF output files . . . . .	24
3.4	Common units and conversions in quantum chemistry . . . . .	24
<b>4</b>	<b>Quick Start</b>	<b>27</b>
4.1	First example RHF calculation for H2O molecules . . . . .	27
4.2	Gaussian basis set . . . . .	40
4.3	Exchange-dependent generalized functions supported by BDF . . . . .	49
4.4	Self-consistent field methods: Hartree-Fock and Kohn-Sham . . . . .	50
4.5	Symmetry and molecular point groups . . . . .	60
4.6	Other techniques for self-consistent field calculations . . . . .	65
4.7	iOI-SCF calculation for large systems and the FLMO method . . . . .	86
4.8	Time-dependent Density Generalized Function Theory . . . . .	107
4.9	Nuclear magnetic resonance shielding constants . . . . .	155
4.10	Relativistic effects . . . . .	159
4.11	QM/MM Combination Approach . . . . .	162
4.12	Structural Optimization and Frequency Calculation . . . . .	169
4.13	Solvation models . . . . .	195
4.14	Point charge model . . . . .	204
4.15	Wave function analysis and single-electron properties . . . . .	205
<b>5</b>	<b>Simple input</b>	<b>207</b>
5.1	简洁输入关键词 . . . . .	207

<b>6</b>	<b>图形界面</b>	<b>213</b>
6.1	初始参数界面	213
6.2	自洽场计算参数界面	215
6.3	结构优化计算参数界面	216
6.4	频率计算参数界面	218
6.5	激发态计算参数界面	219
6.6	分子轨道定域化参数界面	220
6.7	自旋轨道耦合计算参数界面	221
6.8	非绝热耦合计算参数界面	222
<b>7</b>	<b>Module Functions</b>	<b>225</b>
7.1	分子自动分片, FLMO 和 iOI 计算 - AUTOFRAG 模块	225
7.2	对称性及预处理 - COMPASS 模块	228
7.3	单、双电子积分计算 - XUANYUAN 模块	235
7.4	Hartree-Fock 及 Kohn-Sham 自洽场计算 - SCF 模块	238
7.5	含时密度泛函 - TDDFT 模块	251
7.6	分子结构优化 - BDFOPT 模块	260
7.7	Hartree-Fock Gradient - GRAD Module	264
7.8	DFT/TDDFT Gradient and Response Properties - RESP Module	265
7.9	Energy and Charge Transfer - ELECOUP Module	270
7.10	Molecular orbital localization - LOCALMO module	272
7.11	Different Basis Set Expansion Tracks - EXPANDMO Module	276
7.12	Møller-Plesset Second Order Perturbation - MP2 Module	277
7.13	Nuclear magnetic shielding constant calculation - NMR module	278
<b>8</b>	<b>算例说明</b>	<b>281</b>
8.1	示例 1: 计算 SCF 能量梯度、结构优化	281
8.2	示例 2: 自动识别对称性 & 指认对称性	282
8.3	示例 3: DFT 计算	287
8.4	示例 4: 检验非阿贝尔群和骨架矩阵法	288
8.5	示例 5: 开壳层体系	289
8.6	示例 6: 势能面扫描	290
8.7	示例 7: 基于双电子积分 Cholesky 分解的 SCF 计算	291
8.8	示例 8: 基于 RI-J 的 DFT 计算	293
8.9	示例 9: 计算电荷转移, 库仑和交换积分	295
8.10	示例 10: 阿贝尔群对称结构的 TD-DFT 梯度计算	299
8.11	示例 11: DFT 基态梯度计算	300
8.12	示例 12: 非阿贝尔群对称性下进行 TD-DFT 梯度的计算	301
8.13	示例 13: 基于 TDDFT 的非绝热耦合计算	302
8.14	示例 14: 限制性结构优化以及开壳层体系的 SA-TDDFT 计算	304
8.15	示例 15: 计算自旋翻转 (spin-flip) 的 TDA	306
8.16	示例 16: iOI 计算 (基于分片方法的大体系 SCF 计算)	309

8.17 示例 17: 双杂化泛函基态单点能计算 . . . . .	311
<b>9 Applications</b>	<b>313</b>
9.1 穆斯堡尔谱 . . . . .	313
<b>10 Frequently Asked Questions</b>	<b>317</b>
10.1 Restart an interrupted computing task ? . . . . .	317
10.2 How is BDF referenced ? . . . . .	319
10.3 False excitation energy/complex excitation energy problem for TDDFT calculations . . . . .	319
10.4 Available Memory and Computational Efficiency of J and K Operators for TDDFT . . . . .	319
10.5 Calculating segmentation fault with available stack area memory . . . . .	320
10.6 OpenMP Parallel Computing . . . . .	321
10.7 OpenMP' s stack area memory size . . . . .	321
10.8 Intel 2018 Edition Fortran Compiler . . . . .	321
10.9 SCF non-convergence . . . . .	321
10.10 SCF energy is far below the expected value (more than 1 Hartree below the expected value) or SCF energy is displayed as a string of asterisks . . . . .	322
10.11 How to use custom base groups . . . . .	322
<b>11 Citation Notes</b>	<b>323</b>
<b>12 References</b>	<b>325</b>
<b>Bibliography</b>	<b>327</b>



## BDF 软件简介

BDF (Beijing Density Functional) 是一个独立完整、具有完全自主知识产权的量子化学计算软件包，也是国际上第一个基于现代密度泛函理论、能准确计算分子体系基态总能量的完全相对论密度泛函程序（早期的类似程序因数值积分精度太差不能准确计算总能量）。

BDF 的研发始于 1993 年，并于 1997 年正式命名 [1]。最初的想法是对稀土、铜系、过渡金属、超重元素等小分子体系进行高精度计算，考察这些体系中的相对论效应，因此一开始就采用基于 Dirac 算符的完全相对论密度泛函理论 (4C-DFT) 和近乎完备的基函数“数值基+STO” (Slater-type orbital)。正因为如此，BDF 对稀土、铜系、超重元素的计算结果一直被作为检验其他近似相对论方法的基准。BDF 对重元素体系电子、分子结构的计算结果被后续 20 余个实验验证。

2009 年开始引入基于高斯基的解析积分，BDF 进入一个新的发展阶段。

毋庸讳言，BDF 一开始就定位于发展新理论、新方法和新算法的平台，因此是一款“科研软件”。基于 BDF 发展的理论和方法包括：相对论含时密度泛函理论 (4C/ZORA/X2C-TDDFT) [2, 3, 4, 5, 6, 7]、精确二分量 (X2C) 相对论理论 [8, 9, 10]、准四分量 (Q4C) 相对论理论 [10, 11]、自旋分离的 X2C 相对论理论 (sf-X2C+so-DKHn) [12, 13]、多体有效量子电动力学 (eQED) [14, 15]、相对论核磁理论 (4C/X2C-NMR) [16, 17, 18, 19, 20, 21, 22, 23]、相对论核自旋-转动理论 (4C-NSR) [23, 24, 25]、相对论能带理论 (X2C-PBC) [26]、X2C 解析梯度和 Hessian [27]；激发态 HF/KS 方法 (mom) [28]；基于“用分子片合成分子” (F2M) 思想的轨道局域化方案 (FLMO) [28, 29, 30]、亚线性标度含时密度泛函理论 (FLMO-TDDFT) [31]、亚线性标度 NMR 方法 (FLMO-NMR) [32]、迭代轨道相互作用“自下而上”自洽场方法 (iOI) [33]；自旋匹配开壳层含时密度泛函理论 (SA-TDDFT) [34, 35, 36, 37, 38]、自旋反转含时密度泛函理论 (SF-TDDFT) [39]、基态/激发态-激发态非绝热耦合含时密度泛函理论 (NAC-TDDFT) [40, 41, 42]、含时密度泛函理论解析能量梯度 [43]、任意单值/双值点群对称化 [44] 等。

除了上述相对论/非相对论密度泛函、含时密度泛函理论，BDF 还有基于“先静态再动态又静态” (SDS) 思想的波函数电子相关方法 SDSPT2 [45]、SDSCI [45]、iCI [46]、iCIPT2 [47, 48]、iCAS [49]、iCISCF [50]、SOC-iCI、iCI-SOC，以及直接求解大矩阵内部本征态的 iVI 方法 [51, 52] 等等。

2021 年 7 月与鸿之微科技 (上海) 股份有限公司签署协议，双方合作推广 BDF 的商业化。鉴于 BDF 的现状，首期商业化版本将以荧/磷光材料发光机理和材料设计为主要应用目标，因此不包括 4C/X2C 相对论、波函数电子相关、固体能带/核磁等方法。即首期商业化的 BDF 将以 DFT、TDDFT 等特色功能为主，包括

- 基态与激发态 (通过  $\Delta$ SCF) 的能量：Hartree-Fock, Kohn-Sham DFT, 包括 LDA, GGA, meta-GGA, 范围分离泛函, 杂化泛函, 双杂化泛函, 等, 支持色散校正
- 激发态计算：TDDFT 和 TDA (包括 FLMO-TDDFT, SF-TDDFT, X-TDDFT, XAS-TDDFT, 等), 以及

### MOM

- TDDFT 激发态偶极矩
- 自旋轨道耦合: sf-X2C/SA-TDDFT/SOC
- 激发态非绝热耦合: NAC-TDDFT
- 基态、 $\Delta$ SCF 激发态、SA-TDDFT 激发态的解析梯度和半数值 Hessian (振动频率)
- 结构优化: 稳定结构优化, 过渡态优化, 最小能量交点 (MECP), 圆锥交点
- 能量转移, 电子转移积分
- QM/MM
- 隐式溶剂化模型
- FLMO-NMR、基于局域轨道 (FLMO) 性质计算和分析

BDF 的研发任重而道远。其成功与否不仅取决于研发人员的长期努力, 更离不开广大用户的鼓励和支持。

## INSTALLATION AND OPERATION

**Attention:** Ordinary users do not need to read the installation and compilation related content, and can directly skip to *BDF program running* and *BDF graphical interface* .

### 2.1 Installation instructions

#### 2.1.1 Hardware environment

In principle, BDF can be compiled and installed on any Unix and Unix-like operating system. We have tested compilation on some common hardware and software environments. For other hardware platforms, users may encounter some problems due to limitations and defects of the operating system and compiler version. In most cases, users can eventually compile and install BDF software successfully by setting the correct compiler flags and system software paths according to their hardware environment.

To compile the BDF software, one needs at least 2 GB of free disk space, depending on the installation method (e.g., CMake keeps the target file), and the actual size after compilation is about 1.3 GB. To run the test cases of the BDF, one should provide at least 1 GB of disk space for caching the intermediate data of the calculation, depending on the size of the calculation system and the integration method used. The amount of cache space required depends on the size of the computational system and the integration method used (the disk space required for the direct integration algorithm is much smaller than for the traditional mode of storing two-electron integrals). In general, for calculations using the direct integration algorithm, more than 4 GB of disk space should be provided for data caching.

#### 2.1.2 Software environment configuration

The minimum requirements for compilers and mathematical libraries for direct compilation and installation from the source code of a BDF are:

- CMake version 3.15 and above (compile with cmake)
- C++ compiler (supports C++03 and above syntax)
- C compiler

- BLAS/LAPACK math library, the interface needs to be a 64-bit integer
- CMake version 3.15 and above (compile with cmake)
- Python 2.7 and above. Python 2 is not compatible with Python 3, and Python 3 is not yet fully adapted

Usually use GCC 4.6 and above to compile normally.

**Optional:**

- C/C++, Fortran compiler for Intel Parallel Studio XE Cluster
- Optimized BLAS/LAPACK library (such as Intel' s MKL, AMD' s ACML, OpenBLAS, etc.)
- To compile the parallel version of BDF, Openmpi 1.4.1 or above is required
- Compiling BDF for GPU requires OpenCL 1.5 or above, and AMD' s Rocm or Nvidia' s Cuda

## 2.2 Cmake compile BDF

### 2.2.1 1. Intel FORTRAN compiler and GNU GCC / G + + compiler are mixed, linked to MKL math library, and OpenMP parallel support

```
# Set up the compiler
$ export FC=ifort
$ export CC=gcc
$ export CXX=g++
# cmake is automatically executed by the setup command
$ ./setup --fc=${FC} --cc=${CC} --cxx=${CXX} --bdfpro --omp --int64 --mkl sequential
→$1
# Build the BDF in the build directory
$ cd build
# Use the make command to compile BDF, specifying 4 CPUs in parallel with the -j4
→parameter
$ make -j4
# Install BDF
$ make install
# After copying BDF PKG Pro under build to any path, write the correct path in bdfrc,
→such as:
$ BDFHOME=/home/user/bdf-pkg-pro
# Run command
$ $BDFHOME/sbin/bdfdrv.py -r **.inp
```

## 2.2.2 2. GNU compiler gfortran/gcc/g++, links to MKL mathematical libraries, OpenMP parallel support

```
# Set up the compiler
$ export FC=gfortran
$ export CC=gcc
$ export CXX=g++
# cmake is automatically executed by the setup command
$ ./setup --fc=${FC} --cc=${CC} --cxx=${CXX} --bdfpro --omp --int64 --mkl sequential
↳$1
# Build the BDF in the build directory
$ cd build
# Use the command make to compile BDF, specifying 4 CPUs in parallel with the -j4
↳parameter
$ make -j4
# Install BDF
$ make install
# After copying BDF PKG Pro under build to any path, write the correct path in bdfrc,
↳such as:
$ BDFHOME=/home/user/bdf-pkg-pro
# Run command
$ $BDFHOME/sbin/bdfdrv.py -r *.inp
```

## 2.2.3 3. Intel compiler ifort/icc/icpc, linking MKL mathematical libraries, OpenMP parallel support

```
# Set up the compiler
$ export FC=ifort
$ export CC=icc
$ export CXX=icpc
# cmake is automatically executed by the setup command
$ ./setup --fc=${FC} --cc=${CC} --cxx=${CXX} --bdfpro --omp --int64 --mkl sequential
↳$1
# Build the BDF in the build directory
$ cd build
# Use the command make to compile BDF, specifying 4 CPUs in parallel with the -j4
↳parameter
$ make -j4
# Install BDF
$ make install
# After copying BDF PKG Pro under build to any path, write the correct path in bdfrc,
↳such as:
```

(continues on next page)

(continued from previous page)

```
$ BDFHOME=/home/user/bdf-pkg-pro
# Run command
$ $BDFHOME/sbin/bdfdrv.py -r **.inp
```

**Warning:**

1. Gcc compiler version 9.0 and above, mixed with Intel FORTRAN compiler, link program error, because the OpenMP version of Intel FORTRAN compiler lags behind GNU compiler. Therefore, GNU 9.0 and above compilers currently do not support mixed compilation of GNU and Intel compilers.
2. Intel FORTRAN version 2018 compiler has many bugs and should be avoided.

## 2.2.4 4 Compile bdfpro and require to generate HZW license file

The main steps are the same as those in the previous three cases. When running the setup command, you need to add a parameter `--hzwlic`, such as:

```
#Cmake is automatically executed by the setup command
$ ./setup --fc=${FC} --cc=${CC} --cxx=${CXX} --bdfpro --hzwlic --omp --int64 --mkl_
↪sequential $1
```

After running the `make install` command, the following output will be given:

```
Please run command '/home/bsuo/bdf-pkg-pro/bdf-pkg-pro/bin/hzwlic.x /home/bsuo/bdf-
↪pkg-pro/build/bdf-pkg-pro' to generate HZW license!
```

Here, `/home/bsuo/bdf-pkg-pro` is the `bdfpro` source file directory, `/home/bsuo/bdf-pkg-pro/build/bdf-pkg-pro` is the binary installation directory of `bdfpro`. Run command:

```
/home/bsuo/bdf-pkg-pro/bdf-pkg-pro/bin/hzwlic.x /home/bsuo/bdf-pkg-pro/build/bdf-pkg-
↪pro
```

After that, `LicenseNumber.txt` is generated in the `/home/bsuo/bdf-pkg-pro/build/bdf-pkg-pro/license` directory.

## 2.3 Program operation

BDF needs to run under a Linux terminal. To run BDF, one needs to prepare an input file, the exact format of which is described in later sections of the manual. The BDF installation directory under tests/input contains some BDF input examples. Here we will use the test cases that come with BDF as an example and briefly explain how to run BDF first.

Running BDF will use a number of environment variables :

Environment variable	Instructions	Must be set or not
BDFHOME	Specify the installation directory of BDF	Yes
BDF_WORKDIR	The working directory of BDF, that is, the execution directory of the current task	No, set automatically
BDF_TMPDIR	Specifies the cache file storage directory for BDF	Yes
BDFTASK	BDF calculation task name, if entered as h2o.inp, task name is h2o	No, set automatically

### 2.3.1 Run BDF standalone and execute the job with a shell script

Assuming that the user directory is /home/user, BDF is installed in /home/user/bdf-pkg-pro. After prepare the input file `ch2-hf.inp`, you need to prepare a shell script and enter the following

```
#!/bin/bash

export BDFHOME=/home/user/bdf-pkg-pro
export BDF_WORKDIR=./
export BDF_TMPDIR=/tmp/$RANDOM

ulimit -s unlimited
ulimit -t unlimited

export OMP_NUM_THREADS=4
export OMP_STACKSIZE=512M

$BDFHOME/sbin/bdfdrv.py -r $1
```

Name the script `run.sh`, use “`chmod +x run.sh`” to give permission to execute the script, and then execute it as follows.

```
# Create a new folder named test in /home/user
$ mkdir test
$ cd test
```

(continues on next page)

(continued from previous page)

```
# Copy /home/user/bdf-pkg-pro/tests/easyinput/ch2-hf.inp to test folder
$ cp /home/user/bdf-pkg-pro/tests/easyinput/ch2-hf.inp
# Run the submit command in the test directory
$ ./run.sh ch2-hf.inp &> ch2-hf.out&
```

---

**Hint:** When BDF prints the output to standard output, you need to use the redirection command > to direct to the file ch2-hf.out.

---

## 2.3.2 Submitting BDF jobs using the PBS job management system

An example script for a PBS submission BDF job is as follows:

```
#!/bin/bash
#PBS -N jobname
#PBS -l nodes=1:ppn=4
#PBS -l walltime=1200:00:00
#PBS -q batch
#PBS -S /bin/bash

#### Set the environment variables #####
#module load tools/openmpi-3.0.1-intel-socket

#module load compiler/intel-compiler-2020

#### Set the PATH to find your applications #####
export BDFHOME=/home/bbs/bdf-pkg-pro

# Specify the temporary file storage directory where BDF runs
export BDF_TMPDIR=/tmp/$RANDOM

# Specify the Stack memory size for OpenMP
export OMP_STACKSIZE=2G

# Specify the number of available OpenMP threads, which should be equal to the number
↳ defined by ppn
export OMP_NUM_THREADS=4

#### Do not modify this section ! #####
cd $PBS_O_WORKDIR

$BDFHOME/sbin/bdfdrv.py -r jobname.inp
```

### 2.3.3 Submit BDF jobs using Slurm job management system

An example script for slurm to submit a BDF job is as follows :

```
#!/bin/bash
#SBATCH --partition=v6_384
#SBATCH -J bdf.slurm
#SBATCH -N 1
#SBATCH --ntasks-per-node=48

#### Set the environment variables #####
#module load tools/openmpi-3.0.1-intel-socket
#module load compiler/intel-compiler-2020

#### Set the PATH to find your applications #####
export BDFHOME=/home/bbs/bdf-pkg-pro

# Specify the temporary file storage directory where BDF runs
export BDF_WORKDIR=./
export BDF_TMPDIR=/tmp/$RANDOM

# Specify the Stack memory size for OpenMP
export OMP_STACKSIZE=2G

# Specify the number of available OpenMP threads, which should be equal to the number_
↳ defined by ppn
export OMP_NUM_THREADS=4

#### Do not modify this section ! #####
$BDFHOME/sbin/bdfdrv.py -r jobname.inp
```

#### Important:

1. The problem with stacksize. The Intel Fortran compiler requires a large amount of stack memory for programs to run, and the default stacksize is usually too small and needs to be specified by `ulimit -s unlimited`.
2. OpenMP Number of threads in parallel. `OMP_NUM_THREADS` is used to set the number of threads in parallel for OpenMP. BDF relies on OpenMP parallelism to improve computational efficiency. If you are using the Bash Shell, you can use the command `export OMP_NUM_THREADS=N` to specify the number of OpenMP parallel threads to use. to use N OpenMP threads to accelerate the computation.
3. OpenMP available heap memory, users can use `export OMP_STACKSIZE=1024M` to specify the size of the heap memory available to each thread of OpenMP, and the total heap memory size is `OMP_STACKSIZE*OMP_NUM_THREADS`.

### 2.3.4 QM/MM Computing Environment Configuration

We recommend using Anaconda to manage and configure the QM/MM computing environment ( see ).

- Configure the runtime environment in anaconda

```
conda create -name yourEnvname python=2.7
conda activate yourEnvname
#Configure Cython and PyYAML
conda install pyyaml #or pip install pyyaml
conda install cython
```

- Installation and configuration of pDynamo-2

BDF pDynamo-2 has been built into the sbin directory of the installation directory, so run the following commands in the sbin directory to install and configure in the sbin directory to install and configure :

```
cd pDynamo_2.0.0
cd installation
python ./install.py
```

After the installation script is run, two environment configuration files, environment\_bash.com, environment\_cshell.com are generated. Users can load this environment file in their .bashrc via source to set up the runtime environment.

---

**Note:** The compilation process automatically selects the C compiler, for MAC systems it is recommended to install the GCC compiler using homebrew and add CC=gcc-8. Other versions of the gcc compiler correspond to gcc-6 or gcc-7, etc. Other versions of the gcc compiler correspond to gcc-6 or gcc-7, respectively. The version above gcc-8 is not tested at the moment.

---

When pDynamo-2 is run, the qmmrun . sh file in the sbin directory is called by default to perform QM calculations. When configuring the environment, you need to make sure that the sbin directory is in the system PATH. You can add it with the following command.

```
export PATH=/BDFPATH/sbin:$PATH
```

- The final step is to specify the BDF program temporary file storage folder, either by running the following command, or by setting the variable in the environment variable in the environment variable.

```
export PDYNAMO_BDFTMP=YourBDF_tmpPATH
```

To check if pDynamo is installed correctly, you can run the examples that come with the software, which are located in the **pDynamo\_2.0.0/book/examples** directory, by running the following command

```
python RunExamples.py
```



## INPUT AND OUTPUT FORMATS

### 3.1 Input format of BDF

There are three types of BDF input file formats: easy input, advanced input, and mixed input. **Easy input** is easy to use and does not require the user to know much about the details of the calculation, which is a low threshold for beginners. **Advanced input** provides powerful control over BDF calculations, with precise control over each calculation module. **Mixed input** of BDF is a way to add some advanced computational functions to the BDF simple input and add fields that precisely control the behavior of BDF computational modules as defined by the advanced input format, while keeping the BDF simple input convenient and beginner-friendly. In short, BDF hybrid input is a combination of BDF simple input and advanced input. For beginners, most computational tasks can be accomplished using only BDF simple inputs. For users who have a basic understanding of quantum chemistry theory and want to learn and use the more advanced features of BDF in depth, they can learn to use the advanced input of BDF.

**Note:** *Inputs other than \*\*file names\**, **shell** commands and **environment variable names**, such as BDF **module names**, **keywords** and **keyword values**, are not case sensitive.

#### 3.1.1 Easy input for BDF

The BDF simple input format is described in detail using the single-point energy calculation of water molecules as an example:

```
#!/H2O.bdf
B3lyp/3-21G

Geometry # Enter atomic coordinates, in unit Angstrom
O 0.00000    0.00000    0.36827
H 0.00000   -0.78398   -0.18468
H 0.00000    0.78398   -0.18468
End Geometry
```

The BDF easy input consists of 3 input blocks:

**First input block**

The first input block is a single line, starting with `#!` followed by the name of the input script, e.g. `#!name.bdf`, which can be a description text.

**Second input block**

starts with the second line and ends with the line before `Geometry`. This input block, which can consist of multiple lines, is the command control line of the BDF, and is used to specify what computational tasks the BDF does. It can consist of multiple lines, and can be used to specify some basic computational control parameters such as basis group, generalization, charge number and spin multiplicity. The content of the command line is separated by spaces for the different keywords. The keywords and their values are separated by equal signs. A keyword without a value is itself a control keyword. Keywords can have one value or multiple values separated by commas. Keywords can have multiple lines. If `#` appears in a line, the line after `#` is a comment statement.

**Third input block**

Starting from the `Geometry` line and ending at the `End Geometry` line, enter the geometry of the molecule, as described in the input format of the molecular structure.

---

**Tip:**

- The third line in this example is an empty line, a blank line in the BDF input, except between the defined molecular coordinates `Geometry ... End geometry`, all other blank lines are non-essential, but for readability, it is strongly recommended that users use blank lines to separate different input blocks and different modules.

Advanced input for BDF

---

The BDF advanced input is the input mode set during the initial development of the BDF and is characterized by **module-guided computation + module parameter control** in the following format

```
$bdfmodule1
# Comment
Keyword1
  value # inline comment
Keyword2
  value
  ...
$end

%cp $BDFTASK.scforb $BDF_TMPDIR/$BDFTASK.inporb
$bdfmodule2
Keyword1
  value
Keyword2
```

(continues on next page)

(continued from previous page)

```

value
...
$end

```

The description is as follows.

- This input contains two BDF computation modules, `bdfmodule1` and `bdfmodule2` (this is only an example, there are not really modules named `bdfmodule1` and `bdfmodule2` 的模块). A computation task may contain more than one BDF module.
- **Module-directed computation:** means that two computation modules are executed sequentially to complete the computation. The input to each module starts with `$bdfmodule` and ends with the first subsequent occurrence of the `$end` keyword, and between `$bdfmodule` and `$end` are the control keywords and their values for that module. where `bdfmodule` is the name of the BDF calculation module, such as `COMPASS`, `XUANYUAN`, `SCF`, etc.
- **BDF module parameter control:** Each module controls its computation behavior by its own keywords, and the parameter control input uses the **keyword + value**, where the keyword value starts from the next line of the line where the keyword is located, and can be a single line or multiple lines depending on the specific keyword.

The advanced input format of BDF requires a certain understanding of quantum chemical theory and the specific functions of each calculation module of BDF. `bdfdrv.py` written in python language, calls different calculation modules sequentially to complete complex calculation tasks. to complete the complex computational tasks, each module through the temporary files and environment variables to exchange data between.

Here, we take water molecules as an example to describe in detail the BDF advanced input format.

```

#Example for BDF advanced input
$compass
Title
  Water molecule, energy calculation
Geometry
O 0.00000    0.00000    0.36827
H 0.00000   -0.78398   -0.18468
H 0.00000    0.78398   -0.18468
End geometry
Basis # basis sets
  3-21G
Group # C2v point groups, which can be entered without input, are automatically
↳judged by the program, and are often used to specify D2h and its subgroup.
↳calculations for higher-order groups.
  C(2v)
$end

```

(continues on next page)

```

$xuanyuan
$end

$scf
RHF # restricted Hatree-Fock
$end

%cp $BDF_WORKDIR/$BDFTASK.scforb $BDF_TMPDIR/$BDFTASK.inporb
$scf
RKS # restricted Kohn-Sham
DFT
  B3lyp      # B3LYP functional, Notice, it is different with B3lyp in Gaussian.
Guess
  Readmo     # Read orbital from inporb as the initial guess orbital
$end

```

The input file shown above contains four computational modules, **COMPASS**, **XUANYUAN** and two **SCF**. **COMPASS** is used to read in the input molecular coordinates, basis functions and other information, and store them as data structures inside BDF. An important task of **COMPASS** is the processing of molecular point groups, including the determination of molecular symmetry and the generation of symmetry-adapted orbitals. **XUANYUAN** is used to calculate single and double electron integrals. The **SCF** module is then called twice to perform self-consistent field calculations, once for the RHF (Restricted Hatree-Fock) and RKS (Restricted Kohn-Sham).

The input to each computation module follows the “**keyword + value**” format, i.e., a keyword, such as `Group` in **COMPASS**, is given followed by a value for that keyword, in this case `C(2v)`. Some keywords are used for logical control, such as `RHF` in the first **SCF** module, which specifies that the **SCF** module performs the RHF calculation, and no additional input values are needed for such keywords. Some of the keywords require multiple lines of input, as described in the keyword descriptions for each module.

Between the first and second **SCF** module, there is a line starting with `%`. Here, we insert a shell command that performs a task of copying a file. The `$BDFTASK.scforb` file generated by the first **SCF** calculation and placed in `BDF_WORKDIR` is copied to `BDF_TMPDIR` and renamed to `$BDFTASK.inporb`. In the second **SCF** module, we specify the keyword `guess`, with the value `readmo`, i.e., read in the molecular orbitals as an initial guess. In the BDF advanced input, the lines starting with `%` are shell command lines. The lines starting with `#` or containing `#` in the input, all the contents after the `#` are comment statements.

The following flowchart of BDF modules and calculations gives the order in which each module is called.

---

#### Tip:

- A complete computational task requires multiple calls to the BDF computational modules. The order in which the modules appear in the advanced input is given by the **BDF module and calculation flow diagram**. The general calculation task will only involve a small part of the module shown in the figure above, for example, most calculation tasks do not require the `AUTOFRAG` module, and the first calculation module is actually `COMPASS`;

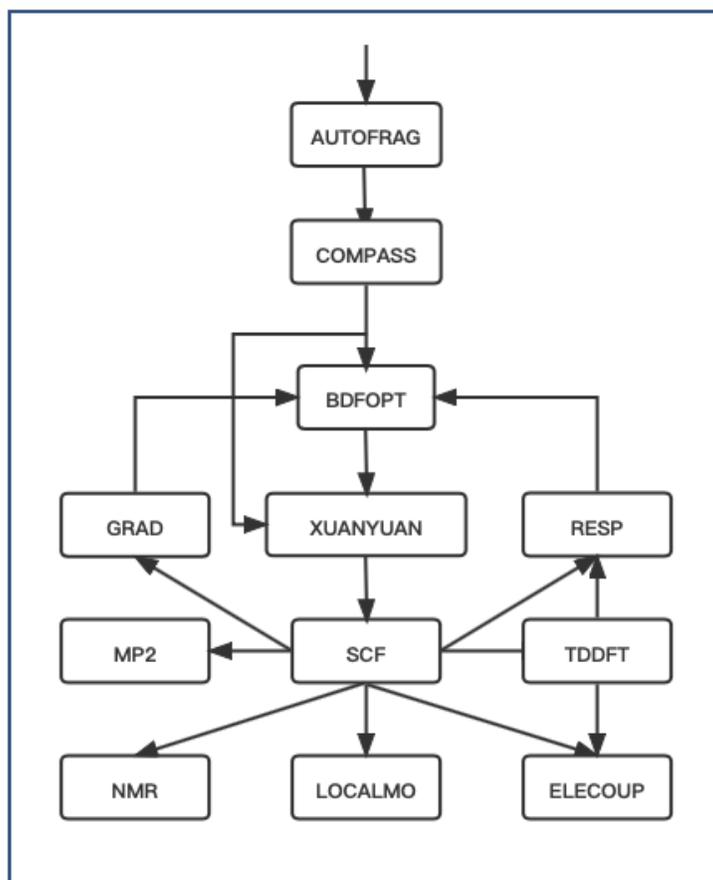


Fig. 3.1: the BDF module and calculation flow diagram

Only for **iOI-SCF** and **FLMO** calculations should the `AUTOFRAG` module appear (and be placed before `COMPASS`) to automatically slice the numerator, and then `COMPASS` and other computational modules should be called to finish the job.

- For example, if the **molecular structure is optimized** by the Kohn-Sham method, the `COMPASS` module preprocesses the molecular structure and the basis group, and then the `BDFOPT` module calls the `XUANYUAN`→`SCF`→`RESP` modules several times in sequence to optimize the molecular structure by calculating the single electron integral, the self-consistent field energy and the gradient of energy to the nucleus coordinates.
- For the actual calculation, the concise input file of the BDF is translated into the advanced input format of the BDF and stored in a hidden file `.bdfinput` in a temporary folder specified by `BDF_TMPDIR`.

The following **BDF modules and menus** give the names and functions of the BDF modules.

Table 3.1: BDF Module and menu

Module name	Function
AUTOFRAG	Automatic molecular fragmentation, driving IOI-SCF and flmo calculations
COMPASS	Molecular structure, basis set and symmetry pretreatment
XUANYUAN	Atomic orbital integral
BDFOPT	Molecular geometry optimization
SCF	Hartree-Fock and Kohn sham self consistent fields
TDDFT	Time dependent density functional calculation
RESP	Hartree-Fock, Kohn sham and TDDFT gradients
GRAD	Hartree-Fock gradients
LOCALMO	Molecular orbital localization
NMR	Calculation of nuclear magnetic shielding constant
ELECOUP	Electron transfer integral, energy transfer integral, localized excited state
MP2	Møller-Plesset second-order perturbation theory

Table 3.2: BDF advanced input description table

Input	Description
<code>\$modulename ... \$end</code>	<code>modulename</code> is the control input for the BDF calculation module, all modulenames are available in the <code>\$BDFHOME/database/program.dat</code> file
<code>#</code>	Lines starting with <code>#</code> or following <code>#</code> in each line are comment statements
<code>*</code>	<code>*</code> is placed at the beginning of the line only, and the lines starting with <code>*</code> are commented out
<code>%</code>	The lines starting with <code>%</code> and ending with <code>%</code> are shell commands, usually used to process intermediate files
<code>&amp;database ...&amp;end</code>	Some complex calculations, such as FLMO, require information such as the definition of molecular fragments, which is usually placed between <code>&amp;database</code> and <code>&amp;end</code> . Please refer to test062

### 3.1.2 Mixed input for BDF

Mixed input combines the simplicity of BDF input with the advanced input format, providing the convenience of BDF simple input and the precise control of BDF computational modules, which is useful when performing complex computations.

The basic structure of a BDF Mixed input file is as follows:

```
#!/name.bdf
Method/functional/basis sets Keyword Keyword = option Keyword = option 1, option 2
Keywords = Options

Geometry
Molecular structure information
End Geometry

$modulename1
...      # Comment statements
$End

$modulename2
...
$End
```

A mixed input file can be divided into 4 input blocks, **the first three of which are formatted for the simple input mode of BDF** and the fourth input block, which is what remains after `End geometry`, is in the same format as the advanced BDF input and is used to provide precise control over the behavior of specific BDF calculation modules, and these parameters are added to the corresponding BDF calculation modules with the highest control priority.

The BDF hybrid input format is described in detail using the cation of water as an example.

```
#!/H2O+.bdf
B3lyp/3-21G iroot=4

Geometry
O 0.00000    0.00000    0.36827
H 0.00000   -0.78398   -0.18468
H 0.00000    0.78398   -0.18468
End Geometry

$scf
Charge # Specify the charge number as +1
1
molden # Export molecular tracks as molden format files
$End
```

The above example adds a line starting with `$scf` and ending with `$end` to control the **SCF** module, in addition to the required BDF simple input. This input is a mix of BDF simple and advanced inputs, and in the input of the **SCF** module, with the keyword `charge` set to 1 for calculating  $\text{H}_2\text{O}^+$  ions and the `moldden` keyword controlling the output of the converged SCF track to a **molden** format file, can be used to visualize molecular structure, orbitals, electron density, analyze wave functions, or calculate single-electron properties. It should be noted that in the second command line of the hybrid input format, `charge = -1` can be used to control the calculation of  $\text{H}_2\text{O}^+$  anions, but if the `charge` keyword is also used in the later input of the `scf` module, the latter has the highest control priority and will override the input in the command line. In other words, in the mixed input format, the advanced input keyword for each BDF calculation module has the highest control priority.

## 3.2 Input format of molecular structure

The molecular structure input of BDF starts from `Geometry` and ends at `End geometry`, and can be entered in three ways: Cartesian, Internal, or specified xyz file format.

**Warning:** The default unit for BDF input coordinates is Å. If you need to enter the molecular structure in atomic units, you need to use the keyword `unit=Bohr`. In BDF's simple input mode, `unit=Bohr` is placed in the second control line. In case of advanced input mode Use the keyword `unit` in the Compass module and specify the value as Bohr, see the example below.

Specify the molecular coordinate units in the control line of the concise input, and enter a bond length of 1.50 Bohr for the  $\text{H}_2$  molecule.

```
#!/ bdfctest.sh
HF/3-21G unit=Bohr

Geometry
  H  0.00 0.00 0.00
  H  0.00 0.00 1.50
End geometry
```

In the advanced input mode, control the molecular coordinate units

```
$compass
Geometry
  H  0.00 0.00 0.00
  H  0.00 0.00 1.50
End geometry
Basis
  3-21G
Unit
```

(continues on next page)

(continued from previous page)

```
Bohr
$end
```

### 3.2.1 Input of Cartesian Coordinate Format for Molecular Structure

```
Geometry # default coordinate unit is angstrom
O 0.00000 0.00000 0.36937
H 0.00000 -0.78398 -0.18468
H 0.00000 0.78398 -0.18468
End geometry
```

### 3.2.2 Input of internal coordinate format for molecular structure

The internal coordinates are entered in the format of defined key length, key angle, and dihedral angle, where the key length is in angstroms and the key angle and dihedral angle are in degrees. Input Examples of input modes are as follows.

```
Geometry
atom1
atom2 1 R12 # R12 is the bond length between atoms 2 and 1
atom3 1 R31 2 A312 # R31 is the bond length between atoms 3 and 1, and
↪A312 is the bond angle defined by atoms 3, 1 and 2
atom4 3 R43 2 A432 1 D4321 # R43 is the bond length between atoms 4 and 3, and
↪a432 is the bond angle defined by atoms 4, 3 and 2, D4321 is the dihedral angle
↪defined by atoms 4, 3, 2 and 1
atom5 3 R53 4 A534 1 D5341 # R53 is the bond length between atoms 5 and 3, and
↪a534 is the bond angle defined by atoms 5, 3 and 4, D5341 is the dihedral angle
↪defined by atoms 5, 4, 3 and 1
...
...
End Geometry
```

Specifically, for water molecules, the internal coordinates are entered as follows.

```
Geometry
O
H 1 0.9
H 1 0.9 2 109.0
End geometry
```

Internal coordinate input, using variables to define the value of internal coordinates as follows ( **Coordinate variables are currently only supported for simple input!** ) :

```

Geometry
O
H 1 R1
H 1 R1 2 A1      # Define the intramolecular coordinates, and the coordinate_
↪values are replaced by variables

R1 = 0.9          # Defines the value of the coordinate variable
A1 = 109.0
End geometry

```

**Warning:**

- Note that the definition of internal coordinates should be kept on a blank line, and the values of internal coordinates and coordinate variables should be separated by a blank line.

Internal coordinate format input, potential energy surface scan as follows ( **currently only simple input supports potential energy surface scan!** ) :

Example 1: Coordinate input for H<sub>2</sub>O , potential energy surface scan, bond length starting at 0.75 Å. The bond length is calculated in steps of 0.05 Å, with 20 points from smallest to largest.

```

Geometry
O
H 1 R1
H 1 R1 2 109     # The intramolecular coordinates are defined, and the OH bond_
↪length is defined as the variable R1

R1 0.75 0.05 20  # Starting value of R1, scanning step size, number of scanning_
↪points. Note to keep the blank line of the previous line
End geometry

```

Example 2: Concise input for H<sub>2</sub>O potential surface scan with bond length starting at 0.75 Å. The bond length is calculated in 0.05 Å steps from smallest to largest 20 points. SCF takes the initial guess track via Read.

```

#! h2oscan.bdf
B3lyp/3-21G Scan Guess=readmo

Geometry
O
H 1 R1

```

(continues on next page)

(continued from previous page)

```
H 1 R1 2 A1 # The intramolecular coordinates are defined, the OH bond length_  
↳is defined as the variable R1, and the Hoh bond angle is A1  
  
A1 = 109.0 # Define the value of the key angle, taking care to leave the_  
↳previous line blank  
  
R1 0.75 0.05 20 # Define the starting value of OH bond length R1, scanning step_  
↳size and scanning points.  
  
End geometry
```

### 3.2.3 Read the molecular coordinates from the specified file

```
Geometry  
file=filename.xyz # Needs to be the file filename.xyz under the current job, only_  
↳xyz format is supported for input.  
  
End geometry
```

### 3.3 BDF output files

File extension	Description
.out	Master output file
.out.tmp	Sub-output files for structural optimization and numerical frequency tasks (output containing calculation steps for energy, gradient, etc.)
.pes1	Molecular structure (E), energy (Hartree) and gradient (Hartree/Bohr) for each step of the structure optimization and numerical frequency task
.egrad1	Energy (Hartree) and gradient (Hartree / Bohr) of the last step of structural optimization and numerical frequency task
.hess	Hessian matrix (Hartree/Bohr <sup>2</sup> )
.unimovib.input	Unimovib input file for thermochemical analysis
.nac	Non-adiabatic coupling vector (Hartree/Bohr)
.chkfil	Temporary documents
.data-punch	Temporary documents
.optgeom	Molecular coordinates in standard orientation (Bohr). For the task of structural optimization, it is the molecular coordinate of the last step of structural optimization
.finaldens	Density matrix for the last step of SCF iteration
.finalfock	Fock matrix for the last SCF iteration
.scforb	Molecular orbitals for the last SCF iteration
.global.scforb	FLMO/iOI calculates the molecular orbitals of the last SCF iteration
.fragment*.*	Output file related to the subsystem calculation of the FLMO/iOI calculation
.ioien-large.out	iOI calculation of subsystem composition information for step 1 and subsequent macro iterations

Some computational tasks may produce other output files not listed above, which are generally temporary files.

### 3.4 Common units and conversions in quantum chemistry

Most of the internal operations of quantum chemistry programs use the atomic unit (a.u.) system. This eliminates the need for unit conversions in various computational formulas, making the code simple and avoiding additional operations and loss of precision. Quantitative programs also generally use the atomic unit system when outputting intermediate data, but most of the data with chemical significance are converted to the usual units.

- Energy 1 a.u. = 1 Hartree

- Mass 1 a.u. =  $1 m_e$  (electron mass)
- Length 1 a.u. = 1 Bohr =  $0.52917720859 \text{ \AA}$
- Electricity 1 a.u. =  $1 e = 1.6022 \times 10^{-19} \text{ C}$
- Electron density 1 a.u. =  $1 e/\text{Bohr}^3$
- Dipole moment 1 a.u. =  $1 e \cdot \text{Bohr} = 0.97174 \times 10^{22} \text{ V/m}^2 = 2.5417462 \text{ Debye}$
- Electrostatic potential 1 a.u. = 1 Hartree/e
- Electric field 1 a.u. =  $1 \text{ Hartree}/(\text{Bohr} \cdot e) = 51421 \text{ V/\AA}$

### 3.4.1 Energy unit conversions

1 unit =	Hartree	$\text{kJ} \cdot \text{mol}^{-1}$	$\text{kcal} \cdot \text{mol}^{-1}$	eV	$\text{cm}^{-1}$
Hartree	1	2625.50	627.51	27.212	$2.1947 \times 10^5$
$\text{kJ} \cdot \text{mol}^{-1}$	$3.8088 \times 10^{-4}$	1	0.23901	$1.0364 \times 10^{-2}$	83.593
$\text{kcal} \cdot \text{mol}^{-1}$	$1.5936 \times 10^{-3}$	4.184	1	$4.3363 \times 10^{-2}$	349.75
eV	$3.6749 \times 10^{-2}$	96.485	23.061	1	8065.5
$\text{cm}^{-1}$	$4.5563 \times 10^{-6}$	$1.1963 \times 10^{-2}$	$2.8591 \times 10^{-3}$	$1.2398 \times 10^{-4}$	1

### 3.4.2 Length unit conversions

1 unit =	Bohr	$\text{\AA}$	nm
Bohr	1	0.52917720859	0.052917720859
$\text{\AA}$	1.88972613	1	0.1
nm	0.188972613	10	1



## QUICK START

This chapter introduces the basic use of the various functions of the BDF, and gives basic examples of calculations and analysis of data readings for specific calculation functions.

This chapter introduces the basic use of the various functions of the BDF, and gives basic examples of calculations and analysis of data readings for specific calculation functions.

### 4.1 First example RHF calculation for H<sub>2</sub>O molecules

Hartree-Fock is the most basic algorithm in quantum chemistry. In this subsection, we will guide the user through a BDF calculation and analyze the input and output information by using an example of Hartree-Fock calculation for a water molecule. Here, we first give the concise inputs to the BDF, and in order to understand the difference between the concise and advanced input modes of the BDF, we also give the advanced input file for each concise input.

#### 4.1.1 Preparing Inputs

First, prepare the input file for the Hartree-Fock calculation of the single-point energy of water molecules, named `h2o.inp`, with the following input:

```
#!/bdf.sh
HF/3-21G

Geometry
O
H 1 R1
H 1 R1 2 109.

R1=1.0      # input bond length with the default unit angstrom
End geometry
```

**The input is interpreted as follows.:**

- The first line must start with `#!` followed by a string named `bdf.sh`, this can be any letter and array of words into a string, can not contain special characters other than `.` in addition to the special characters. The

first line is the system reservation line, the user can use this string to mark the calculation task.

- The second line, HF/3-21G is the calculation parameter control line for the BDF. HF stands for Hartree-Fock, and 3-21G specifies that the calculation uses the 3-21G basis group. The key parameter control line can be multiple consecutive lines.
- The third row is an empty line and can be ignored. It is entered here to distinguish between different inputs and to enhance the readability of the input, and is recommended to be kept by the user.
- The fourth and tenth lines are `Geometry` and `End geometry`, respectively, marking the start and end of the molecular geometry input, and the default unit of coordinates is angstrom.
- The fifth through ninth lines enter the structure of the water molecule in the internal coordinate mode. (See *Internal coordinate format for molecular structure input* for details)

This simple input corresponds to the advanced BDF input as follows:

```
$compass
Geometry
  O
  H 1 1.0
  H 1 1.0 2 109.
End geometry
Basis
  3-21g # set basis set as 3-21g
$end

$xuanyuan
$end

$scf
RHF      # Restrictive Hartree-Fock Method
Charge   # The charge of the molecule is set to 0, and the default calculation is_
↪for neutral molecules with zero charge
  0
Spinmulti # spin-multiplicity 2S+1, the default calculation is for double electron_
↪system
  1
$end
```

As can be seen from the advanced input, BDF will execute the modules **COMPASS**, **XUANYUAN** and **SCF** in order to complete the single-point energy calculation of the water molecule. **COMPASS** is used to read in the basic information such as molecular structure, basis functions, etc., determine the symmetry of the molecule, rotate the molecule to the standard orientation (Standard orientation, see the BDF use of group theory), generate the symmetry-adapted orbitals, etc., and store such information into the `h2o.chkfil`. The keywords in **COMPASS** are

- The molecular structure defined between `Geometry` and `End geometry`;

- `Basis` defines the basis group as 3-21G;

After executing the **COMPASS** module, BDF uses the **XUANYUAN** module to calculate the single and double electron integrals. Since the BDF defaults to the SCF method of **repeated calculation of double electron integrals**, i.e. **Integral Direct SCF** .

Finally, the BDF executes the **SCF** module to complete the Hartree-Fock based self-consistent field calculation.

- The `RHF` specifies the use of the restricted Hartree-Fock method;
- `Charge` specifies that the charge of the system is 0;
- `Spinmulti` specifies that the spin multi of the system is 1.

Here `RHF` is a mandatory keyword, and `Charge` and `Spinmulti` can be ignored for the restricted method.

## 4.1.2 Performing the calculation

To perform the calculation, a shell script named `run.sh` is prepared and placed in the directory where the input file `h2o.inp` is located. The contents are as follows.

```
#!/bin/bash

# Set the BDF installation directory
export BDFHOME=/home/bsuo/bdf-pkg-pro
# Set the BDF temporary file storage directory
export BDF_TMPDIR=/tmp/$RANDOM

# Set the available heap memory to be unrestricted, which may be limited by system
↳administration if computing in a supercomputing environment
ulimit -s unlimited
# Set the available computation time to be unlimited, which may be limited by system
↳administration if computing in a supercomputing environment
ulimit -t unlimited

# Set the number of OpenMP parallel threads
export OMP_NUM_THREADS=4
# Set the OpenMP available heap memory size
export OMP_STACKSIZE=1024M

# Perform BDF calculations, note that the default output is printed to standard output
$BDFHOME/sbin/bdfdrv.py -r h2o.inp
```

The above is a Bash Shell script that defines some basic environment variables and executes the calculation using `$BDFHOME/sbin/bdfdrv.py`. The environment variables defined in the script are:

- `BDFHOME` variable specifies the directory where BDF is installed.

- `BDF_TMPDIR` variable specifies the BDF runtime temporary file storage directory.
- `ulimit -s unlimited` sets the available stack area memory for the program to be unlimited.
- `ulimit -t unlimited` sets the program execution time to be unlimited.
- `export OMP_NUM_THREADS=4` sets the number of OpenMP threads available for parallel computation.
- `export OMP_STACKSIZE=1024M` sets the available Stack area memory for OpenMP to be 1024 megabytes.

The command to perform the calculation is

```
$ ./run.sh h2o.inp &>h2o.out&
```

Since BDF prints the default output to the standard output, we use the Linux redirect command here to redirect the standard output to the file `h2o.out`.

### 4.1.3 Analysis of the calculation results

After the computation, the files `h2o.out`, `h2o.chkfil`, `h2o.scforb` will be obtained.

- `h2o.out` is a text file, user readable, storing the BDF output printing information.
- `h2o.chkfil` is a binary file, not user readable, used to pass data between different modules of the BDF; `h2o.chkfil` is a binary file, not user readable, used to pass data between different modules of the BDF.
- `h2o.scforb` is a text file, user-readable, storing information on molecular orbital factors, orbital energies, etc. for self-consistent iterations of `scf`, mainly used for restarting or as initial guess orbits for other `scf` calculations.

If the input file is in BDF simple input mode, `h2o.out` will first give some basic user setup information,

```
|===== BDF Control parameters =====|  
  
1: Input BDF Keywords  
  soc=None      scf=rhf      skeleton=True     xcfuntype=None  
  xcfun=None     direct=True     charge=0         hamilton=None  
  spinmulti=1  
  
2: Basis sets  
  ['3-21g']  
  
3: Wavefunction, Charges and spin multiplicity  
  charge=0      nuclearcharge=10  spinmulti=1  
  
5: Energy method  
  scf  
  
7: Acceleration method
```

(continues on next page)

(continued from previous page)

ERI

```
8: Potential energy surface method
   energy
```

```
|=====|
```

Here, the

- Input `BDF Keywords` gives some basic control parameters.
- `Basis set` gives the basis set used for the calculation.
- `Wavefunction`, `Charges` and `spinmulti` gives the system charges, total nuclear charges and spin multiplicity (2S+1).
- `Energy method` gives the energy calculation method.
- `Accleration method` gives the two-electron integral calculation acceleration method.
- `Potential energy surface method` gives the potential energy surface calculation method, here it is a single point energy calculation.

Subsequently, the system executes the **\*\*COMPASS\*\*** module, which gives the following prompt :

```
|*****|
```

```
Start running module compass
Current time 2021-11-18 11:26:28
```

```
|*****|
```

The Cartesian coordinates of the input molecular structure in **Bohr** are then printed, as well as details of the basis functions for each type of atom

```
|-----|
```

Atom	Cartcoord(Bohr)			Charge	Basis	Auxbas	Uatom	Nstab	Alink	Mass
O	0.000000	0.000000	0.000000	8.00	1	0	0	0	E	15.9949
H	1.889726	0.000000	0.000000	1.00	2	0	0	0	E	1.0073
H	-0.615235	1.786771	0.000000	1.00	2	0	0	0	E	1.0073

```
|-----|
```

```
End of reading atomic basis sets ..
Printing basis sets for checking ....
```

(continues on next page)

(continued from previous page)

```

Atomic label: O      8
Maximum L   1 6s3p ----> 3s2p NBF =    9
#--->s function
      Exp Coef          Norm Coef          Con Coef
          322.037000    0.192063E+03    0.059239    0.000000    0.000000
          48.430800    0.463827E+02    0.351500    0.000000    0.000000
          10.420600    0.146533E+02    0.707658    0.000000    0.000000
           7.402940    0.113388E+02    0.000000   -0.404454    0.000000
           1.576200    0.355405E+01    0.000000    1.221562    0.000000
           0.373684    0.120752E+01    0.000000    0.000000    1.000000
#--->p function
      Exp Coef          Norm Coef          Con Coef
          7.402940    0.356238E+02    0.244586    0.000000
          1.576200    0.515227E+01    0.853955    0.000000
          0.373684    0.852344E+00    0.000000    1.000000

Atomic label: H      1
Maximum L   0 3s ----> 2s NBF =    2
#--->s function
      Exp Coef          Norm Coef          Con Coef
          5.447178    0.900832E+01    0.156285    0.000000
          0.824547    0.218613E+01    0.904691    0.000000
          0.183192    0.707447E+00    0.000000    1.000000

```

Subsequently, the molecular symmetry is automatically determined and the rotation to the standard orientation mode is decided according to the user settings.

```

Auto decide molecular point group! Rotate coordinates into standard orientation!
Threshold= 0.10000E-08 0.10000E-11 0.10000E-03
geomsort being called!
gsym: C02V, noper=    4
Exiting zgeomsort....
Representation generated
Binary group is observed ...
Point group name C(2V)                                4
User set point group as C(2V)
Largest Abelian Subgroup C(2V)                        4
Representation generated
C|2|V|                2

Symmetry check OK
Molecule has been symmetrized
Number of symmery unique centers:                    2

```

(continues on next page)

(continued from previous page)

Atom	Cartcoord(Bohr)			Charge	Basis	Auxbas	Uatom	Nstab	Alink	Mass
O	0.000000	0.000000	0.000000	8.00	1	0	0	0	E	15.9949
H	1.889726	0.000000	0.000000	1.00	2	0	0	0	E	1.0073
H	-0.615235	1.786771	0.000000	1.00	2	0	0	0	E	1.0073

Atom	Cartcoord(Bohr)			Charge	Basis	Auxbas	Uatom	Nstab	Alink	Mass
O	0.000000	-0.000000	0.219474	8.00	1	0	0	0	E	15.9949
H	-1.538455	0.000000	-0.877896	1.00	2	0	0	0	E	1.0073
H	1.538455	-0.000000	-0.877896	1.00	2	0	0	0	E	1.0073

Careful users may have noticed that the coordinates of the water molecules here are different from the ones entered. Finally, **COMPASS** generates symmetry adapted orbital and gives the integrable representations to which the dipole and quadrupole moments belong, printing a multiplication table for the  $C(2v)$  point group, giving the total number of basis functions and the number of symmetry adapted orbital for each integrable representation.

```

Number of irreps:      4
IRREP:      3   4   1
DIMEN:      1   1   1

Irreps of multipole moment operators ...
Operator  Component   Irrep    Row
Dipole    x             B1       1
Dipole    y             B2       1
Dipole    z             A1       1
Quadpole  xx            A1       1
Quadpole  xy            A2       1
Quadpole  yy            A1       1
Quadpole  xz            B1       1
Quadpole  yz            B2       1
Quadpole  zz            A1       1

Generate symmetry adapted orbital ...
Print Multab
  1  2  3  4
  2  1  4  3
  3  4  1  2
  4  3  2  1

```

(continues on next page)

(continued from previous page)

```

-----|
Symmetry adapted orbital

Total number of basis functions:      13      13

Number of irreps:      4
Irrep :   A1      A2      B1      B2
Norb  :     7      0      4      2
-----|

```

Here, the  $C(2v)$  point group has 4 one-dimensional integrable representations, labeled A1, A2, B1, B2, with 7, 0, 4, 2 symmetrically matched orbitals, respectively.

**Attention:** Different quantum chemistry software may use different molecular standard orientations, resulting in some molecular orbitals being labeled with different integrable representations in different programs.

Finally, the COMPASS calculation ends normally, giving the following output.

```

|*****|

Total cpu      time:      0.00 S
Total system  time:      0.00 S
Total wall    time:      0.02 S

Current time   2021-11-18 11:26:28
End running module compass
|*****|

```

**Note:** For each module execution of BDF, there will be informaton about the start of the execution and the time printed after the end of the execution, so that it is convenient for the user to locate exactly which calculation module has made an error.

The second module executed in this example is **XUANYUAN**, which is mainly used to calculate single and double electron integrals. Here, the **XUANYUAN** module only calculates and stores single-electron integrals and special double-electron integrals that require pre-screening of the integrals. If not specified, the BDF defaults to the direct calculation of the double electron integral to construct the Fock matrix. If user write in `compass` module the key word `Saorb`, double electron integral will be calculated and stored. The output of the **XUANYUAN** module is relatively simple and does not require special attention. Here, we give the most critical output.

```

[aint_1e]
  Calculating one electron integrals ...
  S T and V integrals ....
  Dipole and Quadupole integrals ....
  Finish calculating one electron integrals ...

-----

Timing to calculate 1-electronic integrals

CPU TIME(S)      SYSTEM TIME(S)      WALL TIME(S)
      0.017            0.000            0.000
-----

Finish calculating 1e integral ...
Direct SCF required. Skip 2e integral!
Set significant shell pairs!

Number of significant pairs:      7
Timing calculate K2 integrals.
CPU:      0.00 SYS:      0.00 WALL:      0.00

```

From the output we see that the single-electron overlap, kinetic and nuclear attraction integrals are computed, and also the dipole and quadrupole moment integrals are computed. The two-electron integral calculation is ignored because the input requires the default integration to be calculated directly by SCF (Direct SCF).

Finally, the BDF invokes the **SCF** module to perform the **RHF** self-consistent field calculation. Information of interest are:

```

Wave function information ...
Total Nuclear charge   :      10
Total electrons        :      10
ECP-core electrons     :       0
Spin multiplicity(2S+1) :       1
Num. of alpha electrons :       5
Num. of beta electrons :       5

```

The nuclear charge number, the total electron number, the core electron number for the pseudopotential calculation, the spin multiplicity, and the alpha and beta electron numbers are given here, and the user should check that the electronic states are correct. Then, the `scf` module first calculates the atoms and generates the initial guess density matrix for the molecular calculations.

```

[ATOM SCF control]
heff=                0
After initial atom grid ...

```

(continues on next page)

(continued from previous page)

```

Finish atom   1  O                -73.8654283850
After initial atom grid ...
Finish atom   2  H                -0.4961986360

Superposition of atomic densities as initial guess.

```

checking for possible linear correlations in the treatment of the basis functions.

```

Check basis set linear dependence! Tolerance = 0.100000E-04

```

It then proceeds to the SCF iterations, where after 8 iterations of convergence the accelerated convergence methods such as **DIIS** and **Level shift** are turned off and the energies are recalculated.

Iter.	idiis	vshift	SCF Energy	DeltaE	RMSDeltaD	MaxDeltaD	Damping	Times (S)
1	0	0.000	-75.465225043	-0.607399386	0.039410497	0.238219747	0.0000	0.00
2	1	0.000	-75.535887715	-0.070662672	0.013896819	0.080831047	0.0000	0.00
3	2	0.000	-75.574187153	-0.038299437	0.004423591	0.029016074	0.0000	0.00
4	3	0.000	-75.583580885	-0.009393732	0.000961664	0.003782740	0.0000	0.00
5	4	0.000	-75.583826898	-0.000246012	0.000146525	0.000871203	0.0000	0.00
6	5	0.000	-75.583831666	-0.000004768	0.000012300	0.000073584	0.0000	0.00
7	6	0.000	-75.583831694	-0.000000027	0.000001242	0.000007487	0.0000	0.00
8	7	0.000	-75.583831694	-0.000000000	0.000000465	0.000002549	0.0000	0.00
diis/vshift <b>is</b> closed at iter = 8								
9	0	0.000	-75.583831694	-0.000000000	0.000000046	0.000000221	0.0000	0.00

Label	CPU Time	SYS Time	Wall Time
SCF iteration time:	0.017 S	0.017 S	0.000 S

Finally, the energy contributions and the Viry ratios of the different terms are printed.

```

Final scf result
E_tot = -75.58383169
E_ele = -84.37566837
E_nn = 8.79183668
E_1e = -121.94337426
E_ne = -197.24569473
E_kin = 75.30232047
E_ee = 37.56770589
E_xc = 0.00000000
Virial Theorem 2.003738

```

According to the Virial Theorem, the absolute value of the total potential energy of the system is two times the kinetic energy of the electron for a non-relativistic system, where the Virial ratio is 2.003738. The energy of the system is:

- $E_{tot}$  is the total energy of the system, i.e.,  $E_{ele} + E_{nn}$ ;

- $E_{\text{ele}}$  is the electron energy, i.e.  $E_{1e} + E_{ee} + E_{xc}$  ;
- $E_{\text{nn}}$  is the nuclear repulsion energy;
- $E_{1e}$  is the single electron energy, i.e.  $E_{ne} + E_{kin}$  ;
- $E_{ne}$  is the energy of attraction of the nucleus to the electron;
- $E_{kin}$  is the electron kinetic energy;
- $E_{ee}$  is the two-electron energy, including Coulomb repulsion and exchange energy.
- $E_{xc}$  is the exchange-related energy, which is not 0 for DFT calculation.

The output of the energy printout is the occupancy of the orbitals, the orbital energy, the HUMO-LOMO energy and the energy gap, as shown below.

```
[Final occupation pattern: ]

Irreps:      A1      A2      B1      B2

detailed occupation for iden/irep:      1  1
  1.00 1.00 1.00 0.00 0.00 0.00 0.00
detailed occupation for iden/irep:      1  3
  1.00 0.00 0.00 0.00
detailed occupation for iden/irep:      1  4
  1.00 0.00
Alpha      3.00      0.00      1.00      1.00

[Orbital energies:]

Energy of occ-orbs:  A1      3
  -20.43281195      -1.30394125      -0.52260024
Energy of vir-orbs:  A1      4
  0.24980046      1.23122290      1.86913815      3.08082943

Energy of occ-orbs:  B1      1
  -0.66958992
Energy of vir-orbs:  B1      3
  0.34934415      1.19716413      2.03295437

Energy of occ-orbs:  B2      1
  -0.47503768
Energy of vir-orbs:  B2      1
  1.78424252

Alpha  HOMO energy:      -0.47503768 au      -12.92643838 eV  Irrep: B2
```

(continues on next page)

(continued from previous page)

```
Alpha LUMO energy:      0.24980046 au      6.79741929 eV Irrep: A1
HOMO-LUMO gap:        0.72483814 au      19.72385767 eV
```

Here

- [Final occupation pattern:] gives the orbital occupation. Since we are performing a restricted Hartree-Fock calculation, the occupation is given only for the Alpha orbit, which is given separately according to the integrable representation. From this example, it can be seen that the first 3 of the A1 orbitals and the 1st of the B1 and B2 orbitals are occupied by 1 electron each. Since this example is an RHF, the alpha and beta orbitals are the same, so A1 indicates 3 double-occupied orbitals, and B1 and B2 indicate 1 double-occupied orbital each.
- [Orbital energies:] The orbital energies are given separately according to the integrable representation.
- Alpha HOMO energy: gives the HOMO orbital energy in units au and eV; the integrable representation to which the orbital belongs, in this case B2.
- Alpha LUMO energy: the LUMO orbital energy is given in units of au and eV; the integrable representation to which the orbital belongs, in this case A1.
- HOMO-LUMO gap: gives the energy difference between the HOMO and LUMO orbitals.

In order to reduce the number of output lines, BDF does not print the orbital composition and molecular orbital coefficients by default, but only gives the partial orbital occupation and orbital energy information according to the integrable representation. Only partial orbital occupancies and orbital energy information are given according to the integrable representation categories, as follows.

```
Symmetry  1 A1

Orbital      1      2      3      4      5      6
Energy     -20.43281  -1.30394  -0.52260  0.24980  1.23122  1.86914
Occ No.      2.00000  2.00000  2.00000  0.00000  0.00000  0.00000

Symmetry  2 A2

Symmetry  3 B1

Orbital      8      9     10     11
Energy     -0.66959  0.34934  1.19716  2.03295
Occ No.      2.00000  0.00000  0.00000  0.00000

Symmetry  4 B2

Orbital     12     13
```

(continues on next page)

(continued from previous page)

Energy	-0.47504	1.78424
Occ No.	2.00000	0.00000

The **SCF** module finally prints the results of Mulliken and Lowdin Bourdin analysis, with information on the dipole moments of the molecules.

```
[Mulliken Population Analysis]
Atomic charges:
  1O      -0.7232
  2H       0.3616
  3H       0.3616
  Sum:    -0.0000

[Lowdin Population Analysis]
Atomic charges:
  1O      -0.4756
  2H       0.2378
  3H       0.2378
  Sum:    -0.0000

[Dipole moment: Debye]
      X           Y           Z
Elec:-.1081E-64  0.4718E-32  -.2368E+01
Nucl:0.0000E+00  0.0000E+00  0.5644E-15
Totl: -0.0000      0.0000     -2.3684
```

**Hint:**

1. add the `iprtmo` keyword to the input of the **SCF** module with a value of 2 to output detailed information about the molecular orbitals.
2. add the `moldden` keyword to the input of the **SCF** module to output the molecular orbitals and occupancies as a moldden format file, which can be used by third-party programs for visualization or wave function analysis (such as [GabEdit](#), [JMol](#), [Molden](#), [Multiwfn](#)), to calculate wavefunction analysis , or calculate single electron property .

## 4.2 Gaussian basis set

In order to solve the Hartree-Fock, Kohn-Sham DFT equations, it is necessary to expand the molecular orbitals into linear combinations of single-electron basis functions.

$$\varphi_i(r) = C_{1,i}\chi_1(r) + C_{2,i}\chi_2(r) + C_{3,i}\chi_3(r) + \cdots + C_{N,i}\chi_N(r)$$

In quantum chemistry calculations, the basis functions have only mathematical meaning, not physical meaning. The more basis functions there are, the more accurate the result will be, but it also depends on how well the basis functions are set up. The Complete Basis Set Limit (CBS) is reached when there are infinitely many basis functions, which is called a complete set, and the molecular orbitals can be perfectly expanded. The actual use of a finite basis set does not reach the CBS, and the resulting error in the calculation result is called the basis set incompleteness error.

As many basis functions as are used, as many molecular orbitals are produced, but only occupied orbitals, and lower order non-occupied orbitals (valence level empty orbitals) are usually chemically meaningful. If the basis functions are taken to be atomic orbitals, it is called linear combination of atomic orbitals (LCAO), but this is only a concept in structural chemistry, and the basis functions used in actual calculations are not the real atomic orbitals.

The commonly used basis functions in quantum chemistry are as follows.

1. Gauss type orbital (GTO) basis functions: Most quantum chemistry programs use GTO basis functions because they are mathematically easy to calculate two-electron integrals.
2. Slater orbital (Slater type orbital, STO) basis functions: Semi-empirical and used by a few quantum chemistry programs (e.g. ADF). It is difficult to calculate two-electron integrals, but its radial behavior is closer to the actual atomic orbitals than the GTO basis functions, so that only a small number of STOs are needed to achieve a large number of GTO results.
3. Plane wave: A basis function specifically suitable for periodic calculations and much less cost effective than the GTO basis function for isolated systems.
4. Numerical atomic orbital (NAO) basis functions: Few programs support them, typically Dmol3, Siesta. NAO basis functions do not have an analytic mathematical form, but are described by discrete distributions of points.

The STO basis function was used in the early days of BDF software, and the GTO basis function is mainly used now.

For GTO basis functions with orbital angular momentum  $L$  higher than  $p^*$  (e.g., GTO basis functions such as  $*d$ ,  $f$ , etc.), there are two ways to represent them. One is written in the form of a Cartesian function (also called a right-angle function).

$$N x^{l_x} y^{l_y} z^{l_z} \exp(-\alpha r^2), \quad L = l_x + l_y + l_z$$

It has  $(L+1)(L+2)/2$  components, e.g., the  $d$  function contains  $xx$ ,  $yy$ ,  $zz$ ,  $xy$ ,  $xz$ ,  $yz$ . The other is written in the form of a spherical function (also called a spherical harmonic function, pure function).

$$N Y_m^L r^L \exp(-\alpha r^2)$$

It has  $2L+1$  components, for example, the  $d$  function contains  $-2$ ,  $-1$ ,  $0$ ,  $+1$ ,  $+2$ .

The advantage of the Cartesian function is that it is easy to calculate the integral, but there are redundant functions; whereas the spherical function corresponds to exactly  $(L + 1)(L + 2)/2$  magnetic quantum numbers, so in quantum chemistry programs the integral is usually calculated first under the Cartesian function and then combined into the integral of the spherical function by a certain linear relation [53].

**Attention:**

1. most modern Gaussian basis sets are optimized under spherical basis functions, except for the older basis sets such as Pople type.
2. Cartesian basis functions have no advantage in terms of accuracy or efficiency, especially for all-electron relativity calculations, which also lead to numerical instability, so spherical basis functions are always used in BDF calculations.
3. Cartesian and spherical basis functions lead to different results. If the results of BDF calculations are repeated with other quantum chemistry programs, it is necessary to check whether the spherical basis functions are used, in addition to ensuring that the structure, method, and basis set are the same.

In the literature, data sets of optimized GTO basis functions for various atoms in different situations have been created and given different names to be called by quantum chemistry programs. These named GTO basis function data sets are called **Gaussian Basis Sets**. The Gaussian Basis Sets built into the BDF are mainly from the following Basis Set Repository websites, and the original literature on the various Basis Sets can be found at the corresponding websites.

- Basis Set Exchange [54] : All-electron basis sets, scalar ECP basis sets, can be exported in BDF format (note: ECP basissets have to be manually repositioned for ECP data). <https://www.basissetexchange.org/>
- Stuttgart/Cologne pseudopotential basis set library: mainly SOECP basis sets, and a few early scalar ECP basis sets. <http://www.tc.uni-koeln.de/PP/clickpse.en.html>
- Turbomole basis set library: all-electron basis set, scalar ECP basis set, SOECP basis set. <http://www.cosmologic-services.de/basis-sets/basissets.php>
- Dyall Relativistic Basis Set: All-electron relativistic basis set. <http://dirac.chem.sdu.dk/basisarchives/dyall/index.html>
- Sapporo basis set library: all-electron basis sets. <http://sapporo.center.ims.ac.jp/sapporo/>
- Clarkson University ECP basis set library: SOECP basis set. <https://people.clarkson.edu/~pchristi/rep.html>
- ccECP Basis Set Library: Scalar ECP Basis Sets. <https://pseudopotentiallibrary.org/>

In addition, there are individual elements with built-in motifs from the original literature.

- All-electron basis set Dirac-RPF-4Z and Dirac-aug-RPF-4Z, including s-, p-region elements [55], d-region elements [56], f-region elements [57]
- Pseudopotential basis set Pitzer-AVDZ-PP, Pitzer-VDZ-PP, Pitzer-VTZ-PP [58]

- Ce - Lu [59], Fr - Pu [60], Am - Og [61, 62] in the pseudopotential basis set CRENB� (Note: the Am - Og basis set on the Basis Set Exchange is wrong!)
- Am - Og [61, 62] in the pseudopotential basis set CRENBŜ (Note: the Am - Og basis set on Basis Set Exchange is wrong!)
- Ac, Th, Pa [63] , U [64] in the pseudopotential basis set Stuttgart-ECPMDFSO-QZVP

BDF users can use either the standard basis sets from the BDF basis set library or custom basis sets.

### 4.2.1 All-electron basis sets

All-electron basis sets are divided into two categories: non-shrinking basis sets and shrinking basis sets. The former can be used for both non-relativistic and relativistic calculations, but mainly for relativistic calculations, while the latter is divided into non-relativistic shrinkage basis sets and relativistic shrinkage basis sets.

All-electron relativistic calculations use Hamiltonians such as DKH, ZORA, X2C, etc. that take relativistic effects into account (see Relativistic effects ) , when it is necessary to use shrinkage basis sets optimized specifically for relativistic calculations, such as the cc-pVnZ-DK series, SARC, ANO-RCC, etc. Most relativistic shrinkage basis sets treat the nucleus as a point charge, but some do take into account the nucleus distribution size effect when doing the shrinkage, which has the most pronounced effect on the shrinkage factor of the *s* and *p* asis functions. Accordingly, a finite nucleus model must also be used in the calculation of molecular integrals. finite nucleus model .

### 4.2.2 Pseudopotential basis sets

The Effective Core Potential (ECP) includes the Pseudopotential (PP) and the Model Core Potential (MCP). The PP in quantum chemical calculations is not fundamentally different from the PP in plane wave calculations, except that it is expressed in a concise analytic form. Most quantum chemistry software, including BDF, supports PP, but fewer quantum chemistry software support MCP, so the names ECP and PP can be used interchangeably without ambiguity.

The pseudopotential basis set needs to be used in conjunction with the pseudopotential, and the basis functions describe only the valence level electrons of the atoms. When heavier elements are involved in the system, the pseudopotential basis set is usually used for them, while the normal basis set is used for the other atoms as usual. The Lan series, the Stuttgart series, and the cc-pVnZ-PP series all belong to this set. For ease of recall, the pseudopotential basis sets of some lighter elements are actually non-relativistic all-electron basis sets, such as the Def2 series of basis sets for elements before the fifth period.

The pseudopotential basis sets are divided into scalar pseudopotential basis sets and spin-orbit coupled pseudopotential (SOECP) basis sets, depending on whether the pseudopotential contains a spin-orbit coupling term or not.

### 4.2.3 Custom basis set files

The BDF can use non-built-in basis sets, where the basis set data is saved in a text format basis set file, placed in the calculation directory, with the file name is the name of the base set to be referenced in the BDF.

**Warning:** The file name of the custom base set file must be in **all capital letters** ! However, when referenced in the input file, the case is arbitrary.

For example, create a text file MYBAS-1 in the calculation directory (note: if you create a text file under Windows OS, the system may hide the extension *.txt*, so the actual name is MYBAS-1.txt) with the following contents

```
# This is my basis set No. 1.                # any blank lines and # leading comment
↳lines
# Supported elements: He and Al

****                                       # a line beginning with four asterisks,
↳followed by a base set of elements
He      2      1                          # element sign, nuclear charge number,
↳highest angular momentum of basis function
S       4       2                          # S type GTO basis function, 4 original
↳functions reduced to 2
                3.836000E+01              # exponents of four S-type Gaussian
↳primitive functions
                5.770000E+00
                1.240000E+00
                2.976000E-01
                2.380900E-02      0.000000E+00 # Two colums of contraction factors,
↳corresponding to two contraction S-type GTO basis functions
                1.548910E-01      0.000000E+00
                4.699870E-01      0.000000E+00
                5.130270E-01      1.000000E+00
P       2       2                          # P type GTO basis function, two original
↳functions are reduced to two
                1.275000E+00
                4.000000E-01
                1.0000000E+00      0.0000000E+00
                0.0000000E+00      1.0000000E+00
****                                       # four asterisks end the base set of he, followed by the
↳base set of another element, or end
Al      13      2
(ellipsis)
```

In the above basis set, the P function is not contracted and can also be written in the following form.

```

(S function, ellipsis)
P      2      0          # 0 indicates non shrinkage, and the shrinkage factor is
↳not required at this time
          1.275000E+00
          4.000000E-01
****
(ellipsis)

```

For pseudopotential basis sets, it is also necessary to provide ECP data after the valence basis function. For example

```

****                                     # for the price basis function, the
↳note is the same as above
Al     13     2
S       4     3
          14.68000000
          0.86780000
          0.19280000
          0.06716000
-0.0022368000    0.0000000000    0.0000000000
-0.2615913000    0.0000000000    0.0000000000
 0.6106597000    0.0000000000    1.0000000000
 0.5651997000    1.0000000000    0.0000000000
P       4     2
          6.00100000
          1.99200000
          0.19480000
          0.05655000
-0.0034030000    0.0000000000
-0.0192089000    0.0000000000
 0.4925534000   -0.2130858000
 0.6144261000    1.0000000000
D       1     1
          0.19330000
          1.0000000000
ECP                                     # ECP data section
Al     10     2     2     # element symbol, number of core electrons, ECP maximum
↳angular momentum, soecp maximum angular momentum (optional)
D potential 4                                     # ECP maximum angular momentum (D
↳function)
  2      1.22110000000000    -0.53798100000000 # R power, exponent, factor (the same
↳below)
  2      3.36810000000000    -5.45975600000000
  2      9.75000000000000    -16.65534300000000
  1     29.26930000000000    -6.47521500000000

```

(continues on next page)

(continued from previous page)

```

S potential  5                               # S number of items projected
  2      1.5631000000000000      -56.20521300000000
  2      1.7712000000000000      149.68995500000000
  2      2.0623000000000000      -91.45439399999999
  1      3.3583000000000000       3.72894900000000
  0      2.1300000000000000       3.03799400000000

P potential  5                               # P number of items projected
  2      1.8231000000000000       93.67560600000000
  2      2.1249000000000000     -189.88896800000001
  2      2.5705000000000000      110.24810400000000
  1      1.7575000000000000       4.19959600000000
  0      6.7693000000000000       5.00335600000000

P so-potential  5                           # the number of items projected by
↳P so. Scalar ECP does not have this part
  2      1.8231000000000000       1.51243200000000 # Scalar ECP does not have this part
  2      2.1249000000000000     -2.94701800000000 # Scalar ECP does not have this part
  2      2.5705000000000000       1.64525200000000 # Scalar ECP does not have this part
  1      1.7575000000000000     -0.08862800000000 # Scalar ECP does not have this part
  0      6.7693000000000000       0.00681600000000 # Scalar ECP does not have this part

D so-potential  4                           # the number of items of D so
↳projection. Scalar ECP does not have this part
  2      1.2211000000000000     -0.00138900000000 # Scalar ECP does not have this part
  2      3.3681000000000000       0.00213300000000 # Scalar ECP does not have this part
  2      9.7500000000000000       0.00397700000000 # Scalar ECP does not have this part
  1      29.2693000000000000      0.03253000000000 # Scalar ECP does not have this part
****

```

For scalar ECP, the SOECP highest angular momentum is 0 (which can be omitted and not written), and it is not necessary to provide the data for the SO projection part.

Once the above data is saved, the MYBAS-1 base set can be called in the BDF input file, which is achieved by the following hybrid input mode.

```

#!/bdfbasis.sh
HF/genbas

Geometry
.....
End geometry

$Compass
Basis
  mybas-1      # give the name of the base set file in the current directory. It
↳is not case sensitive here

```

(continues on next page)

```
$End
```

The custom base set must be entered in BDF's mixed mode. In the second line the input base set is set to **genbas**, and the custom base set file name needs to use the keyword `Basis` in the **COMPASS** module with a value of `mybas-1`, which means that the base set file named `MYBAS-1` is called.

## 4.2.4 Basis set designation

### Use the same BDF built-in basis set for all atoms

In simple input mode, the basis set is specified in `method/generic/basis set` or `method/basis set`. Here, the `basis sets` are the BDF built-in ones listed in the previous sections base set names, and the input characters are case-insensitive, as follows.

```
#!/basisexample.sh
TDDFT/PBE0/3-21g

Geometry
H 0.000 0.000 0.000
Cl 0.000 0.000 1.400
End geometry
```

```
#!/basisexample.sh
HF/lanl2dz

Geometry
H 0.000 0.000 0.000
Cl 0.000 0.000 1.400
End geometry
```

In case of advanced input mode, the basis set used for the calculation is specified in the `compass` module using the keyword `basis`, for example

```
$compass
Basis
lanl2dz
Geometry
H 0.000 0.000 0.000
Cl 0.000 0.000 1.400
End geometry
$end
```

where `lanl2dz` calls the built-in `LanL2DZ` basis set (registered in the `basisname` `basisname` file), which is case-insensitive.

### Specifying different basis sets for different elements

You have to use the mixed input mode, i.e. set the basis set to `genbas` in `Methods/Generic/Bases`, and add the **COMPASS** module input, specifying the basis set using the `basis-block ...end basis` keyword.

If you specify a different name for a different element, you need to put it in the **COMPASS** module's `basis-block ...end basis` block. where the first line is the default base set and the subsequent lines specify other base sets for different elements in the format `element= base set name * or *element1, element2, ..., element n= base set name`.

For example, an example of using different basis sets for different atoms in mixed input mode is as follows.

```
#!/ multibasis.sh
HF/genbas

Geometry
H  0.000  0.000  0.000
Cl 0.000  0.000  1.400
End geometry

$compass
Basis-block
lanl2dz
H = 3-21g
End Basis
$end
```

In the above example, the 3-21G basis set is used for H, while the default LanL2DZ basis set is used for Cl which is not additionally defined.

In case of advanced input, the following is used.

```
$compass
Basis-block
lanl2dz
H = 3-21g
End Basis
Geometry
H  0.000  0.000  0.000
Cl 0.000  0.000  1.400
End geometry
$end
```

### Assigning different basis sets to different atoms of the same element

The BDF can also assign different base sets with different names to different atoms of the same element, which need to be distinguished by an arbitrary number after the element symbol to distinguish them. For example

```

#! CH4.sh
RKS/B3lyp/genbas

Geometry
  C      0.000   -0.000    0.000
  H1     -0.000   -1.009   -0.357
  H2     -0.874    0.504   -0.457
  H1      0.874    0.504   -0.357
  H2      0.000    0.000    1.200

End geometry

$compass
Basis-block
  6-31g
  H1= cc-pvdz
  H2= 3-21g
End basis
$end

```

In the above example, the cc-pVDZ basis set is used for the two hydrogen atoms of type H1, the 3-21G basis set for the two hydrogen atoms of type H2, and the 6-31G basis set for the carbon atoms. Note that the symmetry equivalent atoms must use the same basis set, which will be checked by the program; if the symmetry equivalent atoms have to use different basis sets, the symmetry can be set to a lower point set symmetry by `Group` or turned off with `Nosymm`.

## 4.2.5 Auxiliary basis sets

The method using density fitting approximation (RI) requires an auxiliary basis set. the Ahlrichs family of basis sets and the Dunning correlation consistency basis set as well as other individual basis sets have specially optimized auxiliary basis sets. the auxiliary basis sets can be specified in BDF by the `RI-J`, `RI-K` and `RI-C` keywords in compass. `RI-J` is used to assign coulomb fitting basis set, `RI-K` is used to assign coulomb exchange fitting basis set, `RI-C` assign coulomb correlation fitting basis set. The auxiliary basis sets supported by BDF are stored in the corresponding folder under the `$BDFHOME/basis_library` path.

High-level density fitting bases can be used on lower-level bases, e.g. `cc-pVTZ/C` can be used to do RI-J on `cc-pVTZ`, and for pople series bases such as `6-31G**` that do not have a standard auxiliary base, `cc-pVTZ/J` can be used to do RI-J or RIJCOSX. Conversely, a high-level orbital basis set combined with a low-level auxiliary basis set introduces a more significant error.

```

$Compass
Basis
  DEF2-SVP
RI-J
  DEF2-SVP

```

(continues on next page)

(continued from previous page)

Geometry			
C	1.08411	-0.01146	0.05286
H	2.17631	-0.01146	0.05286
H	0.72005	-0.93609	0.50609
H	0.72004	0.05834	-0.97451
H	0.72004	0.84336	0.62699
End Geometry			
\$End			

In the above example, the def2-SVP basis set was used to calculate the CH<sub>4</sub> methane molecule, while the def2-SVP standard Coulomb fitting basis set was used for accelerated calculations.

**Hint:** The RI calculation function of BDF is used to accelerate wave function calculation methods such as MCSCF, MP2 etc. It is not recommended for users in SCF, TDDFT, etc. The MPEC method does not depend on redundant functions and is comparable to the RI method in terms of computational speed and accuracy. The MPEC method does not depend on the redundancy function and is comparable to the RI method in terms of speed and accuracy.

### 4.3 Exchange-dependent generalized functions supported by BDF

The density flooding theory (DFT) of the BDF supports restricted, unrestricted and restricted open-shell Kohn-Sham calculations, referred to as RKS, UKS and ROKS, respectively. The inputs are close to those of RHF, UHF and ROHF. The key is to specify the exchange-related generic functions. bdf supports a variety of generic functions such as LDA, GGA, Meta-GGA, Hybrid, RS Hybrid and Hybrid Meta-GGA.

**Attention:**

1. VWN5 is used for the LDA correlation term of B3LYP, while VWN3 is used for the LDA correlation term of GB3LYP corresponding to B3LYP in the Gaussian program 2.
2. For range-separated generalized calculations, the `rs` values must be set manually in the `Xuanyuan` module (see the list of keywords in the `xuanyuan` module). `wB97`, `wB97X`, `CAM-B3LYP` and `LC-BLYP` have `rs` values of 0.40, 0.30, 0.33 and 0.33, respectively.
3. For the two-hybrid generalized function calculation, an `MP2` module must be added after the `SCF` module (see example test116 in the *example description*) and the final result is read from the output of the `MP2` module.
4. User-defined generalized functions can be implemented in the `SCF` module by adjusting the proportion of HF exchange terms and the proportion of MP2 related terms in the generalized function using the `facex` and `facco` keywords (see the list of keywords in the `SCF` module).

5. BDF uses `libxc`, in principle supports all the general functions included in `libxc`, but it takes time to refine and add. Users can give us feedback on the required generic functions so that we can add them as needed.

Note that while all general functions support (without dispersion correction) single-point energy calculations for the ground state, some functions are only partially supported by the general function are supported. The following is a list of the general functions supported for various computational tasks.

## 4.4 Self-consistent field methods: Hartree-Fock and Kohn-Sham

The self-consistent fields of the BDF include the Hartree-Fock and Kohn-Sham methods.

### 4.4.1 Restricted Hartree-Fock Method

An example of the Restricted Hartree-Fock method (RHF) was mentioned in the first example section and will not be repeated here.

### 4.4.2 Unrestricted Hartree-Fock method

For systems with unpaired electrons, the UHF method is required, and the restricted open-shell Hartree-Fock (RHF) method can also be used, see later. For odd-electron systems, the BDF defaults to a spin multiplet of 2 and uses the UHF calculation. For example, to calculate the  $C_3H_5$  molecule,

```
#!bdf.sh
UHF/3-21G

geometry
C          0.00000000    0.00000000    0.00000000
C          0.00000000    0.00000000    1.45400000
C          1.43191047    0.00000000    1.20151555
H          0.73667537   -0.61814403   -0.54629970
H         -0.90366611    0.32890757   -0.54629970
H          2.02151364    0.91459433    1.39930664
H          2.02151364   -0.91459433    1.39930664
H         -0.79835551    0.09653770    2.15071009
end geometry
```

The output of the UHF calculation is similar to that of the RHF, in that the output from the `scf` module can be checked for correct charge and spin multiplicity.

```
Wave function information ...
Total Nuclear charge   :      23
Total electrons       :      23
Ecp-core electrons    :        0
Spin multiplicity(2S+1) :        2
Num. of alpha electrons :      12
Num. of beta  electrons :      11
```

The orbital occupancy is given separately for Alpha and Beta orbitals.

```
[Final occupation pattern: ]

Irreps:          A

detailed occupation for iden/irep:      1  1
  1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
  1.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
Alpha           12.00

detailed occupation for iden/irep:      2  1
  1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
  1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
  0.00 0.00 0.00 0.00 0.00 0.00 0.00
Beta            11.00
```

The orbital energy, HOMO-LUMO gap is also printed separately according to Alpha and Beta orbits.

```
[Orbital energies:]

Energy of occ-orbsA:   A           12
-11.18817955   -11.18789391   -11.17752809   -1.11801069   -0.85914580
-0.78861789    -0.65514687    -0.61300160    -0.55514631   -0.49906127
-0.37655522    -0.30477047

Energy of vir-orbsA:   A           25
 0.18221017    0.28830234     0.31069644     0.32818004     0.35397043
 0.38822931    0.42917813     0.49394022     0.93909970     0.94842069
 0.96877856    0.97277131     1.02563249     1.05399606     1.11320732
 1.17687697    1.26547430     1.31245896     1.32719078     1.34493766
 1.37905664    1.45903968     1.80285556     1.93877012     2.01720415
```

(continues on next page)

(continued from previous page)

Energy of occ-orbsB:     A                    11				
-11.19670896	-11.16769083	-11.16660825	-1.07470168	-0.84162305
-0.74622771	-0.63695581	-0.58068095	-0.53876236	-0.46400924
-0.37745766				
Energy of vir-orbsB:     A                    26				
0.15755278	0.18938428	0.30608423	0.33204779	0.33996597
0.38195612	0.39002159	0.43644421	0.52237314	0.94876233
0.96144960	0.97568581	1.01804430	1.05467405	1.09547593
1.13390456	1.19865205	1.28139866	1.32654541	1.33938005
1.34914150	1.38200544	1.47565481	1.79509704	1.96917149
2.03513467				
Alpha	HOMO energy:	-0.30477047 au	-8.29322996 eV	Irrep: A
Alpha	LUMO energy:	0.18221017 au	4.95819299 eV	Irrep: A
Beta	HOMO energy:	-0.37745766 au	-10.27114977 eV	Irrep: A
Beta	LUMO energy:	0.15755278 au	4.28723115 eV	Irrep: A
HOMO-LUMO gap:		0.46232325 au	12.58046111 eV	

Other output information can be found in the example of RHF calculation and will not be described in detail here.

#### 4.4.3 Restricted open-shell Hartree-Fock method

Restricted open-shell Hartree-Fock (Restricted open-shell Hartree-Fock - ROHF) can also be calculated for the open-shell molecular system. An example of ROHF calculation for the CH<sub>2</sub> triplet state is given here.

```
#!/bdf.sh
rohf/cc-pvdz spinmulti=3

geometry # 输入坐标单位 Angstrom
C      0.000000      0.00000      0.31399
H      0.000000     -1.65723     -0.94197
H      0.000000      1.65723     -0.94197
end geometry
```

Here, the ROHF is specified in the second line and the triplet state is calculated using the keyword `spinmulti=3`. The output of ROHF is similar to that of UHF, but its Alpha orbitals are the same as Beta so the corresponding Alpha 和 Beta orbitals are equal in energy, as follows.

```
[Orbital energies:]

Energy of occ-orbsA:     A1                    3
-11.42199273     -0.75328533     -0.22649749
Energy of vir-orbsA:     A1                    8
```

(continues on next page)

(continued from previous page)

0.05571960	0.61748052	0.70770696	0.83653819	1.29429307
1.34522491	1.56472153	1.87720054		
Energy of vir-orbsA:	A2	2		
1.34320056	1.53663810			
Energy of occ-orbsA:	B1	1		
-0.37032603				
Energy of vir-orbsA:	B1	6		
0.06082087	0.66761691	0.77091474	1.23122892	1.51131609
1.91351353				
Energy of occ-orbsA:	B2	1		
-0.16343739				
Energy of vir-orbsA:	B2	3		
0.65138659	1.35768658	1.54657952		
Energy of occ-orbsB:	A1	2		
-11.42199273	-0.75328533			
Energy of vir-orbsB:	A1	9		
-0.22649749	0.05571960	0.61748052	0.70770696	0.83653819
1.29429307	1.34522491	1.56472153	1.87720054	
Energy of vir-orbsB:	A2	2		
1.34320056	1.53663810			
Energy of occ-orbsB:	B1	1		
-0.37032603				
Energy of vir-orbsB:	B1	6		
0.06082087	0.66761691	0.77091474	1.23122892	1.51131609
1.91351353				
Energy of vir-orbsB:	B2	4		
-0.16343739	0.65138659	1.35768658	1.54657952	

Due to the different occupation numbers of Alpha and Beta orbitals, the HOMO, LUMO orbitals, and orbital energies of Alpha differ from those of Beta, as follows. .. code-block:

Alpha	HOMO energy:	-0.16343739 au	-4.44735961 eV	Irrep: B2
Alpha	LUMO energy:	0.05571960 au	1.51620803 eV	Irrep: A1
Beta	HOMO energy:	-0.37032603 au	-10.07708826 eV	Irrep: B1
Beta	LUMO energy:	-0.22649749 au	-6.16331290 eV	Irrep: A1
HOMO-LUMO gap:		-0.06306010 au	-1.71595329 eV	

#### 4.4.4 RKS, UKS, and ROKS Calculations

For the Restricted Kohn-Sham (RKS) method, an example of the RKS calculation for an H<sub>2</sub>O molecule is given here in a concise input mode, using the B3lyp generalized function.

```
#!/bdf.sh
B3lyp/3-21G

geometry
O
H 1 R1
H 1 R1 2 109.

R1=1.0      # OH bond length, unit is Angstrom
end geometry
```

The corresponding advanced mode input is:

```
$compass
geometry # On default: bond length unit in angstrom
o
h 1 1.0
h 1 1.0 2 109.
end geometry
basis
  3-21g
$end

$xuanyuan
$end

$scf
rks # Restricted Kohn-Sham calculation
dft # ask for B3lyp functional, it is different with B3lyp implemented in Gaussian.
  b3lyp
$end
```

Here, the input requires the use of the B3lyp generic function. Compared to Hartree-Fock, the output has an additional contribution from the Exc term, as follows.

```
Final scf result
E_tot =          -75.93603354
E_ele =          -84.72787022
E_nn  =           8.79183668
E_1e  =         -122.04354727
```

(continues on next page)

(continued from previous page)

```

E_ne = -197.45852687
E_kin = 75.41497960
E_ee = 44.81744844
E_xc = -7.50177140
Virial Theorem 2.006909

```

The ROKS calculation for the  $\text{H}_2\text{O}^+$  ion is succinctly entered as follows.

```

#!/bdf.sh
ROKS/B3lyp/cc-pvdz charge=1

geometry
O
H 1 R1
H 1 R1 2 109.

R1=1.0 # OH bond length in angstrom
end geometry

```

**Hint:** In contrast to Hartree-Fock, Kohn-Sham requires the use of the `dft` keyword in advanced input to specify the exchange-related generic function. For concise input, only the exchange-dependent generic function and basis group need to be specified. The system will choose to use RKS or UKS depending on the spin state, and must be entered explicitly if ROKS is to be used.

#### 4.4.5 Kohn-Sham calculations based on RS heterogeneous generalizations

CAM-B3LYP and other RS hybridization generalization functions, dividing the Coulomb interaction into long and short ranges, the

$$\frac{1}{r_{12}} = \frac{1 - [\alpha + \beta \cdot \text{erf}(\mu r_{12})]}{r_{12}} + \frac{\alpha + \beta \cdot \text{erf}(\mu r_{12})}{r_{12}}$$

When using the BDF advanced input, the  $\mu$  parameter can be adjusted using the keyword `RS` in the `xuanyuan` module. the default  $\mu$  parameter for CAM-B3lyp is 0.33. other  $\mu$  values in RS Hybrid GGA see `RSOMEGA` keyword. for example, for 1,3-Butadiene molecules, the RKS advanced mode input using CAM-B3lyp is.

```

$compass
basis
cc-pVDZ
geometry
C -2.18046929 0.68443844 -0.00725330
H -1.64640852 -0.24200621 -0.04439369

```

(continues on next page)

```
H -3.24917614 0.68416040 0.04533562
C -1.50331750 1.85817167 -0.02681816
H -0.43461068 1.85844971 -0.07940766
C -2.27196552 3.19155924 0.02664018
H -3.34067218 3.19128116 0.07923299
C -1.59481380 4.36529249 0.00707382
H -2.12887455 5.29173712 0.04421474
H -0.52610710 4.36557056 -0.04551805
end geometry
$end

$xuanyuan
rs
0.33 # define mu=0.33 in CAM-B3lyp functional
$end

$scf
rks # restricted Kohn-Sham
dft
cam-b3lyp
$end
```

#### 4.4.6 Exact exchange term and correlation term components for custom heterogeneous generalized functions, double heterogeneous generalized functions

For some calculations, it may be necessary for the user to manually adjust the exact exchange term components of the generic function to obtain satisfactory accuracy. In this case, you can add the `facex` keyword to the `$scf` module, for example, to change the exact exchange term component of the B3LYP generic function from the default 20% to 15%, you can write .. code-block:: bdf

```
$scf ...dft
    b3lyp
facex 0.15
$end
```

Similarly, it is possible to customize the MP2-related term components of a two-hybrid generic function with the `facco` keyword. Note that not all generic functions support custom `facex` and `facco` (see the SCF module for a list of keywords ).

## 4.4.7 Dispersion correction for weak interactions

Common exchange-correlation general functions such as B3lyp do not describe weak interactions well and require dispersion correction when calculating energy or doing molecular structure optimization. BDF uses the D3 dispersion correction method developed by Stefan Grimme, which requires specifying the D3 keyword in the input to the SCF module, as follows

```
#!/bdf.sh
B3lyp/cc-pvdz

geometry
O
H 1 R1
H 1 R1 2 109.

R1=1.0      # OH bond length in angstrom
end geometry

$scf
D3 # Grimme's dispersion correction
$end
```

### Tip:

- The BDF mixed-mode input method is used here to precisely control the SCF calculation by adding the SCF module keyword on top of the simple input.

The dispersion correction is added at the end of the Kohn-Sham calculation, and the calculated output is as follows.

```
diis/vshift is closed at iter = 8
9 0 0.000 -76.380491166 -0.000000000 0.000000017 0.000000168 0.0000 0.02

Label          CPU Time          SYS Time          Wall Time
SCF iteration time:      0.467 S          0.033 S          0.233 S

Final DeltaE = -7.5459638537722640E-011
Final DeltaD = 1.6950036756030376E-008 5.0000000000000002E-005

Final scf result
E_tot = -76.38106481
E_ele = -85.17290149
E_disp= -0.00057364
E_nn = 8.79183668
```

(continues on next page)

(continued from previous page)

```

E_1e = -122.51287853
E_ne = -198.42779201
E_kin = 75.91491348
E_ee = 44.84995532
E_xc = -7.50940464
Virial Theorem 2.006140

```

Here the total energy  $E_{\text{tot}}$  includes the dispersion correction energy,  $E_{\text{disp}} = -0.00057364$  .

#### 4.4.8 Improving the accuracy of integration lattice points for Kohn-Sham calculations

Although the BDF defines default integration grid points for different general functions according to their accuracy requirements (e.g., the Meta-GGA class of general functions requires high integration grid points, and the BDF defaults to Fine grid points), the user may also wish to adjust the integration grid points. The valid values of the Grid are Ultra coarse , Coarse , medium , fine , Ultra fine , sg1 etc. From Ultra coarse to sg1, the number of integration points increases and the numerical integration accuracy increases.

Example: M062X calculation of H<sub>2</sub>O molecule. This generalized function is a heterogeneous Meta-GGA type generalized function, which requires a dense integration grid, so the input uses a mixture of advanced input and simple input, as shown below.

```

#!bdf.sh
M062X/cc-pvdz

geometry
O
H 1 R1
H 1 R1 2 109.

R1=1.0 # OH bond length in angstrom
end geometry

$scf
grid # set numerical integration grid as ultra fine
  ultra fine
$end

```

The BDF uses Ultra coarse integration lattice points at the beginning steps of the Kohn-Sham calculation, as shown below.

```

Switch to Ultra Coarse grid ...
[ATOM SCF control]

```

(continues on next page)

(continued from previous page)

```

heff=                      0
After initial atom grid ...
After initial atom grid ...

Generating Numerical Integration Grid.

  1 O      Second Kind Chebyshev ( 21)  Lebedev ( -194)
    Atoms:      1
  2 H      Second Kind Chebyshev ( 21)  Lebedev ( -194)
    Atoms:      2      3
Partition Function:  SSF  Partitioning with Scalar=  0.64.
Gtol, Npblock, Icoulpot, Iop_adaptive :  0.10E-04   128      0      0
Number of symmetry operation =  4

Basis Informations for Self-adaptive Grid Generation, Cutoff=  0.10E-04
  1O      GTO( 14) Ntot=  26 MaxL= 2 MaxNL= 0 MaxRad= 0.530E+01
basis details in form ( N L Zeta Cutradius):
( 1 0  0.117E+05  0.02) ( 1 0  0.176E+04  0.06) ( 1 0  0.401E+03  0.13)
( 1 0  0.114E+03  0.24) ( 1 0  0.370E+02  0.42) ( 1 0  0.133E+02  0.70)
( 1 0  0.503E+01  1.14) ( 1 0  0.101E+01  2.53) ( 1 0  0.302E+00  4.64)
( 2 1  0.177E+02  0.66) ( 2 1  0.385E+01  1.42) ( 2 1  0.105E+01  2.72)
( 2 1  0.275E+00  5.30) ( 3 2  0.119E+01  2.73)
  2H      GTO( 5) Ntot=  7 MaxL= 1 MaxNL= 0 MaxRad= 0.730E+01
basis details in form ( N L Zeta Cutradius):
( 1 0  0.130E+02  0.71) ( 1 0  0.196E+01  1.82) ( 1 0  0.445E+00  3.82)
( 1 0  0.122E+00  7.30) ( 2 1  0.727E+00  3.26)
Numerical Grid Generated SUCCESSFULLY!
Total and symmetry independent Grid Number:      4352      1181

```

When the energy converges to within 0.01 Hartree, it switches to the Ultra fine integration grid point and the output is shown below.

```

Switch to Ultra Fine grid ...
[ATOM SCF control]
heff=                      0
After initial atom grid ...
After initial atom grid ...

Generating Numerical Integration Grid.

  1 O      Second Kind Chebyshev (100)  Lebedev (-1202)
    Atoms:      1
  2 H      Second Kind Chebyshev (100)  Lebedev (-1202)
    Atoms:      2      3

```

(continues on next page)

(continued from previous page)

```

Partition Function: SSF Partitioning with Scalar= 0.64.
Gtol, Npblock, Icoulpot, Iop_adaptive : 0.10E-04 128 0 0
Number of symmetry operation = 4

Basis Informations for Self-adaptive Grid Generation, Cutoff= 0.10E-04
 10 GTO( 14) Ntot= 26 MaxL= 2 MaxNL= 0 MaxRad= 0.530E+01
basis details in form ( N L Zeta Cutradius):
( 1 0 0.117E+05 0.02) ( 1 0 0.176E+04 0.06) ( 1 0 0.401E+03 0.13)
( 1 0 0.114E+03 0.24) ( 1 0 0.370E+02 0.42) ( 1 0 0.133E+02 0.70)
( 1 0 0.503E+01 1.14) ( 1 0 0.101E+01 2.53) ( 1 0 0.302E+00 4.64)
( 2 1 0.177E+02 0.66) ( 2 1 0.385E+01 1.42) ( 2 1 0.105E+01 2.72)
( 2 1 0.275E+00 5.30) ( 3 2 0.119E+01 2.73)
 2H GTO( 5) Ntot= 7 MaxL= 1 MaxNL= 0 MaxRad= 0.730E+01
basis details in form ( N L Zeta Cutradius):
( 1 0 0.130E+02 0.71) ( 1 0 0.196E+01 1.82) ( 1 0 0.445E+00 3.82)
( 1 0 0.122E+00 7.30) ( 2 1 0.727E+00 3.26)
Numerical Grid Generated SUCCESSFULLY!
Total and symmetry independent Grid Number: 94208 24827

```

Here, the integration lattice points of both H and O atoms are 100\*1202, where 100 is the number of radial lattice points and 1202 is the number of angular lattice points.

## 4.5 Symmetry and molecular point groups

BDF supports the consideration of molecular point group symmetries in the computation. Most computational tasks support any real representation point group (all Abelian groups, as well as  $C_{nv}$ ,  $D_n$ ,  $D_{nh}$ ,  $D_{nd}$ ,  $T_d$ ,  $O$ ,  $O_h$ ,  $I$ ,  $I_h$ ; the special point groups  $C_{\infty v}$ ,  $D_{\infty h}$  are nominally supported but are treated as  $C_{20v}$  and  $D_{20h}$  respectively. except for some computational tasks (e.g., open-shell TDDFT, TDDFT/SOC, etc.) that support only  $D_{2h}$  and its subgroups (i.e.,  $C_1$ ,  $C_i$ ,  $C_s$ ,  $C_2$ ,  $D_2$ ,  $C_{2h}$ ,  $C_{2v}$ ,  $D_{2h}$ , generally referred to as Abelian groups) while individual atoms are treated as  $O_h$  groups), but not the complex representation point groups ( $C_n$ ,  $C_{nh}$  ( $n \geq 3$ );  $S_{2n}$  ( $n \geq 2$ );  $T$ ,  $T_h$ ).

The program can automatically determine the point group to which a molecule belongs based on the coordinates of the molecule entered by the user in COMPASS module, and automatically change to the appropriate subgroup when the molecule belongs to the complex representation point group. After determining the point group to which the molecule belongs, the program generates the group operator, the characteristic scale, and the integrable representation of the point group for subsequent calculations. Take ammonia molecule as an example.

```

#! NH3.sh
HF/cc-pVDZ

geometry
N          -0.00000000  -0.00000000  -0.10000001

```

(continues on next page)

(continued from previous page)

```

H          0.00000000  -0.94280900  0.23333324
H         -0.81649655   0.47140450  0.23333324
H          0.81649655   0.47140450  0.23333324
end geometry

$compass
Title
  NH3
thresh
  medium
$end

```

The corresponding advanced input mode in **COMPASS** is

```

$COMPASS
Title
  NH3
Basis
  cc-pvdz
Geometry
N          -0.00000000  -0.00000000  -0.10000001
H           0.00000000  -0.94280900  0.23333324
H         -0.81649655   0.47140450  0.23333324
H          0.81649655   0.47140450  0.23333324
End geometry
thresh
  medium
$END

```

Note that since the initial structure does not strictly satisfy the  $C_{3v}$  symmetry, the `thresh medium` is used here to choose a looser threshold for judging the symmetry (the default is `tight`, or a looser one can be `loose`). As can be seen from the output file, the program automatically identifies the molecule as belonging to the  $C_{3v}$  point group.

```

gsym: C03V, noper= 6
Exiting zgeomsort...
Representation generated
Point group name C(3V) 6
User set point group as C(3V)
Largest Abelian Subgroup C(S) 2

```

Note that the subscripts of the point group names need to be enclosed in parentheses, e.g.  $C_{\infty v}$ ,  $D_{\infty h}$  groups need to be written as `C(LIN)`, `D(LIN)`. Next, the integrable representation information, the CG coefficient table, etc. are printed. At the end of the COMPASS section output, the program gives a list of the integrable representations under the point group and the number of orbits belonging to each integrable representation.

```
|-----|
      Symmetry adapted orbital

Total number of basis functions:      29      29

Number of irreps:    3
Irrep  :    A1      A2      E1
Norb   :    10      1      18
|-----|
```

### 4.5.1 Order of integrable representations

Very often, the user needs to specify in the input file information such as the number of orbits occupied by each integrable representation (specified in the input to the SCF module) and how many excited states are calculated under each integrable representation (specified in the input to the TDDFT module), which is generally provided in the form of an array, e.g.

```
$TDDFT
Nroot
 3 1 2
$END
```

It indicates that the first integrability indicates the calculation of 3 excited states, the second integrability indicates the calculation of 1 excited state, and the third integrability indicates the calculation of 2 (see the TDDFT section of this manual for details). This necessarily requires the user to know the order of the integrable representations inside the BDF program at the time of writing the input file. The order of the integrable representations for all point groups supported by the BDF is given below.

Table 4.1: The order of irreducible representations under different point groups

C(1)	A
C(i)	Ag, Au
C(s)	A', A''
C(2)	A, B
C(2v)	A1, A2, B1, B2
C(2h)	Ag, Bg, Au, Bu
D(2)	A, B1, B3, B2
D(2h)	Ag, B1g, B3g, B2g, Au, B1u, B3u, B2u
C(nv) (n=2k+1, k>=1)	A1, A2, E1, ..., Ek
C(nv) (n=2k+2, k>=1)	A1, A2, B1, B2, E1, ..., Ek
D(n) (n=2k+1, k>=1)	A1, A2, E1, ..., Ek
D(n) (n=2k+2, k>=1)	A1, A2, B1, B2, E1, ..., Ek
D(nh) (n=2k+1, k>=1)	A1', A2', E1', ..., Ek', A1'', A2'', E1'', ..., Ek'',
D(nh) (n=2k+2, k>=1)	A1g, A2g, B1g, B2g, E1g, ..., Ekg, A1u, A2u, B1u, B2u, E1u, ..., Eku
D(nd) (n=2k+1, k>=1)	A1g, A2g, E1g, ..., Ekg, A1u, A2u, E1u, ..., Eku
D(nd) (n=2k+2, k>=1)	A1', A2', B1', B2', E1', ..., Ek', A1'', A2'', B1'', B2'', E1'', ..., Ek''
T(d)	A1, A2, E, T1, T2
O	A1, A2, E, T1, T2
O(h)	A1g, A2g, Eg, T1g, T2g, A1u, A2u, Eu, T1u, T2u
I	A, T1, T2, F, H
I(h)	Ag, T1g, T2g, Fg, Hg, Au, T1u, T2u, Fu, Hu

The user can also force the program to calculate under a subgroup of the point group to which the molecule belongs by using the group keyword in the COMPASS module input, e.g.

```
#! N2.sh
HF/def2-TZVP group=D(2h)

geometry
  N 0.00 0.00 0.00
  N 0.00 0.00 1.10
end geometry
```

or

```
$COMPASS
Title
  N2
Basis
```

(continues on next page)

```
def2-TZVP
Geometry
N 0.00 0.00 0.00
N 0.00 0.00 1.10
End geometry
Group
D(2h)
$END
```

That is, the program is forced to compute the  $N_2$  molecule under the  $D_{2h}$  point group, even though the  $N_2$  molecule actually belongs to the  $D_{\infty h}$  point group. Note that the program automatically checks whether the point group entered by the user is a subgroup of the point group to which the molecule actually belongs; if not, the program quits with an error.

## 4.5.2 Standard orientation

For the convenience of the calculation and the analysis of the results, the program rotates the molecules to the standard orientation after determining the point group used for the calculation, so that the symmetry axes of the molecules coincide as much as possible with the coordinate axes and the symmetry plane is as perpendicular as possible to the coordinate axes. This has the advantage that many quantities involved in the calculation are exactly equal to zero (e.g., some molecular orbital coefficients, some components of the gradient, etc.), which makes it easier to analyze the results.

The BDF determines the standard orientation of a molecule according to the following rules.

1. take a weighted average of all atomic coordinates of the molecule by nuclear charge to obtain the nuclear charge center of the molecule, and then translate the molecule so that the nuclear charge center is at the origin of the coordinate system.
2. if the molecule has an axis of symmetry, rotate the highest order axis of symmetry (principal axis) of the molecule to the z-axis direction.
3. if the molecule has  $\sigma_v$  symmetry planes, rotate one of the  $\sigma_v$  symmetry planes to the xz-plane direction, keeping the major axis direction constant in the process.
4. if the molecule has other dual or quadruple axes besides the principal axes, rotate one of the axes (any of the quadruple axes if they exist, otherwise any of the dual axes) to the x-axis direction, keeping the direction of the principal axes unchanged in the process.
5. if the above conditions cannot uniquely determine the orientation of the molecule because the symmetry of the molecule is too low, rotate the molecule so that the axis of inertia (i.e., the eigenvector of rotational inertia) of the molecule and each coordinate axis are oriented in the same direction.

For some special cases, the above rule still cannot uniquely determine the orientation of the molecule. For example, molecules belonging to the  $C_{2v}$  point group have two  $\sigma_v$  symmetry planes, and either of them may be rotated to the xz direction at step 3 of the above rule. In the BDF, a  $C_{2v}$  molecule with a planar structure, such as a water molecule, will be rotated to the xz plane.

```

-----|
Atom   Cartcoord(Bohr)                Charge Basis Auxbas Uatom Nstab Alink  Mass
O      0.000000  -0.000000  0.219474  8.00  1    0    0    0    E   15.9949
H     -1.538455   0.000000 -0.877896  1.00  2    0    0    0    E    1.0073
H      1.538455  -0.000000 -0.877896  1.00  2    0    0    0    E    1.0073
-----|

```

In contrast, other quantization programs may choose to rotate the numerator to the yz plane. This leads to another problem: according to the customary convention, the  $x$  operator under the  $C_{2v}$  point group belongs to the B1 integrable representation and the  $y$  operator belongs to the B2 integrable representation, so if a quantization program chooses to rotate the numerator to the yz plane, its B1 and B2 integrable representations are defined opposite to the BDF, i.e., the B1 representation of the program corresponds to the B2 representation of the BDF, and the B2 representation of the program corresponds to the B1 representation of the BDF. And if the molecule of this  $C_{2v}$  point group is not a planar structure (e.g., ethylene oxide), it is even more difficult to predict whether the standard orientation of the molecule in the BDF is consistent with other quantization software. Therefore, if the user wishes to calculate the molecules of the  $C_{2v}$  point group and compare the results with those of other quantization programs (or try to duplicate the results calculated by other quantization programs in the literature), the user must confirm how the B1 and B2 representations of the quantization program correspond to the BDF.

## 4.6 Other techniques for self-consistent field calculations

### 4.6.1 Initial Guesses for Self-Consistent Field Calculations

The initial guess track of the self-consistent field calculation has a great impact on the convergence of the calculation. the BDF supports several initial guesses, as follows.

- Atom : Combining molecular density matrix guesses using atomic density matrix, default option.
- Huckel : semi-empirical Huckel method guess.
- Hcore : diagonalized single-electron Hamiltonian guess.
- Readmo : read in molecular orbitals as initial guess.
- \* Readdm : read in the density matrix as initial guess

BDF defaults to Atom guesses. The initial guess of the BDF can be changed in concise input mode using the keyword guess, as follows

```

#! ch3cho.sh
HF/6-31G guess=Hcore unit=Bohr

```

(continues on next page)

(continued from previous page)

```

geometry      # notice: unit in Bohr
C           0.1727682300      -0.0000045651      -0.8301598059
C          -2.3763311896       0.0000001634       0.5600567139
H           0.0151760290       0.0000088544      -2.9110013387
H          -2.0873396672       0.0000037621       2.5902220967
H          -3.4601725077      -1.6628370597       0.0320271859
H          -3.4601679801       1.6628382651       0.0320205364
O           2.2198078005       0.0000024315       0.2188182082
end geometry

```

Here, we use the keyword `guess=Hcore` in the second line to specify the use of the Hcore guess. 18 iterations of the SCF converge.

Iter.	idiis	vshift	SCF Energy	DeltaE	RMSDeltaD	MaxDeltaD	Damping	Times(S)
1	0	0.000	-130.488739529	174.680929376	0.401531162	5.325668770	0.0000	0.03
2	1	0.000	-115.595786784	14.892952744	0.407402695	5.323804678	0.0000	0.02
3	2	0.000	-126.823748834	-11.227962049	0.115300517	1.591646800	0.0000	0.03
4	3	0.000	-150.870636785	-24.046887951	0.011394798	0.154813426	0.0000	0.02
5	4	0.000	-151.121829169	-0.251192384	0.004498398	0.037875784	0.0000	0.03
6	5	0.000	-150.900123989	0.221705180	0.008483436	0.119865266	0.0000	0.02
7	6	0.000	-151.582006133	-0.681882144	0.011892345	0.122063906	0.0000	0.02
8	7	0.000	-152.441656890	-0.859650757	0.007907887	0.062113717	0.0000	0.03
9	8	0.000	-152.729229838	-0.287572947	0.003318529	0.037884676	0.0000	0.02
10	2	0.000	-152.795374919	-0.066145081	0.005951772	0.054625652	0.0000	0.02
11	3	0.000	-152.839276725	-0.043901806	0.000860488	0.010210210	0.0000	0.03
12	4	0.000	-152.841131472	-0.001854746	0.000733951	0.007678730	0.0000	0.02
13	5	0.000	-152.841752921	-0.000621449	0.000348937	0.003519950	0.0000	0.02
14	6	0.000	-152.841816238	-0.000063316	0.000053288	0.000787592	0.0000	0.03
15	7	0.000	-152.841819180	-0.000002942	0.000021206	0.000157533	0.0000	0.02
16	8	0.000	-152.841819505	-0.000000325	0.000004796	0.000031694	0.0000	0.02
17	2	0.000	-152.841819522	-0.000000016	0.000000698	0.000005497	0.0000	0.03
18	3	0.000	-152.841819522	-0.000000000	0.000000236	0.000002276	0.0000	0.02
diis/vshift is closed at iter = 18								
19	0	0.000	-152.8418195227	-0.000000000	0.000000078	0.000000848	0.0000	0.03

**Warning:** The unit of numerator input for this example is Bohr, and the keyword `unit=Bohr` must be used to specify that the length of the coordinates is in Bohr.

This example corresponds to the advanced input

```

$compass
geometry

```

(continues on next page)

(continued from previous page)

```

C 0.1727682300 -0.0000045651 -0.8301598059
C -2.3763311896 0.0000001634 0.5600567139
H 0.0151760290 0.0000088544 -2.9110013387
H -2.0873396672 0.0000037621 2.5902220967
H -3.4601725077 -1.6628370597 0.0320271859
H -3.4601679801 1.6628382651 0.0320205364
O 2.2198078005 0.0000024315 0.2188182082

end geometry
unit # set unit of coordinates as Bohr
    bohr
basis
    6-31g
$end

$xuanyuan
$end

$scf
rhf
guess # ask for hcore guess
    hcore
$end

```

## 4.6.2 Reading in the initial guess orbitals

By default, the SCF calculation in BDF uses the atomic density matrix to construct the molecular density matrix to generate the initial guess orbitals. In practice, the user can read in the converged SCF molecular orbitals as the initial guess orbitals for the current SCF calculation. In this example, we first calculate a neutral H<sub>2</sub>O molecule and get the converged orbitals as the initial guess orbitals for the H<sub>2</sub>O<sup>+</sup> ions.

In the first step, the H<sub>2</sub>O molecule is calculated and the input file is prepared and named as `h2o.inp`. The contents are as follows:

```

#!bdf.sh
RKS/B3lyp/cc-pvdz

geometry
O
H 1 R1
H 1 R1 2 109.

R1=1.0 # OH bond length in angstrom
end geometry

```

After performing the calculation, the working directory generates the readable file `h2o.scforb`, which holds the convergence orbits of the SCF calculation.

In the second step, the convergence orbit of the  $\text{H}_2\text{O}$  molecule is used as an initial guess for the  $\text{H}_2\text{O}^+$  ion calculation, and the input file `h2o+.inp` is prepared, with the following contents.

```
#!/bdf.sh
ROKS/B3lyp/cc-pvdz guess=readmo charge=1

geometry
O
H 1 R1
H 1 R1 2 109.

R1=1.0      # OH bond length in angstrom
end geometry

%cp $BDF_WORKDIR/h2o.scforb $BDF_TMPDIR/${BDFTASK}.inorb
```

Here, key word `guess=readmo` is used to assign the initial guess orbital to be read. Initial guess orbital is copied by using `%` guided copy command from `h2o.scforb` in environmental variable `BDF_WORKDIR` defined file to `${BDFTASK}.inorb` file in `BDF_TMPDIR`. Here, `BDF_WORKDIR` is the directory of executing tasks, `BDF_TMPDIR` is directory to store temporary files in BDF.

### 4.6.3 Extending small basis group convergence orbits to large basis group initial guesses

Initial guess orbits can be generated from different basis groups, again to accelerate computational convergence. This requires an extension of the initial guess track file. The track extensions should use the same base group, such as the `cc-pVXZ` series, `ANO-RCC` series, and other base groups. The orbital expansion currently supports only the advanced input mode. For  $\text{CH}_3\text{CHO}$  molecules, the orbitals are first calculated with `cc-pVDZ` and then expanded to the initial guess orbitals calculated with the `cc-pVQZ` basis set with the following inputs.

```
# First SCF calculation using small basis set cc-pvdz
$compass
geometry
C      0.1727682300      -0.0000045651      -0.8301598059
C     -2.3763311896       0.0000001634       0.5600567139
H      0.0151760290       0.0000088544      -2.9110013387
H     -2.0873396672       0.0000037621       2.5902220967
H     -3.4601725077      -1.6628370597       0.0320271859
H     -3.4601679801       1.6628382651       0.0320205364
O      2.2198078005       0.0000024315       0.2188182082
end geometry
```

(continues on next page)

(continued from previous page)

```

basis
  cc-pvdz
unit # set unit of coordinates as Bohr
  Bohr
$end

$xuanyuan
$end

$scf
rhf
$end

#change chkfil name into chkfil1
%mv $BDF_WORKDIR/$BDFTASK.chkfil $BDF_WORKDIR/$BDFTASK.chkfil1

$compass
geometry
C      0.1727682300      -0.0000045651      -0.8301598059
C      -2.3763311896      0.0000001634      0.5600567139
H       0.0151760290      0.0000088544      -2.9110013387
H      -2.0873396672      0.0000037621      2.5902220967
H      -3.4601725077     -1.6628370597      0.0320271859
H      -3.4601679801      1.6628382651      0.0320205364
O       2.2198078005      0.0000024315      0.2188182082
end geometry
basis
  cc-pvqz
unit
  Bohr
$end

# change chkfil to chkfil1. notice, should use cp command since we will use
# "$BDFTASK.chkfil" in the next calculation
%cp $BDF_WORKDIR/$BDFTASK.chkfil $BDF_WORKDIR/$BDFTASK.chkfil2

# copy converged SCF orbital as input orbital of the module expandmo
%cp $BDF_WORKDIR/$BDFTASK.scforb $BDF_WORKDIR/$BDFTASK.inporb

# Expand orbital to large basis set. The output file is $BDFTASK.exporb
$expandmo
overlap
$end

```

(continues on next page)

(continued from previous page)

```

$xuanyuan
$end

# use expanded orbital as initial guess orbital
%cp $BDF_WORKDIR/$BDFTASK.exporb $BDF_WORKDIR/$BDFTASK.scforb
$scf
RHF
guess
  readmo
iprtmo
  2
$end

```

In the above input, the first RHF calculation is performed using the **cc-pVDZ** basis set, then the convergence track from the first SCF calculation is extended to the **cc-pVQZ** basis set using the `expandmo` module, and finally `guess=readmo` is used as the initial guess track to be read into the SCF.

The output of the `expandmo` module is that

```

|*****|
|
| Start running module expandmo
| Current time   2021-11-29  22:20:50
|
|*****|
| $expandmo
| overlap
| $end
| /Users/bsuo/check/bdf/bdfpro/ch3cho_exporb.chkfil1
| /Users/bsuo/check/bdf/bdfpro/ch3cho_exporb.chkfil2
| /Users/bsuo/check/bdf/bdfpro/ch3cho_exporb.inporb
| Expanding MO from small to large basis set or revise ...
|
| 1 Small basis sets
|
| Number of basis functions (NBF):      62
| Maxium NBF of shell :                 6
|
| Number of basis functions of small basis sets:      62
|
| 2 Large basis sets
|
| Number of basis functions (NBF):      285

```

(continues on next page)

(continued from previous page)

```

Maxium NBF of shell :          15

Overlap expanding :              1
Read guess orb
Read orbital title:  TITLE - SCF Canonical Orbital
nsbas_small  62
nsbas_large 285
ipsmall      1
iplarge      1
Overlap of dual basis ...
Overlap of large basis ...
Write expanded MO to scratch file ...
|*****|

Total cpu    time:           0.42  S
Total system time:           0.02  S
Total wall   time:           0.47  S

Current time  2021-11-29  22:20:51
End running module expandmo
|*****|

```

It can be seen that the small base group has 62 tracks and the large base group has 285 tracks. expandmo reads in the regular tracks for SCF convergence, extends them to the large base group and writes them to a temporary file.

The output of the second SCF calculation is that

```

/Users/bsuo/check/bdf/bdfpro/ch3cho_exporb.scforb
Read guess orb:  nden=1  nreps= 1  norb=  285  lenmo=  81225
Read orbital title:  TITLE - orthognal Expand CMO
Orbitals initialization is completed.

.....
Iter.  idiis vshift  SCF Energy   DeltaE   RMSDeltaD   MaxDeltaD   Damping Times(S)
1     0   0.000 -152.952976892  122.547522034  0.002218985  0.246735859  0.0000  16.30
2     1   0.000 -152.983462881  -0.030485988  0.000367245  0.026196100  0.0000  16.83
3     2   0.000 -152.983976045  -0.000513164  0.000086429  0.006856831  0.0000  17.18
4     3   0.000 -152.984012062  -0.000036016  0.000016763  0.001472939  0.0000  17.02
5     4   0.000 -152.984019728  -0.000007666  0.000010400  0.001012788  0.0000  17.42
6     5   0.000 -152.984021773  -0.000002045  0.000003396  0.000328178  0.0000  17.28
7     6   0.000 -152.984022197  -0.000000423  0.000001082  0.000075914  0.0000  17.40
8     7   0.000 -152.984022242  -0.000000044  0.000000154  0.000008645  0.0000  17.28
9     8   0.000 -152.984022243  -0.000000001  0.000000066  0.000005087  0.0000  19.38
diis/vshift is closed at iter =  9

```

(continues on next page)

(continued from previous page)

```
10    0    0.000 -152.984022243  -0.000000000  0.000000007  0.000000584  0.0000  18.95
```

Label	CPU Time	SYS Time	Wall Time
SCF iteration time:	517.800 S	0.733 S	175.617 S

#### 4.6.4 Calculation of excited states by the maximum occupation of molecular orbitals (mom) method

The mom (maximum occupation method) is a  $\Delta$ SCF method that can be used to calculate excited states.

```
#-----
#
# mom method: J. Liu, Y. Zhang, and W. Liu, J. Chem. Theory Comput. 10, 2436 (2014).
#
# gs = -169.86584128
# ab = -169.62226127
# T = -169.62483480
# w(S) = 6.69eV
#-----
$COMPASS
Title
mom
Basis
6-311++GPP
Geometry
C      0.000000    0.418626    0.000000
H     -0.460595    1.426053    0.000000
O      1.196516    0.242075    0.000000
N     -0.936579   -0.568753    0.000000
H     -0.634414   -1.530889    0.000000
H     -1.921071   -0.362247    0.000000
End geometry
Check
$END

$XUANYUAN
$END

$SCF
UKS
DFT
B3LYP
```

(continues on next page)

(continued from previous page)

```
alpha
  10 2
beta
  10 2
$END

%cp ${BDFTASK}.scforb $BDF_TMPDIR/${BDFTASK}.inporb

# delta scf with mom
$SCF
UKS
DFT
B3LYP
guess
  readmo
alpha
  10 2
beta
  10 2
ifpair
hpalpha
  1
  10 0
  11 0
iaufbau
  2
$END

# pure delta scf for triplet
$SCF
UKS
DFT
B3LYP
alpha
  11 2
beta
  9 2
$END
```

This example performs three SCF calculations.

- For the first SCF, the ground state of the formamide molecule is calculated using the UKS method. The input specifies the occupancy of alpha and beta orbitals using the alpha and beta keywords, respectively. The base state of the formamide molecule is the singlet state S0, where the specified alpha and beta occupancies are the same. 10

2 The integrable designation indicates that A' and A'' have 10 and 2 occupied orbitals, respectively. The SCF module will fill the orbitals with electrons according to the construction principle, from low to high orbital energy.

- For the second SCF, the S1 state of the formamide molecule is calculated using the UKS and mom methods. The key points here are: 1. the convergent orbitals read into the previous UKS step are specified using guess=readmo; 2. the occupation number of each symmetry orbital is set using alpha, beta keywords; 3. the variable ifpair is set, which needs to be used in conjunction with hpalha, hpbeta to specify the hole-particle - HP) orbital pairs for electronic excitation; 4. The variable hpalha is set to specify the HP orbital pairs for excitation. The number 1 indicates the excitation of a pair of HP orbitals, and the following two rows specify the orbital excitation. The first column indicates the excitation of electrons from the 10th alpha orbital to the 11th alpha orbital in the first integrable representation; the elements of the second column are all zero, indicating that no excitation is done for the orbital in the second integrable representation; 5. The iaufbau variable is set to 2, specifying that the mom calculation is to be performed.
- For the third SCF, the T1 state of the formamide molecule is calculated using the UKS method. In the input, we specify the orbital occupancies using the alpha and beta keywords, where the number of occupancies of the alpha orbital is 11 2, indicating 11 and 2 electrons occupying the alpha orbital with symmetries A' and A'', respectively, and the occupancy of the beta orbital is 9 2. Since the required state for the solution is the lowest energy state for a given number of orbital occupancies, there is no need to specify iaufbau.

Here, the first SCF calculation converges to the result that

```

Superposition of atomic densities as initial guess.
skipaocheck T F
Solve HC=EC in pflmo space. F      12      75
Initial guess energy = -169.2529540680

[scf_cycle_init_ecdenpot]
Meomory for coulpotential      0.00 G

Start SCF iteration.....

Iter.  idiis vshift  SCF Energy      DeltaE      RMSDeltaD      MaxDeltaD      Damping Times(S)
 1     0   0.000 -169.411739263 -0.158785195  0.005700928  0.163822560  0.0000  0.20
Turn on DFT calculation ...
 2     1   0.000 -169.743175119 -0.331435856  0.008905349  0.340815886  0.0000  0.42
 3     2   0.000 -169.232333660  0.510841459  0.006895796  0.296788710  0.0000  0.43
 4     3   0.000 -169.863405142 -0.631071482  0.000364999  0.015732911  0.0000  0.43
 5     4   0.000 -169.863345847  0.000059295  0.000209771  0.009205878  0.0000  0.42
 6     5   0.000 -169.865811301 -0.002465454  0.000027325  0.000606909  0.0000  0.43
 7     6   0.000 -169.865831953 -0.000020651  0.000008039  0.000357726  0.0000  0.43
 8     7   0.000 -169.865833199 -0.000001246  0.000003927  0.000114311  0.0000  0.42
 9     8   0.000 -169.865833401 -0.000000201  0.000000182  0.000004399  0.0000  0.43
diis/vshift is closed at iter = 9
10    0   0.000 -169.865833402 -0.000000000  0.000000139  0.000003885  0.0000  0.43

```

(continues on next page)

(continued from previous page)

```

Label                CPU Time      SYS Time      Wall Time
SCF iteration time:    8.650 S       0.700 S       4.050 S

Final DeltaE = -4.4343551053316332E-010
Final DeltaD =  1.3872600382452641E-007  5.0000000000000002E-005

Final scf result
  E_tot =          -169.86583340
  E_ele =          -241.07729109
  E_nn =             71.21145769
  E_1e =          -371.80490197
  E_ne =          -541.14538673
  E_kin =           169.34048477
  E_ee =           148.48285541
  E_xc =           -17.75524454
Virial Theorem       2.003102

```

It can be seen that the first SCF calculation uses the atom guess and the energy of  $S_0$  is calculated to be -169.8658334023 a.u.. The second SCF calculation reads in the convergent orbitals of the first SCF and does the SCF calculation using the mom method, and the output file first indicates that the molecular orbitals were read in and gives the occupation

```

[Final occupation pattern: ]

Irreps:      A'      A''

detailed occupation for iden/irep:      1  1
  1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
  0.00 0.00 0.00 0.00 0.00 0.00

detailed occupation for iden/irep:      1  2
  1.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
  0.00
Alpha      10.00    2.00

```

Here, the 10th alpha orbital of the  $A'$  integrable representation is the occupied orbital and the 11th orbital is the empty orbital. The second SCF calculation reads in the converged orbitals of the first SCF and does the SCF calculation using the mom method, where the input asks to excite the electrons of the 10th orbital represented by  $A'$  to the 11th

orbital. The output file first suggests that the molecular orbitals were read in and gives the occupation

```

Read initial orbitals from user specified file.

/tmp/20117/mom_formamide.inporb
Read guess orb:  nden=2  nreps= 2  norb=  87  lenmo=  4797
Read orbital title:  TITLE - SCF Canonical Orbital

Initial occupation pattern: iden=1  irep= 1  norb(irep)=  66
  1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 0.00
  1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
  0.00 0.00 0.00 0.00 0.00 0.00

Initial occupation pattern: iden=1  irep= 2  norb(irep)=  21
  1.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
  0.00

Initial occupation pattern: iden=2  irep= 1  norb(irep)=  66
  1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
  0.00 0.00 0.00 0.00 0.00 0.00

Initial occupation pattern: iden=2  irep= 2  norb(irep)=  21
  1.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
  0.00

```

Here, iden=1 is the alpha orbital and irep=1 refers to the first integrable representation, and there are a total of norb=66 orbitals, where the occupation number of the 10th orbital is 0.00 and the occupation number of the 11th orbital is 1.00. After 14 SCF iterations, the converged S1 state energy is -169.6222628003 a.u., as follows.

Iter.	idiis	vshift	SCF Energy	DeltaE	RMSDeltaD	MaxDeltaD	Damping	Times (S)
1	0	0.000	-169.505632070	125.031578610	0.020428031	1.463174456	0.0000	0.45

(continues on next page)

(continued from previous page)

```

2   1   0.000 -169.034645773   0.470986296   0.036913522   1.562284831   0.0000   0.43
3   2   0.000 -165.750862892   3.283782881   0.032162782   1.516480990   0.0000   0.43
4   3   0.000 -169.560678610  -3.809815718   0.008588866   0.807859419   0.0000   0.43
5   4   0.000 -169.596211021  -0.035532411   0.003887621   0.367391029   0.0000   0.42
6   5   0.000 -169.620128518  -0.023917496   0.001826050   0.172456003   0.0000   0.43
7   6   0.000 -169.621976725  -0.001848206   0.000486763   0.044630527   0.0000   0.43
8   7   0.000 -169.622245116  -0.000268391   0.000113718   0.004980035   0.0000   0.43
9   8   0.000 -169.622261269  -0.000016153   0.000112261   0.009715905   0.0000   0.42
10  2   0.000 -169.622262553  -0.000001284   0.000043585   0.004092668   0.0000   0.42
11  3   0.000 -169.622262723  -0.000000169   0.000031601   0.002792075   0.0000   0.42
12  4   0.000 -169.622262790  -0.000000067   0.000010125   0.000848297   0.0000   0.43
13  5   0.000 -169.622262798  -0.000000007   0.000003300   0.000273339   0.0000   0.43
diis/vshift is closed at iter = 13
14  0   0.000 -169.622262800  -0.000000002   0.000001150   0.000079378   0.0000   0.42

```

```

Label          CPU Time      SYS Time      Wall Time
SCF iteration time:      13.267 S      0.983 S      6.000 S

```

```
Final DeltaE = -1.8403909507469507E-009
```

```
Final DeltaD =  1.1501625138328933E-006  5.0000000000000002E-005
```

```
Final scf result
```

```

E_tot =          -169.62226280
E_ele =          -240.83372049
E_nn =           71.21145769
E_1e =          -368.54021347
E_ne =          -537.75897296
E_kin =           169.21875949
E_ee =           145.28871749
E_xc =           -17.58222451
Virial Theorem      2.002385

```

```
[Final occupation pattern: ]
```

```
Irreps:      A'      A''
```

```

detailed occupation for iden/irep:      1  1
  1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 0.00
  1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

```

(continues on next page)

(continued from previous page)

```
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00
```

After convergence of SCF, the orbital occupancy is printed again and it can be seen that the 10th orbital in the **alpha** orbital with A' integrable representation has no electron occupation and the 11th orbital has one electron occupation and the 11th orbital is occupied by one electron.

The third SCF calculation gives the **T1** state energy as -169.6248370697 a.u. The output is as follows :

```
Iter.  idiis vshift  SCF Energy      DeltaE      RMSDeltaD      MaxDeltaD      Damping Times (S)
 1     0   0.000 -169.411739263  -0.158785195  0.083821477  9.141182225  0.0000  0.17
Turn on DFT calculation ...
 2     1   0.000 -169.480549474  -0.068810211  0.066700318  6.978728919  0.0000  0.40
 3     2   0.000 -169.277735673   0.202813801  0.014778190  0.648183923  0.0000  0.42
 4     3   0.000 -169.613991196  -0.336255522  0.005923909  0.621843348  0.0000  0.42
 5     4   0.000 -169.620096778  -0.006105582  0.001967168  0.164506160  0.0000  0.40
 6     5   0.000 -169.623636999  -0.003540220  0.002722812  0.246425639  0.0000  0.42
 7     6   0.000 -169.624704514  -0.001067515  0.001064536  0.098138798  0.0000  0.42
 8     7   0.000 -169.624814882  -0.000110368  0.000525436  0.046392861  0.0000  0.42
 9     8   0.000 -169.624834520  -0.000019637  0.000179234  0.012966641  0.0000  0.42
10     2   0.000 -169.624836694  -0.000002174  0.000063823  0.004902276  0.0000  0.42
11     3   0.000 -169.624836922  -0.000000227  0.000017831  0.001440089  0.0000  0.43
12     4   0.000 -169.624837025  -0.000000103  0.000034243  0.002618897  0.0000  0.42
13     5   0.000 -169.624837065  -0.000000039  0.000006158  0.000466001  0.0000  0.40
14     6   0.000 -169.624837068  -0.000000003  0.000003615  0.000354229  0.0000  0.42
diis/vshift is closed at iter = 14
15     0   0.000 -169.624837069  -0.000000001  0.000000966  0.000070404  0.0000  0.42

Label          CPU Time      SYS Time      Wall Time
SCF iteration time:      13.150 S      0.950 S      5.967 S

Final DeltaE = -1.1375220765330596E-009
Final DeltaD =  9.6591808698539483E-007  5.0000000000000002E-005

Final scf result
  E_tot =          -169.62483707
  E_ele =          -240.83629476
  E_nn =           71.21145769
  E_1e =          -368.57834907
  E_ne =          -537.80483706
  E_kin =          169.22648799
  E_ee =          145.32683246
  E_xc =          -17.58477815
Virial Theorem      2.002354
```

## 4.6.5 Handling Non-Convergence of Self-Consistent Field Calculations

When the SCF calculation is completed, the user must check whether the SCF has converged or not. The user must check the convergence of the SCF and only if it converges can the results of the SCF calculation (energy, Bourget analysis, orbital energy, etc.) be used and subsequent calculations performed. Note that the convergence of the SCF cannot be judged only by the presence or absence of errors at the end of the output file. Because even if the SCF does not converge, the program does not exit immediately, but only after the output of the SCF iterations and before the output of the SCF energy. indicates that

```
Warning !!! Total energy not converged!
```

And even in this case, the program still prints the energy, orbital information, and the results of the booster analysis after this information. Although these results cannot be used as official calculation results, they are useful for analyzing the reasons for the non-convergence of the SCF.

Common causes of SCF non-convergence include

1. the HOMO-LUMO energy gap is too small, resulting in repeated changes in the occupation of the frontline orbitals. For example, two orbitals  $\psi_1$  and  $\psi_2$ , at the Nth SCF iteration  $\psi_1$  is the occupied orbitals and  $\psi_2$  is the empty orbitals, however, after constructing the Fock matrix based on such orbital occupancy and diagonalizing it, the orbitals of the N+1th SCF iteration are obtained, but the orbital energy of  $\psi_1$  is higher than that of  $\psi_2$ , so the electrons are transferred from  $\psi_1$  orbitals to  $\psi_2$  orbitals. But then, the Fock matrix of the N+1th SCF iteration changes a lot compared to the Nth SCF iteration, resulting in a lower orbital energy of  $\psi_1$  than  $\psi_2$  in the N+2nd SCF iteration, so the orbital occupation number returns to the case of the Nth SCF iteration, and thus the orbital occupation number of the SCF iteration always changes and never converges. This situation is typified by the fact that the SCF energy alternately oscillates between two energies (or oscillates irregularly within a certain range) with an oscillation amplitude around  $10^{-4} \sim 1$  Hartree, and the orbital occupation number printed at the end of the SCF is not as expected.
2. The HOMO-LUMO energy gap is small, and although the orbital occupation number does not change for each step of the iteration, the orbital shape changes repeatedly, leading to non-convergence of the SCF oscillation. The typical performance of this case is similar to the previous one, but the amplitude of the oscillation is generally slightly smaller and the orbital occupation number printed at the end of the SCF is qualitatively consistent with the expectation.
3. The numerical integration grid point is too small or the accuracy of the two-electron integration is too low, resulting in small oscillations of the SCF that do not converge due to numerical errors. This situation is typically characterized by irregular oscillations of the SCF energy with an amplitude below  $10^{-4}$  Hartree, and the number of orbital occupations printed at the end of the SCF is qualitatively as expected.
4. The basis group is nearly linearly correlated, or the projection of the basis group on the grid is nearly linearly correlated because the grid is too small. This situation is typically characterized by the SCF energy varying by more than 1 Hartree (not necessarily in oscillation, but also monotonically or essentially monotonically), the SCF energy being much lower than expected, and the number of orbit occupancies printed at the end of the SCF being completely unphysically realistic. When the SCF energy is very much lower than expected, the SCF energy may not even be displayed as a number, but as a string of asterisks.

The following part is the common solution to various kinds of SCF convergence failure (also applied to programs

other than BDF):

1. add an energy shift vshift, for both category 1 and category 2 cases, by adding to the \$scf module of the input file.

```
vshift  
0.2
```

If significant oscillations are still observed, it shall gradually increase vshift until convergence. vshift tends to make the convergence of SCF monotonic, but setting vshift too large increases the number of iterations used to converge, so it is appropriate to increase maxiter when increasing vshift. When vshift increases to 1.0 and still fails to converge, one should Consider other methods.

2. increase the density matrix damping damp for the class 2 case (it also has a slight effect on the class 1 case) by adding to the \$scf module of the input file.

```
damp  
0.7
```

Note that damp can be used in conjunction with vshift, and the two effects are to some extent mutually reinforcing. If significant oscillations are still observed with damping set to 0.7, increase the damping while ensuring that it is less than 1. For example, next try 0.9, 0.95, etc. Similar to vshift, damp also tends to improve the monotonicity of SCF convergence, but too large a damp leads to slower convergence, so maxiter can be increased. when damp is increased to 0.99 and still fails to converge, other methods should be considered.

3. turn off DIIS for cases 1 and 2, and when increasing vshift and damp does not converge, DIIS will speed up SCF convergence in most cases, but it may slow down or even prevent convergence when the HOMO-LUMO energy gap is particularly small. In the latter case, you can turn off DIIS by adding the NoDIIS keyword to the \$scf module, increase maxiter, and set vshift and damp depending on the convergence.
4. Turn off SMH, applied to the 1 and 2 situation and the above three solvation failed to converge. The solvation is to add NoSMH key word in \$scf module, increase maxiter, and set vshift and damp depending on the convergence. We have not yet encountered a situation where SMH does not converge and does not converge, but since SMH is a very new method for convergence of SCF, turning off SMH can be an alternative. However, since SMH is a very new method for convergence of SCF, we cannot rule out that SMH may have a negative impact on convergence in rare cases, so turning off SMH can be an alternative.
5. Switch to the FLMO or iOI method for cases of type 1 and type 2, where the molecules are large (e.g. larger than 50 atoms) and where it is suspected that the SCF does not converge because the initial guessing accuracy of the atoms is too low or because of qualitative errors. See the sections on FLMO and iOI methods .
6. Calculate a similar system that converges more easily and then use the wave function of that system as an initial guess to converge the original system, for both type 1 and type 2 cases. For example, if the SCF calculation of a neutral dibasic transition metal complex does not converge, one can calculate the monovalent cation of its closed-shell layer and use the orbitals of the monovalent cation as the first guess for the SCF calculation of the neutral molecule after convergence (but note that since BDF does not yet support reading the RHF/RKS wave function as the first guess for the UHF/UKS calculation, the monovalent cation of the closed-shell layer should be calculated

using UHF /UKS calculation). In extreme cases it is even possible to calculate the higher valence cations first, then add a small number (e.g. 2) of electrons to reconverge the SCF, then add a small number of subs, and so on until the original wave function of the neutral system is obtained. Another common tool is to perform the SCF calculation under the small basis group first, and then use the expandmo module to project the SCF orbitals of the small basis group onto the original basis group after convergence, and then iterate the SCF under the original basis group until convergence.

7. Increase the grid points, which is applicable to the case of type 3 and sometimes also valid for the case of type 4. This is done by using grid keywords, e.g.

```
grid
fine
```

Note: (1) For meta-GGA generalized functions, the default grid point is already fine, so the grid point should be set to ultra fine; (2) Increasing the grid point will increase the time spent in each SCF iteration step; (3) Increasing the grid point will make the converged energy incomparable with other calculations without changing the grid, so if you want to compare this calculation with previous calculations, or to compare the energy/free energy obtained from this calculation with other calculations, etc., you must recalculate all relevant calculations already done with the same grid point as this input file. Therefore, if you want to compare this calculation with previous calculations, or if you want to compare the energy/free energy obtained from this calculation with the results of other calculations, etc., you must recalculate all the relevant calculations already done with the same grid points as this input file, even if those calculations already done can converge without increasing the grid points. If the results do not improve after increasing the grid points, you should try other methods; if the results improve but still do not converge, you can further try to change the fine to ultra fine; if it still does not converge, you should consider the following methods.

8. The threshold value for the double electron integration is set tightly for the category 3 case and sometimes for the category 4 case as well. This is done by adding to the SCF module.

```
optscreen
1
```

This method, like increasing the grid point, also increases the time consumed for each SCF iteration and also leads to results that are not comparable to those without the optscreen. This method is only applicable to calculations without MPEC or MPEC+COSX enabled.

9. set the threshold for determining the linear correlation of the base group loose, for the category 4 case. This is done by adding to the \$scf module.

```
checklin
tollin
1.d-6
```

This method will make the converged energy incomparable with other calculations without changing the tollin. It is not recommended to set the tollin larger than 1.d-5, otherwise it will lead to larger errors. If the tollin is set to 1.d-5 and still there is a non-convergence of type 4, then the methods described above, such as increasing the grid point and

changing the two-electron integration threshold, should be considered.

Note that among the above methods, if a method does not converge the SCF but makes it converge better than before, the next method should be tried with

```
guess
readmo
```

The orbit of the last SCF iteration of the previous method is read as the first guess. However, if the previous method backfired and caused the SCF convergence to deteriorate, the next method should be tried either by starting again with an atomic guess or by picking the track of the last iteration of the other previously tried method as the first guess (this of course requires the user to back up the tracks obtained for each SCF convergence method in advance).

#### 4.6.6 Acceleration algorithms for self-consistent field calculations

An important feature of the BDF is the acceleration of the energy and gradient calculations of SCF and TDDFT using the **MPEC+COSX** method. Set up the MPEC+COSX calculation with the following inputs.

```
#!/ amylose2.sh
HF/cc-pvdz MPEC+COSX

Geometry
H      -5.27726610038004      0.15767995434597      1.36892178079618
H      -3.89542800401751     -2.74423996083456     -2.30130324998720
H      -3.40930212959730      3.04543096108345      1.73325487719318
O      -4.25161610042910     -0.18429704053319      1.49882079466485
H      -4.12153806480025      0.39113300040060     -0.47267019103680
O      -3.93883902709049     -2.16385597983528     -1.37984323910654
H      -3.65755506365314     -2.55190701717719      0.56784675873394
H      -2.66688104102718     -3.13999999152083     -0.32869523309397
O      -3.68737510690803      2.57255697808269      0.79063986197194
H      -2.16845111442446      1.40439897322928      1.59675986910159
H      -0.80004208156425      3.67692503357694     -0.87083105709857
C      -3.47036908085237      0.21757398797107      0.38361581865084
C      -3.08081604941874     -2.23618399620817     -0.25179522317288
H      -1.85215308213129     -1.05270701067006      0.92020982572454
C      -2.73634509645279      1.50748698767418      0.67208385967460
O      -0.95388209186676      2.93603601652216     -0.08659407523165
H      -2.34176605974133      2.08883703173396     -1.35500112054343
C      -2.46637306624908     -0.89337899823852      0.07760781649778
C      -1.77582007601201      1.83730601785282     -0.45887211416401
O      -1.70216504605578     -0.48600696920536     -1.07005315975028
H      -0.26347504436884      0.90841605388912     -1.67304510231922
C      -0.87599906000257      0.65569503172715     -0.80788211986139
```

(continues on next page)

(continued from previous page)

H	1.05124197574425	-4.08129295376550	-0.80486617677089
H	1.91283792081157	2.93924205088598	-0.71300301703422
O	0.07007992244287	0.29718501862843	0.19143889205868
H	1.28488995808993	-0.48228594245462	-1.27588009910221
O	0.83243796215244	-3.05225096122844	-0.51820416035526
H	0.03099092283770	-2.15700599981123	1.08682384153403
H	0.99725792474852	-3.21082099855794	1.38542783977374
O	1.92550793896406	1.99389906198042	-1.25576903593383
H	2.32288890226196	1.52348902475463	0.72949896259198
H	5.42304993860699	1.71940008598879	-1.13583497057179
C	1.35508593454345	-0.11004196264200	-0.25348109013556
C	0.98581793175676	-2.43946398581436	0.75228585517262
H	1.91238990103301	-0.83125899736406	1.66788890655085
C	2.32240292575108	1.05122704465611	-0.25278704698785
O	4.65571492366175	1.63248206459704	-0.36643098789343
H	3.77658595927138	0.23304608296485	-1.60079803407907
C	1.86060292384221	-1.20698497780059	0.68314589788694
C	3.72997793572998	0.57134806164321	-0.56599702816882
O	3.14827793673614	-1.62888795836893	0.20457391544942
H	5.12279093584136	-0.96659193933436	0.00181296891020
C	4.14403492674986	-0.60389595307832	0.31494395641232
O	4.31314989648861	-0.29843197973243	1.69336596603165
H	3.37540288537848	0.07856300492440	2.10071295465512

**End geometry**

If in advanced input mode, simply add the keyword MPEC+COSX to the COMPASS module input, e.g.

\$compass

**Geometry**

H	-5.27726610038004	0.15767995434597	1.36892178079618
H	-3.89542800401751	-2.74423996083456	-2.30130324998720
H	-3.40930212959730	3.04543096108345	1.73325487719318
O	-4.25161610042910	-0.18429704053319	1.49882079466485
H	-4.12153806480025	0.39113300040060	-0.47267019103680
O	-3.93883902709049	-2.16385597983528	-1.37984323910654
H	-3.65755506365314	-2.55190701717719	0.56784675873394
H	-2.66688104102718	-3.13999999152083	-0.32869523309397
O	-3.68737510690803	2.57255697808269	0.79063986197194
H	-2.16845111442446	1.40439897322928	1.59675986910159
H	-0.80004208156425	3.67692503357694	-0.87083105709857
C	-3.47036908085237	0.21757398797107	0.38361581865084
C	-3.08081604941874	-2.23618399620817	-0.25179522317288
H	-1.85215308213129	-1.05270701067006	0.92020982572454
C	-2.73634509645279	1.50748698767418	0.67208385967460

(continues on next page)

(continued from previous page)

O	-0.95388209186676	2.93603601652216	-0.08659407523165
H	-2.34176605974133	2.08883703173396	-1.35500112054343
C	-2.46637306624908	-0.89337899823852	0.07760781649778
C	-1.77582007601201	1.83730601785282	-0.45887211416401
O	-1.70216504605578	-0.48600696920536	-1.07005315975028
H	-0.26347504436884	0.90841605388912	-1.67304510231922
C	-0.87599906000257	0.65569503172715	-0.80788211986139
H	1.05124197574425	-4.08129295376550	-0.80486617677089
H	1.91283792081157	2.93924205088598	-0.71300301703422
O	0.07007992244287	0.29718501862843	0.19143889205868
H	1.28488995808993	-0.48228594245462	-1.27588009910221
O	0.83243796215244	-3.05225096122844	-0.51820416035526
H	0.03099092283770	-2.15700599981123	1.08682384153403
H	0.99725792474852	-3.21082099855794	1.38542783977374
O	1.92550793896406	1.99389906198042	-1.25576903593383
H	2.32288890226196	1.52348902475463	0.72949896259198
H	5.42304993860699	1.71940008598879	-1.13583497057179
C	1.35508593454345	-0.11004196264200	-0.25348109013556
C	0.98581793175676	-2.43946398581436	0.75228585517262
H	1.91238990103301	-0.83125899736406	1.66788890655085
C	2.32240292575108	1.05122704465611	-0.25278704698785
O	4.65571492366175	1.63248206459704	-0.36643098789343
H	3.77658595927138	0.23304608296485	-1.60079803407907
C	1.86060292384221	-1.20698497780059	0.68314589788694
C	3.72997793572998	0.57134806164321	-0.56599702816882
O	3.14827793673614	-1.62888795836893	0.20457391544942
H	5.12279093584136	-0.96659193933436	0.00181296891020
C	4.14403492674986	-0.60389595307832	0.31494395641232
O	4.31314989648861	-0.29843197973243	1.69336596603165
H	3.37540288537848	0.07856300492440	2.10071295465512

**End geometry**

Basis

cc-pvdz

MPEC+COSX # ask for the MPEC+COSX method

\$end

The SCF module will output a prompt about whether **MPEC+COSX** are both set to True.

```

--- PRINT: Information about SCF Calculation ---
ICTRL_FRAGSCF= 0
IPRIMO= 1
MAXITER= 100
THRENE= 0.10E-07 THRDEN= 0.50E-05
DAMP= 0.00 VSHIFT= 0.00

```

(continues on next page)

(continued from previous page)

```

IFDIIS= T
THRDIIS= 0.10E+01
MINDIIS= 2 MAXDIIS= 8
iCHECK= 0
iAUFBAU= 1
INIGUESS= 0
IfMPEC= T
IfCOSX= T

```

Here, IfMPEC= T and IfCOSX= T indicates that the **MPEC+COSX** method is used to compute. the SCF iterative process is as follows.

```

[scf_cycle_init_ecdenpot]
Meomory for coulpotential          0.02 G

Start SCF iteration.....

Iter.   idiis  vshift      SCF Energy          DeltaE          RMSDeltaD          ↵
↪MaxDeltaD      Damping      Times (S)
  1     0     0.000   -1299.6435521238   -23.7693069405    0.0062252375     0.
↪2842668435     0.0000         2.69
  2     1     0.000   -1290.1030630508     9.5404890730    0.0025508000     0.
↪1065204344     0.0000         1.65
  3     2     0.000   -1290.2258798561    -0.1228168053    0.0014087449     0.
↪0742227520     0.0000         1.67
  4     3     0.000   -1290.4879683983    -0.2620885422    0.0002338141     0.
↪0153879051     0.0000         1.64
  5     4     0.000   -1290.4955210658    -0.0075526675    0.0000713807     0.
↪0049309441     0.0000         1.57
  6     5     0.000   -1290.4966349620    -0.0011138962    0.0000156009     0.
↪0010663736     0.0000         1.51
  7     6     0.000   -1290.4966797420    -0.0000447800    0.0000043032     0.
↪0002765334     0.0000         1.44
  8     7     0.000   -1290.4966810419    -0.0000012999    0.0000014324     0.
↪0000978302     0.0000         1.37
  9     8     0.000   -1290.4966794202     0.0000016217    0.0000003030     0.
↪0000173603     0.0000         1.40
 10     2     0.000   -1290.4966902283    -0.0000108081    0.0000000659     0.
↪0000034730     0.0000         1.11
diis/vshift is closed at iter = 10
 11     0     0.000   -1290.5003691464    -0.0036789181    0.0000225953     0.
↪0009032949     0.0000         5.85

```

(continues on next page)

(continued from previous page)

Label	CPU Time	SYS Time	Wall Time
SCF iteration time:	179.100 S	1.110 S	22.630 S
Final DeltaE = -3.678918126752251E-003			
Final DeltaD = 2.259533940614071E-005 5.000000000000000E-005			
Final scf result			
E_tot =	-1290.50036915		
E_ele =	-3626.68312754		
E_nn =	2336.18275840		
E_1e =	-6428.96436179		
E_ne =	-7717.90756825		
E_kin =	1288.94320647		
E_ee =	2802.28123424		
E_xc =	0.00000000		
Virial Theorem	2.001208		

On a desktop with an i9-9900K CPU, eight OpenMP threads compute in parallel in 22 seconds. The SCF calculation under the same conditions without MPEC+COSX method, the computation takes 110 seconds, and **MPEC+COSX** speeds up the computation by a factor of about 5.

## 4.7 iOI-SCF calculation for large systems and the FLMO method

For large systems (e.g., systems with atomic numbers larger than 300), the traditional SCF calculation methods are often no longer applicable because, in addition to the longer construction time of the Fock matrix at each step The reasons for this are the following factors:

- The Fock matrix diagonalization time increases as a percentage of the total calculation time. When the system is large enough, the construction time of the Fock matrix per step is proportional to the square of the system size. The Fock matrix diagonalization time is proportional to the square of the system size, but the Fock matrix diagonalization time is proportional to the third power of the system size, so for particularly large systems (e.g., thousands of protons), the Fock Therefore, for particularly large systems (e.g., thousands of elements), the Fock matrix diagonalization will account for a significant proportion of the total computation time.
- The greater number of locally stable wave functions in large systems leads to a lower probability of convergence of the SCF calculation to the user's desired state for large systems. In other words, the SCF may converge to many different solutions, only one of which is the user's desired one, thus increasing the probability that the user can determine whether the SCF solution is his or her desired one, and (if it converges) the probability of convergence to the user's desired state. Therefore, it increases the time overhead for the user to determine whether the SCF solution is the desired one and to resubmit the computation (if it converges to a non-desired solution).
- The convergence of the SCF for large systems is more difficult than for small systems, requiring more iterative steps or even failing to converge at all. This is not only because the above the number of locally stable solutions becomes

larger, but also partly because the mass of the general atomic density-based SCF first guess becomes worse as the system increases. The quality of the general atomic density-based SCFs deteriorates as the system increases.

One solution to this problem is to divide the system into segments (a process called binning; these segments can overlap with each other) and do a separate SCF for each segment. BDF's FLMO method (Fragment localized molecular orbital) is a fragmentation-based method, in which the SCF of each fragment is converged and the converged wave function of each fragment is localized. The resulting local orbital is then used to generate a first guess for the total system calculation. This provides some additional benefits over a slice based approach that does not rely on local orbits.

- The SCF iteration can be performed on the local orbital basis, where the Fock matrix does not need to be fully diagonalized, but only block diagonalized, i.e., the orbit is rotated so that the occupied-empty block is zero, a step that is less computationally intensive than the full diagonalization.
- The occupied-empty blocks of the Fock matrix in the local orbital basis are very sparse, and this sparsity can be exploited to further reduce the computational effort of block diagonalization.
- The user can specify the occupation number of a local orbital before the global SCF calculation, and thus selectively obtain the occupied or unoccupied electronic states of that local orbital. For example, to calculate a metal cluster consisting of one Fe(II) and one Fe(III), one can control which Fe converges to the divalent group and which Fe converges to the trivalent group by specifying the occupation number of Fe 3d orbitals. In the current version of BDF another approach is actually supported, i.e. the direct specification of the formal oxidation and spin states of the atoms (see below). For convenience reasons, it is generally recommended that the user specify which electronic state to converge to directly by the formal oxidation and spin states of the atom.
- The SCF calculation yields the converged local orbitals directly, instead of just the regular orbitals as in the normal SCF calculation. If the user needs to obtain convergent local orbitals for wave function analysis, etc., then the FLMO method can save a lot of computational time compared to the traditional approach of obtaining the regular orbitals first and then performing localization, and can also avoid the problem of large systems with many iterations of localization that tend to be non-convergent.

FLMO has been used to obtain the localized orbitals of molecules, iOI-SCF, FLMO-MP2, O(1)-NMR, and other methods, and also to calculate the singlet states of open-shell layers for the study of single-molecule magnets and other problems.

#### 4.7.1 Calculating the Fractionated Domain Molecular Orbital FLMO (Manual Fractionation)

In order to give the user an intuitive understanding of FLMO, we give an example of FLMO calculation. Here, we want to calculate the domains of the 1,3,5,7-octatetraene  $C_8H_{10}$  molecule by the FLMO method. We first calculate 4 molecular slices, each consisting of a central atom, a buffer atom and a linked H-atom. Because of the simple molecular structure, the molecular slices are obtained by manual slicing, i.e., the central atom of each molecular slice is a C=C double bond and all hydrogen atoms attached to it, and the buffer atoms are the C=C double bonds directly linked to the C=C double bond and the hydrogen atoms attached to them, i.e., 1,3-butadiene for slice 1 and slice 4, and 1,3,5-hexatriene for slice 2 and slice 3. After the convergence of the SCF calculation, the molecular slices were used to obtain the molecular slices domain orbitals by the Boys domainization method. After all the molecular slices were calculated, the pFLMO

(primitive Fragment Localized Molecular Orbital) of the whole molecule was synthesized from the four molecular slices. The pFLMO is used as an initial guess to calculate the entire C<sub>8</sub>H<sub>10</sub> molecule and to obtain the localized FLMO. input example is as follows

```
##### Fragment 1
%echo "-----CHECKDATA: Calculate the 1st fragment -----"
$COMPASS
Title
  Fragment 1
Basis
  6-31G
Geometry
c   0.5833330000  0.0  0.0000000000
c   1.9203330000  0.0  0.0000000000
h   0.0250410000  0.0  -0.9477920000
h   0.0250620000  0.0  0.9477570000
h   2.4703130000  0.0  -0.9525920000
c   2.6718330000  0.0  1.3016360000   B
c   4.0088330000  0.0  1.3016360000   B
h   4.7603330000  0.0  2.6032720000   L
h   2.1218540000  0.0  2.2542280000   B
h   4.5588130000  0.0  0.3490440000   B
End geometry
$END

$XUANYUAN
$END

$SCF
RHF
iprtmo
  2
$END

$localmo
FLMO
$end

# copy pFLMO punch file
%cp $BDF_WORKDIR/$BDFTASK.flmo $BDF_TMPDIR/fragment1
%cp $BDF_WORKDIR/$BDFTASK.flmo $BDF_WORKDIR/fragment1

##### Fragment 2
%echo "-----CHECKDATA: Calculate the 2nd fragment -----"
$COMPASS
```

(continues on next page)

(continued from previous page)

```

Title
  Fragment 2
Basis
  6-31G
Geometry
c   0.5833330000  0.0  0.0000000000  B
c   1.9203330000  0.0  0.0000000000  B
h   0.0250410000  0.0 -0.9477920000  L
h   0.0250620000  0.0  0.9477570000  B
h   2.4703130000  0.0 -0.9525920000  B
c   2.6718330000  0.0  1.3016360000
c   4.0088330000  0.0  1.3016360000
h   2.1218540000  0.0  2.2542280000
h   4.5588130000  0.0  0.3490440000
c   4.7603330000  0.0  2.6032720000  B
c   6.0973330000  0.0  2.6032720000  B
h   4.2103540000  0.0  3.5558650000  B
h   6.6473130000  0.0  1.6506800000  B
h   6.8488330000  0.0  3.9049090000  L
End geometry
$END

$XUANYUAN
$END

$SCF
RHF
iprtmo
  2
$END

$localmo
FLMO
$end

# copy pFLMO punch file
%cp $BDF_WORKDIR/$BDFTASK.flmo $BDF_TMPDIR/fragment2
%cp $BDF_WORKDIR/$BDFTASK.flmo $BDF_WORKDIR/fragment2
%ls -l $BDF_TMPDIR
%rm -rf $BDF_TMPDIR/$BDFTASK.*

# Fragment 3
%echo "-----CHECKDATA: Calculate the 3rd fragment -----"

```

(continues on next page)

(continued from previous page)

```
$COMPASS
Title
  Fragment 3
Basis
  6-31G
Geometry
  c  2.6718330000  0.0  1.3016360000  B
  c  4.0088330000  0.0  1.3016360000  B
  h  1.9203330000  0.0  0.0000000000  L
  h  2.1218540000  0.0  2.2542280000  B
  h  4.5588130000  0.0  0.3490440000  B
  c  4.7603330000  0.0  2.6032720000
  c  6.0973330000  0.0  2.6032720000
  h  4.2103540000  0.0  3.5558650000
  h  6.6473130000  0.0  1.6506800000
  c  6.8488330000  0.0  3.9049090000  B
  c  8.1858330000  0.0  3.9049090000  B
  h  6.2988540000  0.0  4.8575010000  B
  h  8.7441260000  0.0  4.8527010000  L
  h  8.7441050000  0.0  2.9571520000  B
End geometry
$END

$XUANYUAN
$END

$SCF
RHF
iprtmo
  2
$END

# flmo_coef_gen=1, iprt=2, ipro=(6,7,8,9), icut=(3,13),
$localmo
FLMO
$end

# copy pFLMO punch file
%cp $BDF_WORKDIR/$BDFTASK.flmo $BDF_TMPDIR/fragment3
%cp $BDF_WORKDIR/$BDFTASK.flmo $BDF_WORKDIR/fragment3
%ls -l $BDF_TMPDIR
%rm -rf $BDF_TMPDIR/$BDFTASK.*
```

(continues on next page)

(continued from previous page)

```

# Fragment 4
%echo "-----CHECKDATA: Calculate the 4th fragment -----"
$COMPASS
Title
  Fragment 4
Basis
  6-31G
Geometry
  h   4.0088330000  0.0  1.3016360000  L
  c   4.7603330000  0.0  2.6032720000  B
  c   6.0973330000  0.0  2.6032720000  B
  h   4.2103540000  0.0  3.5558650000  B
  h   6.6473130000  0.0  1.6506800000  B
  c   6.8488330000  0.0  3.9049090000
  c   8.1858330000  0.0  3.9049090000
  h   6.2988540000  0.0  4.8575010000
  h   8.7441260000  0.0  4.8527010000
  h   8.7441050000  0.0  2.9571520000
End geometry
$END

$XUANYUAN
$END

$SCF
RHF
iprtmo
  2
$END

# flmo_coef_gen=1, iprt=1, ipro=(6,7,8,9,10), icut=(1)
$localmo
FLMO
$end

# copy pFLMO punch file
%cp $BDF_WORKDIR/$BDFTASK.flmo $BDF_TMPDIR/fragment4
%cp $BDF_WORKDIR/$BDFTASK.flmo $BDF_WORKDIR/fragment4
%ls -l $BDF_TMPDIR
%rm -rf $BDF_TMPDIR/$BDFTASK.*

# Whole Molecule calculation
%echo "-----CHECKDATA: From fragment to molecular SCF calculation-----"

```

(continues on next page)

(continued from previous page)

```
$COMPASS
Title
  Whole Molecule calculation
Basis
  6-31G
Geometry
  c  0.5833330000  0.0  0.0000000000
  c  1.9203330000  0.0  0.0000000000
  h  0.0250410000  0.0  -0.9477920000
  h  0.0250620000  0.0  0.9477570000
  h  2.4703130000  0.0  -0.9525920000
  c  2.6718330000  0.0  1.3016360000
  c  4.0088330000  0.0  1.3016360000
  h  2.1218540000  0.0  2.2542280000
  h  4.5588130000  0.0  0.3490440000
  c  4.7603330000  0.0  2.6032720000
  c  6.0973330000  0.0  2.6032720000
  h  4.2103540000  0.0  3.5558650000
  h  6.6473130000  0.0  1.6506800000
  c  6.8488330000  0.0  3.9049090000
  c  8.1858330000  0.0  3.9049090000
  h  6.2988540000  0.0  4.8575010000
  h  8.7441260000  0.0  4.8527010000
  h  8.7441050000  0.0  2.9571520000
End geometry
Nfragment
  4
Group
  C(1)
$END

$XUANYUAN
$END

$SCF
RHF
FLMO
iprtmo
  2
sylv
threshconv
  1.d-8 1.d-6
$END
```

(continues on next page)

(continued from previous page)

```

&DATABASE
fragment 1 9      # Fragment 1 with 9 atoms
 1 2 3 4 5 6 7 8 9 # atom number in the whole molecule
fragment 2 12
 1 2 4 5 6 7 8 9 10 11 12 13
fragment 3 12
 6 7 8 9 10 11 12 13 14 15 16 18
fragment 4 9
 10 11 12 13 14 15 16 17 18
&END

```

In the input, we give the annotations. The calculation of each molecular slice consists of

`$BDFTASK.flmo`, where the domain orbitals are stored, is copied to `$BDF_TMPDIR` by inserting the shell `cp $BDF_WORKDIR/$BDFTASK.flmo $BDF_TMPDIR/fragment*` after the `localmo` module.

After the 4 molecular fragments are calculated, the whole molecule calculation is done, and the input starts from `# Whole Molecule calculation`. In the compass, there is the keyword `Nfragment 4`, which prompts to read in 4 molecule fragments, and the molecule fragment information is defined in the `&DATABASE` field.

The SCF calculation for the whole molecule starts by reading in the four molecular slices of the fixed-domain orbitals, constructing the pFLMO, and giving the orbital stretch factor `Mos` (molecular orbital spread, where a larger `Mos` for a given fixed-domain orbital means that the fixed-domain orbital is more off-domain, and vice versa), as follows.

```

Reading fragment information and mapping orbitals ...

Survived FLMO dims of frag( 11):      8      17      0      46      9
Survived FLMO dims of frag( 15):      8      16      0      66      12
Survived FLMO dims of frag( 15):      8      16      0      66      12
Survived FLMO dims of frag( 11):      8      17      0      46      9
Input Nr. of FLMOs (total, occ., soc., vir.) :  98  32  0  66
  nmo != nbas
           98           92
  Local Occupied Orbitals Mos and Moc
Max_Mos:   1.89136758 Min_Mos:   0.31699600 Aver_Mos:   1.32004368
  Local Virtual Orbitals Mos and Moc
Max_Mos:   2.46745638 Min_Mos:   1.46248295 Aver_Mos:   2.14404812
The prepared Nr. of pFLMOs (total, occ., vir.) :  98  32  0  66

Input Nr. of FLMOs (total, double-occ., single-occ, vir.) :  98  32  0  66
No. double-occ orbitals:           29
No. single-occ orbitals:            0
No. virtual orbitals:              63

```

(continues on next page)

(continued from previous page)

```

iden=      1      29      63      32      66
Transfer dipole integral into Ao basis ...

Transfer quadrupole integral into Ao basis ...

Eliminate the occupied linear-dependent orbitals !
Max_Mos:    1.89136758 Min_Mos:    0.31699600 Aver_Mos:    1.32004368
      3 linear dependent orbitals removed by preliminary scan
Initial MO/AO dimension are :      29      92
  Finally                29 orbitals left. Number of cutted MO    0
Max_Mos:    1.89136758 Min_Mos:    0.31699600 Aver_Mos:    1.29690971
Perform Lowdin orthonormalization to occ pFLMOs
Project pFLMO occupied components out of virtual FLMOs
Max_Mos:    2.46467150 Min_Mos:    1.46222542 Aver_Mos:    2.14111949
      3 linear dependent orbitals removed by preliminary scan
Initial NO, NV, AO dimension are :      29      63      92
  Finally                92 orbitals left. Number of cutted MO    0
Max_Mos:    2.46467150 Min_Mos:    1.46222542 Aver_Mos:    2.15946681
Perform Lowdin orthonormalization to virtual pFLMOs                63
  Local Occupied Orbitals Mos and Moc
Max_Mos:    1.88724854 Min_Mos:    0.31689707 Aver_Mos:    1.29604628
  Local Virtual Orbitals Mos and Moc
Max_Mos:    2.53231018 Min_Mos:    1.46240853 Aver_Mos:    2.16493518
Prepare FLMO time :      0.03 S      0.02 S      0.05 S
Finish FLMO-SCF initial ...

```

It can be seen that the maximum Mos of pFLMO for the whole molecule is less than 2.6, and the pFLMO is fixed-domain regardless of the occupied or imaginary orbitals. The initial guess of the overall molecule is made by using pFLMO, and it enters the SCF iteration, using the block diagonalization method to keep the minimum perturbation of the orbitals, and the output is as follows.

```

Check initial pFLMO orbital MOS
  Local Occupied Orbitals Mos and Moc
Max_Mos:    1.88724854 Min_Mos:    0.31689707 Aver_Mos:    1.29604628
  Local Virtual Orbitals Mos and Moc
Max_Mos:    2.53231018 Min_Mos:    1.46240853 Aver_Mos:    2.16493518
  DNR !!
Final iter :    79 Norm of Febru  0.86590E-06
X --> U time:      0.000      0.000      0.000
block diag      0.017      0.000      0.017
block norm :    2.3273112079137773E-004

  1    0    0.000 -308.562949067 397.366768902  0.002100841  0.027228292  0.0000  0.53

```

(continues on next page)

(continued from previous page)

```

DNR !!
Final iter : 57 Norm of Febru 0.48415E-06
X --> U time: 0.000 0.000 0.017
block diag 0.000 0.000 0.017
block norm : 1.3067586006786384E-004

2 1 0.000 -308.571009930 -0.008060863 0.000263807 0.003230630 0.0000 0.52
DNR !!
Final iter : 43 Norm of Febru 0.64098E-06
X --> U time: 0.000 0.000 0.000
block diag 0.017 0.000 0.017
block norm : 3.6831175797520882E-005

```

After the SCF converges, the system prints the Mos information of molecular orbitals once again.

```

Print pFLMO occupation for checking ...
Occupied alpha orbitals ...
Local Occupied Orbitals Mos and Moc
Max_Mos: 1.91280597 Min_Mos: 0.31692300 Aver_Mos: 1.30442588
Local Virtual Orbitals Mos and Moc
Max_Mos: 2.53288468 Min_Mos: 1.46274299 Aver_Mos: 2.16864691
Write FLMO coef into scratch file ... 214296
Reorder orbital via orbital energy ... 1 1

```

It can be seen that the Mos of the final FLMO does not change much compared with the pFLMO and maintains a good domain fixation.

The above manual slicing method is tedious for molecules with complex structures, because not only the definition of each molecular slice needs to be given manually, but also the correspondence between the atomic number of each slice and the total system needs to be given in the &DATABASE domain. In contrast, a more convenient approach is to use the following automatic slicing method.

## 4.7.2 Calculation of open-shell-layer singlet states using FLMO (automatic binning)

The study of single-molecule magnets, as well as certain catalytic systems, etc., often encounters so-called antiferromagnetic coupled states, which generally consist of two electrons of opposite spin occupying different atomic centers in the form of open-shell layers (open-shell layer singlet states), but may also involve multiple single electrons. BDF can be combined with the FLMO method to calculate open-shell layer singlet states. For example, the following example uses the FLMO method to calculate the spin-broken ground state of a system containing Cu(II) and nitrogen-oxygen stabilized radicals.

```

$autofrag
method

```

(continues on next page)

(continued from previous page)

```
flmo
nprocs
  2 # ask for 2 parallel processes to perform FLMO calculation
spinocc
# Set +1 spin population on atom 9 (O), set -1 spin population on atom 16 (Cu)
  9 +1 16 -1
# Add no buffer atoms, except for those necessary for saturating dangling bonds.
# Minimizing the buffer radius helps keeping the spin centers localized in
# different fragments
radbuff
  0
$end

$compass
Title
  antiferromagnetically coupled nitroxide-Cu complex
Basis
  LANL2DZ
Geometry
C      -0.16158257  -0.34669203   1.16605797
C      0.02573099  -0.67120566  -1.13886544
H      0.90280854  -0.26733412   1.24138440
H     -0.26508467  -1.69387001  -1.01851639
C     -0.81912799   0.50687422   2.26635740
H     -0.52831123   1.52953831   2.14600864
H     -1.88351904   0.42751668   2.19103081
N     -0.38402395   0.02569744   3.58546820
O      0.96884699   0.12656182   3.68120994
C     -1.01167974   0.84046608   4.63575398
H     -0.69497152   0.49022160   5.59592309
H     -0.72086191   1.86312982   4.51540490
H     -2.07607087   0.76110974   4.56042769
N     -0.40937388  -0.19002965  -2.45797639
C     -0.74875417   0.18529223  -3.48688305
Cu     -1.32292113   0.82043400  -5.22772307
F     -1.43762557  -0.29443417  -6.57175160
F     -1.72615042   2.50823941  -5.45404079
H     -0.45239892  -1.36935628   1.28640692
H      1.09012199  -0.59184704  -1.06353906
O     -0.58484750   0.12139125  -0.11715881
End geometry
$end
```

(continues on next page)

(continued from previous page)

```

$xuanyuan
$end

$scf
uks
dft
  PBE0
spinmulti
  1
D3
molden
$end

$localmo
FLMO
Pipek # Pipek-Mezey localization, recommended when pure sigma/pure pi LMOs are needed.
      # Otherwise Boys is better
$end

```

FLMO calculations do not currently support concise input. In this example, the `autofrag` module is used to automatically fragment the molecule and generate the basic input for the FLMO calculation. BDF first generates the molecular fragments based on the molecular structure in the `compass` and the parameter definition information of `autofrag`, as well as the input file for the molecular fragment localization orbital calculation. Then the pFLMO (primitive Fragment Local Molecular Orbital) of the whole molecule is assembled with the domain-fixed orbital of the fragment as the initial guess orbital for the global SCF calculation, and then the open-shell layer singlet state of the whole molecule is obtained by the global SCF calculation while keeping the domain-fixed orbital at each iteration step. In the calculation, the output of the molecular fragment calculation is saved as `_${BDFTASK}.fragmentN.out`, **N** is the fragment number, and the standard output prints only the output of the overall molecular calculation for the sake of output brevity.

The output will give information about the molecular fragmentation that

```

----- Buffered molecular fragments -----
BMolefrag   1:  [[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 19, 20, 21], [], [14],
→ [14, 15], 0.0, 1.4700001016690913]
BMolefrag   2:  [[14, 15, 16, 17, 18], [2, 4, 20], [21], [21], 0.0, 1.
→4700001016690913]
-----
Automatically assigned charges and spin multiplicities of fragments:
-----

```

Fragment	Total No. of atoms	Charge	SpinMult	SpinAlignment
1	17	0	2	Alpha
2	9	0	2	Beta

(continues on next page)

(continued from previous page)

```
-----
Generate BDF input file ....
```

Here it can be seen that we have generated two molecular fragments, specifying that molecular slice **1** consists of 17 atoms with a spin multiplicity of 2, and that molecular slice **2** consists of 9 atoms with a spin multiplicity of 2, but with the opposite spin direction as molecular slice **1**, i.e. one more beta electron than alpha electron, instead of one more alpha electron than beta electron. The 2 molecular slices are then calculated separately, with the following message (assuming the environment variable `OMP_NUM_THREADS` is set to 4):

```
Starting subsystem calculations ...
Number of parallel processes:  2
Number of OpenMP threads per process:  2
Please refer to test117.fragment*.out for detailed output

Total number of not yet converged subsystems:  2
List of not yet converged subsystems:  [1, 2]
Finished calculating subsystem  2 (  1 of  2)
Finished calculating subsystem  1 (  2 of  2)

Starting global calculation ...
```

This care of the computational resource settings. The total computational resources are the product of the number of parallel processes and the number of OpenMP threads per process, where the number of processes is set by the `nprocs` keyword of the `autofrag` module, and the total computational resources are This takes care of the computational resource settings. The total computational resources are the product of the number of parallel pset by the environment variable `OMP_NUM_THREADS`, and the number of threads per process is automatically obtained by dividing the total computational resources by the number of processes.

The computational output of the overall numerator is similar to a normal SCF calculation, but with a chunked diagonalized Fock matrix to keep the orbit definite.

```
Check initial pFLMO orbital MOS
  Openshell alpha :
    Local Occupied Orbitals Mos and Moc
Max_Mos:   1.89684048 Min_Mos:   0.25791767 Aver_Mos:   1.15865182
    Local Virtual Orbitals Mos and Moc
Max_Mos:   8.01038107 Min_Mos:   1.56092594 Aver_Mos:   3.04393282
  Openshell beta  :
    Local Occupied Orbitals Mos and Moc
Max_Mos:   3.00463332 Min_Mos:   0.21757580 Aver_Mos:   1.24636228
    Local Virtual Orbitals Mos and Moc
Max_Mos:   8.00411948 Min_Mos:   1.78248588 Aver_Mos:   3.04672070
```

(continues on next page)

(continued from previous page)

```

...

  1   0   0.000 -849.642342776 1158.171170064 0.046853948 4.840619682 0.5000 3.
↪54
DNR !!
SDNR: warning: rotation angle too large, aborting
Final iter :    5 Norm of Febru 0.20133E+00
X --> U time:      0.000      0.000      0.000
block diag      0.000      0.000      0.000
block norm :    0.290774097871744

DNR !!
Final iter :   359 Norm of Febru 0.82790E-06
X --> U time:      0.000      0.000      0.000
block diag      0.020      0.000      0.010
block norm :    8.589840290871769E-003

```

The orbital stretch (**Mos**) information is given at the beginning of the iteration, the smaller the number, the better the orbital fixity. **Mos** is printed again after the SCF converges. From the results of the Bourget analysis,

```

[Mulliken Population Analysis]
Atomic charges and Spin densities :
  1C      -0.2481    0.0010
  2C      -0.1514    0.0013
  3H       0.2511   -0.0002
  4H       0.2638   -0.0006
  5C      -0.3618   -0.0079
  6H       0.2511    0.0240
  7H       0.2436   -0.0013
  8N       0.0128    0.3100
  9O      -0.2747    0.6562
 10C      -0.5938   -0.0092
 11H       0.2696    0.0040
 12H       0.2414    0.0242
 13H       0.2302   -0.0016
 14N       0.1529   -0.0202
 15C      -0.2730    0.0162
 16Cu      0.8131   -0.5701
 17F      -0.5019   -0.2113
 18F      -0.4992   -0.2143
 19H       0.2207    0.0008
 20H       0.2666   -0.0000
 21O      -0.3128   -0.0008

```

(continues on next page)

(continued from previous page)

```
Sum:    -0.0000    0.0000
```

It can be seen that the spin densities of **-0.5701** for Cu and **0.6562** for 9O are consistent with the pre-specified spins, indicating that the calculations do converge to the desired open-shell singlet state. Note that the absolute value of the spin density here is less than 1, indicating that the spin densities on Cu and 9O are not strictly localized to these two atoms, but are partially off-domain to the next atom.

In the above example, the `autofrag` module input looks complicated, but the `spinocc` and `radbuff` keywords are not necessary for the FLMO method, i.e. the following input file will still work successfully, except that it does not ensure that the spin orientation of Cu and O is the user-specified orientation :

```
$autofrag
method
  flmo
nprocs
  2
$end
```

The `nprocs` indicates the parallelization of the SCF computation for each subsystem, i.e., multiple subsystems are allowed to be computed at the same time. If the `nprocs` keyword is omitted, which is equivalent to setting `nprocs` to 1, the program will compute all subsystems in turn, with each subsystem occupying 8 OpenMP threads and waiting for one subsystem to finish before computing the next one. There is no difference in the result compared to using `nprocs`, but the efficiency of the calculation may be reduced. Therefore, `nprocs` only affects the efficiency of the FLMO computation, not its results, i.e., the following writeup will also run successfully, but the computation time may be slightly longer than writing `nprocs` :

```
$autofrag
method
  flmo
$end
```

Note that setting `nprocs` too large or too small may result in an increase in computation time. For the sake of discussion, assume that the environment variable `OMP_NUM_THREADS` is set to 8 for the FLMO calculation of a larger molecule.

```
nprocs
  4
```

It indicates that :

1. When the program starts subsystem computation, it calls 4 concurrent BDF processes simultaneously, each of which computes one subsystem. If the total number of subsystems  $N$  is less than 4, then only  $N$  concurrent BDF processes are called.
2. Each BDF process uses 2 OpenMP threads. When the total number of subsystems is less than 4, some subsystem

computations may use 3 or 4 OpenMP threads, but the number of concurrent OpenMP threads for the entire computation task is never more than 8.

3. At the beginning of the computation, the entire computation uses exactly 8 OpenMP threads, but as the computation nears its end, when less than 4 subsystems remain to be computed, the number of OpenMP threads used for the entire computation may be less than 8.

The optimal value of `nprocs` is determined by two main factors.

1. Because the parallel efficiency of OpenMP is generally less than 100%, if four tasks of equal duration are run simultaneously, each using two OpenMP threads, the time spent is generally less than if each task is run in turn and each task uses eight OpenMP threads.
2. The computation times for each subsystem are not identical, and may even differ by a factor of several. If some tasks take significantly longer than others, running 4 tasks at the same time with 2 threads per subsystem may be slower than computing each subsystem sequentially with 8 threads, because some computational resources will be idle later in the computation when the 4 subsystems are being computed simultaneously. This is also known as the load balancing problem.

Therefore, if `nprocs` too small or too large, the computation may be less efficient. In general, `nprocs` is set to about 1/5 to 1/3 of the total number of subsystems. If there is no proper estimate of the number of subsystems generated by the system before the calculation, `nprocs` can be simply set to a smaller positive integer such as 1 or 2. The exception is that `nprocs` can be made larger if it is known that the various subsystems of the calculation are computationally similar. `nprocs` For example, in the example at the beginning of this subsection, although there are only two subsystems, the smaller subsystem contains transition metal atoms Cu and the larger subsystem is purely organic, so both subsystems have similar computation times and can be computed simultaneously.

### 4.7.3 iOI-SCF method

The iOI method can be considered as an improvement of the FLMO method. In the FLMO method, even if automatic binning is used, the user still needs to specify the size of the molecular slice with the keywords `radcent` and `radbuff`. Although both keywords have default values (3.0 and 2.0, respectively), neither the default values nor the user-specified values are guaranteed to be optimal for the current system. If the molecular slice is too small, the quality of the obtained fixed-domain orbitals is too poor; if the molecular slice is too large, it leads to too much computation and non-convergence of the fixed-domain iterations. In contrast, the iOI method starts from a relatively small piece of molecule and keeps increasing and fusing the pieces until the piece of molecule reaches the desired size, and then performs the global calculation. Each increase and fusion of molecular slices is called a Macro-iteration. An example is as follows.

```
$autofrag
method
  ioi # To request a conventional FLMO calculation, change ioi to flmo
nprocs
  2 # Use at most 2 parallel processes in calculating the subsystems
```

(continues on next page)

(continued from previous page)

```
$end

$compass
Title
  hydroxychloroquine (diprotonated)
Basis
  6-31G(d)
Geometry # snapshot of GFN2-xTB molecular dynamics at 298 K
C   -4.2028   -1.1506    2.9497
C   -4.1974   -0.4473    4.1642
C   -3.7828    0.9065    4.1812
C   -3.4934    1.5454    2.9369
C   -3.4838    0.8240    1.7363
C   -3.7584   -0.5191    1.7505
H   -4.6123   -0.8793    5.0715
C   -3.3035    3.0061    2.9269
H   -3.1684    1.2214    0.8030
H   -3.7159   -1.1988    0.9297
C   -3.1506    3.6292    4.2183
C   -3.3495    2.9087    5.3473
H   -2.8779    4.6687    4.2878
H   -3.2554    3.3937    6.3124
N   -3.5923    1.5989    5.4076
Cl  -4.6402   -2.7763    3.0362
H   -3.8651    1.0100    6.1859
N   -3.3636    3.6632    1.7847
H   -3.4286    2.9775    1.0366
C   -3.5305    5.2960   -0.0482
H   -2.4848    5.4392   -0.0261
H   -3.5772    4.3876   -0.6303
C   -4.1485    6.5393   -0.7839
H   -3.8803    6.3760   -1.8559
H   -5.2124    6.5750   -0.7031
C   -3.4606    7.7754   -0.2653
H   -2.3720    7.6699   -0.3034
H   -3.7308    7.9469    0.7870
N   -3.8415    8.9938   -1.0424
H   -3.8246    8.8244   -2.0837
C   -2.7415    9.9365   -0.7484
H   -1.7736    9.4887   -0.8943
H   -2.8723   10.2143    0.3196
C   -2.7911   11.2324   -1.6563
H   -1.7773   11.3908   -2.1393
```

(continues on next page)

(continued from previous page)

```

H   -3.5107   10.9108   -2.4646
H   -3.0564   12.0823   -1.1142
C   -5.1510    9.6033   -0.7836
H   -5.5290    9.1358    0.1412
H   -5.0054   10.6820   -0.6847
C   -6.2224    9.3823   -1.8639
H   -6.9636   10.1502   -1.7739
H   -5.8611    9.4210   -2.8855
O   -6.7773    8.0861   -1.6209
H   -7.5145    7.9086   -2.2227
C   -4.0308    4.9184    1.3736
H   -3.7858    5.6522    2.1906
C   -5.5414    4.6280    1.3533
H   -5.8612    3.8081    0.7198
H   -5.9086    4.3451    2.3469
H   -6.1262    5.5024    1.0605

```

**End geometry**

```

MPEC+cosx  # Accelerate the SCF iterations using MPEC+COSX. Not mandatory
$end

```

```

$xuanyuan

```

```

rs # the range separation parameter omega (or mu) of wB97X

```

```

0.3

```

```

$end

```

```

$scf

```

```

rks

```

```

dft

```

```

wB97X

```

```

iprt # Increase print level for more verbose output. Not mandatory

```

```

2

```

```

charge

```

```

2

```

```

$end

```

```

$localmo

```

```

FLMO

```

```

$end

```

Note that in the iOI calculation, the meaning of the `nprocs` keyword is the same as in the FLMO calculation, and the appropriate value needs to be chosen according to the size of the molecule, and the different values of `nprocs` still only affect the calculation speed but not the results. The difference with the FLMO calculation is that the iOI calculation involves multiple macro iterations (see below), and the number of subsystems in each macro iteration is decreasing, so the optimal value of `nprocs` should be conservative, e.g., 1/10-1/5 of the number of subsystems in the macro iteration

at step 0.

At the beginning of the program, the molecule is divided into 5 molecular slices.

```

----- Buffered molecular fragments -----
BMolefrag   1:  [[4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 18, 19], [1, 16, 2, 3, 7, 15, ↵
↵17, 46, 47, 48, 49, 50, 51], [20], [20, 21, 22, 23], 2.0, 2.193]
BMolefrag   2:  [[20, 21, 22, 23, 24, 25, 26, 27, 28, 46, 47, 48, 49, 50, 51], [18,
↵ 19, 29, 30], [8, 31, 38], [8, 4, 11, 31, 32, 33, 34, 38, 39, 40, 41], 2.0, 2.037]
BMolefrag   3:  [[2, 3, 7, 15, 17], [1, 16, 4, 8, 5, 6, 9, 10, 11, 12, 13, 14], ↵
↵[18], [18, 19, 46], 2.0, 3.5]
BMolefrag   4:  [[29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, ↵
↵45], [23, 24, 25, 26, 27, 28, 20, 21, 22], [46], [46, 18, 47, 48], 2.0, 3.386]
BMolefrag   5:  [[1, 16], [2, 3, 7, 5, 6, 9, 10, 4, 8], [15, 11, 18], [15, 12, 17, ↵
↵11, 13, 18, 19, 46], 2.0, 2.12]
-----
Automatically assigned charges and spin multiplicities of fragments:
-----

```

Fragment	Total No. of atoms	Charge	SpinMult	SpinAlignment
1	26	1	1	N.A.
2	22	1	1	N.A.
3	18	1	1	N.A.
4	27	1	1	N.A.
5	14	1	1	N.A.

```

-----

```

Here SpinAlignment is shown as N.A. because all molecular sheets are in closed-shell layer and therefore there is no problem of spin orientation.

After that the subsystem calculation starts.

```

Starting subsystem calculations ...
Number of parallel processes:  2
Number of OpenMP threads per process:  2
Please refer to test106.fragment*.out for detailed output

Macro-iter 0:
Total number of not yet converged subsystems:  5
List of not yet converged subsystems:  [4, 1, 2, 3, 5]
Finished calculating subsystem  4 (  1 of  5)
Finished calculating subsystem  2 (  2 of  5)
Finished calculating subsystem  1 (  3 of  5)
Finished calculating subsystem  5 (  4 of  5)
Finished calculating subsystem  3 (  5 of  5)
Maximum population of LMO tail: 110.00000
=====

```

(continues on next page)

(continued from previous page)

```
Elapsed time of post-processing: 0.10 s
Total elapsed time of this iteration: 34.28 s
```

After that the program fuses these 5 molecular sheets two by two and expands the buffer to obtain 3 larger subsystems. The definition of the 3 larger subsystems is given in `#{BDFTASK}.ioienlarge.out`.

```
Finding the optimum iOI merge plan...
Initial guess merge plan...
Iter 0 Number of permutations done: 1
New center fragments (in terms of old center fragments):
Fragment 1: 5 3
NBas: 164 184
Fragment 2: 2 4
NBas: 164 174
Fragment 3: 1
NBas: 236
Center fragment construction done, total elapsed time 0.01 s
Subsystem construction done, total elapsed time 0.01 s
```

That is, the new subsystem 1 is obtained by fusing (and expanding the buffer) the old subsystems 5 and 3, the new subsystem 2 is obtained by fusing (and expanding the buffer) the old subsystems 2 and 4, and the new subsystem 3 is obtained directly by expanding the buffer of the old subsystem 1. Then the SCF calculation of these larger subsystems is performed using the converged fixed domain orbitals of the original 5 smaller subsystems as the first guess.

```
Macro-iter 1:
Total number of not yet converged subsystems: 3
List of not yet converged subsystems: [2, 3, 1]
Finished calculating subsystem 3 ( 1 of 3)
Finished calculating subsystem 1 ( 2 of 3)
Finished calculating subsystem 2 ( 3 of 3)
Fragment 1 has converged
Fragment 2 has converged
Fragment 3 has converged
Maximum population of LMO tail: 0.04804
=====
*** iOI macro-iteration converged! ***

Elapsed time of post-processing: 0.04 s
Total elapsed time of this iteration: 33.71 s
```

At this point, the program automatically determines that the size of these subsystems is sufficient to converge the LMO of the system to the required accuracy, so the iOI macro iteratively converges and the iOI global computation is performed. iOI global computation produces an output similar to the FLMO global computation, but to further accelerate

the block diagonalization of the Fock matrix, some of the converged LMOs are frozen in the iOI global computation, thus reducing the number of Fock matrices requiring block diagonalization. The dimensionality of the Fock matrix to be diagonalized is reduced, but also introduces a small error (typically of the order of  $10^{-6} \sim 10^{-5}$  Hartree). As an example, take the last step of the SCF iteration.

```

DNR !!
   47 of   90 occupied and   201 of   292 virtual orbitals frozen
SDNR. Preparation:      0.01      0.00      0.00
  norm and abs(maximum value) of Febru  0.35816E-03 0.11420E-03 gap =   1.14531
Survived/total Fia =      472      3913
  norm and abs(maximum value) of Febru  0.36495E-03 0.11420E-03 gap =   1.14531
Survived/total Fia =      443      3913
  norm and abs(maximum value) of Febru  0.16908E-03 0.92361E-04 gap =   1.14531
Survived/total Fia =      615      3913
  norm and abs(maximum value) of Febru  0.11957E-03 0.21708E-04 gap =   1.14531
Survived/total Fia =      824      3913
  norm and abs(maximum value) of Febru  0.68940E-04 0.15155E-04 gap =   1.14531
Survived/total Fia =      965      3913
  norm and abs(maximum value) of Febru  0.56539E-04 0.15506E-04 gap =   1.14531
Survived/total Fia =      737      3913
  norm and abs(maximum value) of Febru  0.30450E-04 0.62094E-05 gap =   1.14531
Survived/total Fia =     1050      3913
  norm and abs(maximum value) of Febru  0.36500E-04 0.82498E-05 gap =   1.14531
Survived/total Fia =      499      3913
  norm and abs(maximum value) of Febru  0.14018E-04 0.38171E-05 gap =   1.14531
Survived/total Fia =     1324      3913
  norm and abs(maximum value) of Febru  0.43467E-04 0.15621E-04 gap =   1.14531
Survived/total Fia =      303      3913
  norm and abs(maximum value) of Febru  0.12151E-04 0.26221E-05 gap =   1.14531
Survived/total Fia =      837      3913
  norm and abs(maximum value) of Febru  0.15880E-04 0.82575E-05 gap =   1.14531
Survived/total Fia =      185      3913
  norm and abs(maximum value) of Febru  0.52265E-05 0.71076E-06 gap =   1.14531
Survived/total Fia =     1407      3913
  norm and abs(maximum value) of Febru  0.31827E-04 0.12985E-04 gap =   1.14531
Survived/total Fia =      253      3913
  norm and abs(maximum value) of Febru  0.77674E-05 0.24860E-05 gap =   1.14531
Survived/total Fia =      650      3913
  norm and abs(maximum value) of Febru  0.56782E-05 0.38053E-05 gap =   1.14531
Survived/total Fia =      264      3913
SDNR. Iter:           0.01      0.00      0.00
Final iter :   16 Norm of Febru  0.25948E-05
X --> U time:       0.000      0.000      0.000
SDNR. XcontrU:       0.00      0.00      0.00
block diag          0.020      0.000      0.000

```

(continues on next page)

(continued from previous page)

```

block norm : 2.321380955939448E-004

Predicted total energy change: -0.0000000659
  9      0      0.000 -1401.6261867529      -0.0011407955      0.0000016329      0.
↪0000904023      0.0000      16.97

```

That is, 47 occupied and 201 imaginary orbits are frozen.

After the convergence of the iOI global computation of SCF, the energy and settling analysis information can be read from the output file following the general SCF computation, which is not repeated here.

## 4.8 Time-dependent Density Generalized Function Theory

The BDF supports a variety of excited state calculation methods, including the linear response time-density generalized function (TDDFT) method based on the Kohn-Sham reference state, and the Tamm-Dancoff approximation (TDA) of the TDDFT method. Compared with other quantization software, the TDDFT module of BDF is unique. The main features are

1. support for various spin-flip (spin-flip) methods.
2. support spin-matching TDDFT method X-TDDFT, which can effectively solve the problem of spin contamination of excited states when the reference state is an open-shell layer, and is suitable for excited state calculation of free radicals, transition metals and other systems.
3. support core excited state (core excited state) related calculations, such as the calculation of the X-ray absorption spectrum (XAS). The iVI method used in BDF can directly calculate all excited states in a higher energy interval without calculating the lower excited states, thus saving computational resources.
4. support the calculation of first-order non-adiabatic coupling matrix element (fo-NACME, or NACME for short), especially between excited states and excited states NACME is mainly used to study non-radiative leap processes, such as using Fermi's golden rule NACME is mainly used to study non-radiative leap processes, such as the calculation of the endoconversion rate constant using Fermi's golden rule, or the study of endoconversion, photochemical processes using non-adiabatic kinetics, etc. Many quantum chemistry programs support NACME between ground and excited states, but fewer programs support NACME between excited and excited states. Therefore, BDF has a unique advantage over most existing quantum chemistry programs for processes such as excited-to-excited state endoconversions and multi-state photochemical reactions.

In addition to TDDFT, BDF also supports the calculation of excited states at the SCF level using the mom 方法.

**Danger:** All general functions of the SCAN family(e.g.,SCAN0, r2SCAN) suffer from the “triplet instability” problem [65], Should not be used for SF-TDDFT calculations(e.g, triple excited states for closed-shell systems). TDA is recommended for this case.

## 4.8.1 Closed-shell system calculations: R-TDDFT

R-TDDFT is used to calculate closed-shell systems. If the ground state calculation starts from the RHF, the TDDFT module performs the TDHF calculation. To calculate the excitation energy of H<sub>2</sub>O molecules using TDDFT, a concise input is given as follows.

```
#!/bdf.sh
TDDFT/B3lyp/cc-pvdz iroot=1

geometry
O
H 1 R1
H 1 R1 2 109.

R1=1.0      # OH bond length in angstrom
end geometry
```

Here, the keyword TDDFT/B3lyp/cc-pvdz specifies that the TDDFT calculation is performed, the generic function used is B3lyp, and the basis group is cc-pVDZ. The corresponding high-level input is

```
$compass
Geometry
O
H 1 1.0
H 1 1.0 2 109.
End geometry
Basis
cc-pvdz
$end

$xuanyuan
$end

$scf
RKS      # Restricted Kohn-sham
DFT      # DFT exchange-correlation functional B3lyp
b3lyp
$end

# input for tddft
$tddft
iroot    # For each irrep, calculate 1 root.
1        #on default, 10 roots are calculated for each irreps if advanced input used
$end
```

The four modules **COMPASS** , **XUANYUAN** , **SCF** and **TDDFT** are called sequentially to complete the computation. The **SCF** module performs the RKS calculation. Based on the results of the RKS calculation, the subsequent **TDDFT** calculation is performed.

Note that since the water molecule belongs to the  $C_{2v}$  point group, there are 4 integrable representations, and the excited states of different integrable representations are solved separately, so there are several ways to specify the number of excited states depending on the user' s requirements, such as

- (1) Calculate 1 excited state for each integrable representation.

```
$TDDFT
iroot
 1
$END
```

At this time, the excited state calculated by each irreducible representation has a high probability of being the excited state with the lowest energy under the irreducible representation, but this cannot be guaranteed, that is to say, there is a small probability that the excited state will converge to the second excited state or even higher. some excited state. If you want to increase the probability of getting the lowest excited state, you can write

```
$TDDFT
iroot
 2
$END
```

At this time, two excited states are calculated for each irreducible representation, and the probability that the first excited state calculated under each irreducible representation is the excited state with the lowest energy under the irreducible representation is higher than when  $iroot=1$ . In addition, at this time, the second excited state calculated under each irreducible representation has a high probability of being the excited state with the second lowest energy under the irreducible representation, but the probability of satisfying this point is higher than that of the first excited state calculated under the irreducible representation. The probability of being the excited state with the lowest energy under this irreducible representation is lower. If  $iroot$  is increased further, the calculated probability that the first excited state is the one with the lowest energy quickly approaches 100%, but never strictly reaches 100%.

For similar reasons, not only is it often necessary to set  $iroot$  greater than 1 when calculating 1 excited state, but when calculating  $N$  ( $N > 1$ ) excited states, if you want to relatively reliably ensure that these  $N$  excited states are the lowest energy The  $N$  excited states of , also need to set  $iroot$  greater than  $N$ . In general,  $iroot$  should be set larger, for example, at least 3 larger than the desired number of excited states, when the molecule satisfies one of the following conditions: (1) the molecule has approximate point group symmetry; (2) the molecule Although it has exact point group symmetry, the calculation is performed at a lower point group due to program limitations or at the user' s request, such as in the open-shell TDDFT (see below) calculation, because the open-shell TDDFT code does not support non- Abelian point group, and the calculation is performed under the largest Abelian subgroup instead. When the molecule does not belong to one of the above cases, the  $iroot$  only needs to be slightly larger than the desired number of excited states, eg 1-2 larger.

- (2) Counting only one B1 excited state and one B2 excited state, and not counting the excited states in the other integrable representations.

```

#! tctest.sh
TDDFT/B3lyp/3-21G nroot=0,0,1,1

Geometry
...
End geometry

```

or

```

$TDDFT
nroot
0 0 1 1 # 也可输入为 0,0,1,1
$END

```

where the `nroot` keyword indicates the number of excited states specified by the user for each integrable representation. Since the program internally arranges the integrable representations of the  $C_{2v}$  point group in the order of A1, A2, B1, and B2 (see the section on the ordering of the integrable representations of the point group), the above input indicates that only one excited state each of B1 and B2 is counted.

- (3) Calculate the lowest 4 excited states without limiting the integrable representations of these excited states

```

#! tctest.sh
TDDFT/B3lyp/3-21G iroot=-4

Geometry
...
End geometry

```

or

```

$TDDFT
iroot
-4
$END

```

In this case, the program uses the initial guessed excitation energy to determine how many excitation states should be solved for each integrable representation, but since the initial guessed order of excitation energy may be different from the fully converged excitation energy, the program cannot strictly guarantee that the 4 excitation states obtained must be the 4 lowest energy states. If the user requires a strict guarantee that the 4 excited states obtained are the lowest 4 excited states, the user should make the program calculate more than 4 excited states, e.g., 8 excited states, and then take the 4 lowest energy states.

The output of the Kohn-Sham calculation has already been described, so here we will only focus on the results of the **TDDFT** calculation. The program output will first give information about the settings of the TDDFT calculation, so that the user can easily check whether the settings are calculated or not, as follows.

```

-----
--- PRINT: Information about TDDFT calculation ---
-----
ERI Maxblk=      8M
[print level]
iprt= 0
[method]
R-TD-DFT
isf= 0
SC Excitations
RPA: (A-B) (A+B) Z=w2*Z
[special choice for method]
ialda= 0
[active space]
Full active space
[algorithm]
Target Excited State in each rep / Diag method :
1  A1      1  1
2  A2      1  1
3  B1      1  1
4  B2      1  1
[dvdson_parameters]
iupdate = 3
Nfac = 50
Nmaxcycle= 50
nblock  = 50
crit_e  = 0.10E-06
crit_vec = 0.10E-04
crit_demo= 0.10E-07
crit_indp= 0.10E-09
guess   = 20
dump    = 0
[output eigenvector control]
cthrrd= 0.100
-----
--- END : Information about TDDFT calculation ---
-----

```

Here,

- R-TD-DFT indicates that the TDDFT based on the restricted basis wave function calculation is being performed.
- isf= 0 indicates that no flipping spin is being computed.
- ialda= 0 indicates that the “Full non-collinear Kernel” is used, which is the default Kernel for the non-spin-flipping TDDFT.

The output below gives the number of roots computed for each non-collinear representation.

```
Target Excited State in each rep / Diag method :
1  A1      1  1
2  A2      1  1
3  B1      1  1
4  B2      1  1
```

The TDDFT module also prints information about occupied orbitals, virtual orbitals, and other active orbitals computed by TDDFT

```
Print [Active] Orbital List
---[Alpha set]---
```

idx	irep	(rep,ibas,type)	F_av (eV)	iact
1	1	A1 1 2	-520.34813	0.05
2	1	A1 2 2	-26.42196	1.84
3	3	B1 1 2	-13.66589	2.96
4	1	A1 3 2	-9.50404	2.49
5	4	B2 1 2	-7.62124	2.12
6	1	A1 4 0	1.23186	9.86
7	3	B1 2 0	3.27539	11.48
8	3	B1 3 0	15.02893	7.40
9	1	A1 5 0	15.44682	6.60
10	1	A1 6 0	24.53525	4.35
11	4	B2 2 0	25.07569	3.88
12	3	B1 4 0	27.07545	6.17
13	2	A2 1 0	33.09515	3.99
14	1	A1 7 0	34.03695	5.08
15	4	B2 3 0	39.36812	4.67
16	3	B1 5 0	43.83066	4.86
17	1	A1 8 0	43.91179	4.34
18	3	B1 6 0	55.56126	4.35
19	1	A1 9 0	56.13188	4.04
20	4	B2 4 0	78.06511	2.06
21	2	A2 2 0	80.16952	2.10
22	1	A1 10 0	83.17934	2.38
23	1	A1 11 0	94.37171	2.81
24	3	B1 7 0	99.90789	2.86

Here, orbitals 1-5 are occupied orbitals and 6-24 are virtual orbitals, where the 5th and 6th orbitals are HOMO and LUMO orbitals, belonging to integrable representation B2 and integrable representation A1 respectively, with orbital energies of -7.62124 eV and 1.23186 eV respectively. Since the H<sub>2</sub>O molecule has 4 irreducible representations, the TDDFT solves for each integrable representation one by one. The system estimates the memory usage before proceeding to the Davidson iteration to solve the Casida equation.

```

=====
Jrep: 1  ExctSym:  A1  (convert to td-psym)
Irep: 1  PairSym:  A1  GsSym:  A1
Nexit:   1    Nsos:   33
=====

Estimated memory for JK operator:           0.053 M
Maxium memory to calculate JK operator:     512.000 M
Allow to calculate  1 roots at one pass for RPA ...
Allow to calculate  2 roots at one pass for TDA ...

Nlarge=           33 Nlimdim=           33 Nfac=           50
Estimated mem for dvdson storage (RPA) =           0.042 M           0.000 G
Estimated mem for dvdson storage (TDA) =           0.017 M           0.000 G

```

Here, the system counts about 0.053 MB of memory needed to store the JK operator, and 512 MB of memory for the input setting (see `memjkop` keyword). The system suggests that the RPA calculation, i.e., the full TDDFT calculation can count 1 root per pass (one pass) and the TDA calculation can count 2 roots per pass. Since the molecular system is small, there is enough memory. For larger molecular systems, if the number of allowed roots per pass output here is less than the system setting, the TDDFT module will construct the JK operator by multiple integration calculations based on the maximum number of allowed roots, resulting in a decrease in computational efficiency and requiring the user to increase memory with the `memjkop` keyword.

Davidson iteration starts computing the output information as follows.

```

Iteration started !

Niter=   1   Nlarge =   33   Nmv =   2
Ndim =   2   Nlimdim=  33   Nres=  31
Approximated Eigenvalue (i,w,diff/eV,diff/a.u.):
  1      9.5246226546      9.5246226546      0.350E+00
No. of converged eigval:   0
Norm of Residuals:
  1      0.0120867135      0.0549049429      0.121E-01      0.549E-01
No. of converged eigvec:   0
Max norm of residues   :  0.549E-01
*** New Directions : sTDDFT-Davidson step ***
Left  Nindp=   1
Right Nindp=   1
Total Nindp=   2
[tddft_dvdson_ZYNI]
Timing For TDDFT_AVmat, Total:           0.08s           0.02s           0.02s
                                MTrans1:           0.00s           0.02s           0.00s
                                COULPOT:           0.00s           0.00s           0.00s
                                AVint  :           0.08s           0.00s           0.02s

```

(continues on next page)

(continued from previous page)

MTrans2:	0.00s	0.00s	0.00s
TDDFT ZYNI-AV time-TOTAL	0.08 S	0.02 S	0.02 S
TDDFT ZYNI-AV time-Coulp	0.08 S	0.02 S	0.02 S
TDDFT ZYNI-AV time-JKcon	0.00 S	0.00 S	0.00 S
tddft JK operator time:	0.00 S	0.00 S	0.00 S
Niter=	2	Nlarge = 33	Nmv = 4
Ndim =	4	Nlimdim= 33	Nres= 29
Approximated Eigenvalue (i,w,diff/eV,diff/a.u.):			
1	9.3817966321	0.1428260225	0.525E-02
No. of converged eigval:	0		
Norm of Residuals:			
1	0.0029082582	0.0074085379	0.291E-02
No. of converged eigvec:	0		0.741E-02

The convergence information is as follows.

Niter=	5	Nlarge = 33	Nmv = 10
Ndim =	10	Nlimdim= 33	Nres= 23
Approximated Eigenvalue (i,w,diff/eV,diff/a.u.):			
1	9.3784431931	0.0000001957	0.719E-08
No. of converged eigval:	1		
### Cong: Eigenvalues have Converged ! ###			
Norm of Residuals:			
1	0.0000009432	0.0000023006	0.943E-06
No. of converged eigvec:	1		0.230E-05
Max norm of residues	:	0.230E-05	
### Cong. Residuals Converged ! ###			
-----			
Orthogonality check2	for	iblock/dim =	0 1
Averaged nHxProd =	10.000		
Ndim =	1	Maximum nonzero deviation	from Iden = 0.333E-15
-----			
Statistics for [dvdson_rpa_block]:			
No. of blocks =	1		
Size of blocks =	50		
No. of eigens =	1		
No. of HxProd =	10	Averaged =	10.000

(continues on next page)

(continued from previous page)

```
Eigenvalues (a.u.) =
  0.3446513056
-----
```

The first line of the above output shows that the computation converges in 5 iterations. The system then prints the information of the converged electronic state.

```
No. 1 w=9.3784 eV -76.0358398606 a.u. f= 0.0767 D<Pab>= 0.0000 Ova= 0.5201
CV(0): A1( 3 )-> A1( 4 ) c_i: 0.9883 Per: 97.7% IPA: 10.736 eV Oai: 0.5163
CV(0): B1( 1 )-> B1( 2 ) c_i: -0.1265 Per: 1.6% IPA: 16.941 eV Oai: 0.6563
Estimate memory in tddft_init mem: 0.001 M
```

The information in the first line is as follows

- No. 1 w= 9.3784 eV means that the first excited state has an excitation energy of 9.3784 eV;
- -76.0358398606 a.u. gives the total energy of the first excited state;
- f= 0.0767 gives the intensity of the oscillator of the jump between the first excited state and the ground state;
- D<Pab>= 0.0000 is the difference between the  $\langle S^2 \rangle$  value of the excited state and the  $\langle S^2 \rangle$  value of the ground state (for spin-conserving jumps, this value reflects the degree of spin contamination of the excited state; for spin-flip jumps, the difference between this value and the theoretical value  $S(S+1)$  (excited state) -  $S(S+1)$  (ground state) reflects the degree of spin contamination of the excited state).
- Ova= 0.5201 is the absolute overlap integral, which takes values in the range [0,1], the closer the value is to 0, the more pronounced the charge transfer characteristics of the corresponding excited state, otherwise the more pronounced the localized excitation characteristics).

Lines 2 and 3 give information on the excited main group states

- CV(0) : where CV(0) means that the excitation is a Core to Virtual orbital excitation and 0 means that it is a Singlet excitation;
- A1( 3 ) -> A1( 4 ) gives the occupied-vacancy orbital pair for the electron leap from the 3rd orbital of A1 to the 4th orbital of A1, which is the HOMO-2 to LUMO excitation when combined with the above output orbital information.
- c\_i: 0.9883 means that the linear combination factor of this jump in the whole excited state is 0.9883;
- Per: 97.7% indicates that this excited state accounts for 97.7% of the total number of excited states.
- IPA: 10.736 eV represents the energy difference of 10.736 eV between the two orbitals involved in the jump.
- Oai: 0.5163 means that if the excited state has only the contribution of this one leap, then the absolute overlap integral of this excited state is 0.5001. From this information, it is convenient to know which leaps are local excitations and which leaps are charge transfer excitations.

After all integrable representations have been solved, all excited states are summarized in order of higher or lower energy output.

No.	Pair	ExSym	ExEnergies	Wavelengths	f	D<S^2>	Dominant
↔Excitations			IPA Ova	En-E1			
1	B2	1 B2	7.1935 eV	172.36 nm	0.0188	0.0000	99.8% CV(0): B2 ( 1
↔)->	A1( 4 )		8.853 0.426	0.0000			
2	A2	1 A2	9.0191 eV	137.47 nm	0.0000	0.0000	99.8% CV(0): B2 ( 1
↔)->	B1( 2 )		10.897 0.356	1.8256			
3	A1	2 A1	9.3784 eV	132.20 nm	0.0767	0.0000	97.7% CV(0): A1 ( 3
↔)->	A1( 4 )		10.736 0.520	2.1850			
4	B1	1 B1	11.2755 eV	109.96 nm	0.0631	0.0000	98.0% CV(0): A1 ( 3
↔)->	B1( 2 )		12.779 0.473	4.0820			

Subsequently, the transition dipole moments also printed.

```
*** Ground to excited state Transition electric dipole moments (Au) ***
```

State	X	Y	Z	Osc.	
1	-0.0000	-0.3266	0.0000	0.0188	0.0188
2	0.0000	0.0000	0.0000	0.0000	0.0000
3	0.0000	0.0000	0.5777	0.0767	0.0767
4	0.4778	-0.0000	0.0000	0.0631	0.0631

## 4.8.2 Calculation of the open-shell layer system: U-TDDFT

The open-shell layer system can be calculated using U-TDDFT, e.g. for  $\text{H}_2\text{O}^+$  ions, the UKS calculation is performed first, and then the U-TDDFT calculation is used to calculate the excited state. A typical input is that

```
#!/bdf.sh
TDDFT/B3lyp/cc-pvdz iroot=4 group=C(1) charge=1

geometry
O
H 1 R1
H 1 R1 2 109.

R1=1.0 # OH bond length in angstrom
end geometry
```

Here, the keyword

- `iroot=4` specifies that 4 roots are calculated for each integrable representation.
- `charge=1` specifies that the charge of the system is +1.
- `group=C(1)` specifies that the C1 point group calculation is forced.

The corresponding high-level input is.

```

$compass
#Notice: The unit of molecular coordinate is angstrom
geometry
O
H 1 R1
H 1 R1 2 109.

R1=1.0 # OH bond length in angstrom
end geometry
basis
cc-pVDZ
group
C(1) # Force to use C1 symmetry
$end

$xuanyuan
$end

$scf
uks
dft
b3lyp
charge
1
spinmulti
2
$end

$tddft
iroot
4
$end

```

A few details to note about this input are

- The `compass` module uses the keyword `group` to force the calculation to use the `C(1)` point group;
- The `scf` module sets the UKS calculation with charge of 1 and `spinmulti` (spin multi,  $2S+1$ ) = 2;
- The `iroot` of the `tddft` module is set to count 4 roots per integrable representation, and since `C1` symmetry is used, the calculation gives the first four excited states of the cation of water.

The U-TDDFT calculation is performed as can be seen from the following output.

```

-----
--- PRINT: Information about TDDFT calculation ---

```

(continues on next page)

(continued from previous page)

```

-----
ERI Maxblk=      8M
[print level]
iprt= 0
[method]
U-TD-DFT
isf= 0
SC Excitations
RPA: (A-B) (A+B) Z=w2*Z

```

The calculation summarizes the output of the 4 excited states as

No.	Pair	ExSym	ExEnergies	Wavelengths	f	D<S^2>	Dominant
↪Excitations			IPA Ova	En-E1			
1	A 2	A	2.1960 eV	564.60 nm	0.0009	0.0024	99.4% CO (bb) : A (
↪ 4	)↪	A( 5 )	5.955 0.626	0.0000			
2	A 3	A	6.3479 eV	195.31 nm	0.0000	0.0030	99.3% CO (bb) : A (
↪ 3	)↪	A( 5 )	9.983 0.578	4.1520			
3	A 4	A	12.0991 eV	102.47 nm	0.0028	1.9312	65.8% CV (bb) : A (
↪ 4	)↪	A( 6 )	14.637 0.493	9.9032			
4	A 5	A	13.3618 eV	92.79 nm	0.0174	0.0004	97.6% CV (aa) : A (
↪ 4	)↪	A( 6 )	15.624 0.419	11.1659			

The third excited state has a large D<S^2> value, indicating a spin contamination problem.

### 4.8.3 Open-shell system: X-TDDFT (also called SA-TDDFT)

X-TDDFT is a spin-matched TDDFT method for calculating the open-shell layer system. The open-shell system U-TDDFT triplet state coupled to a double-occupied to imaginary orbital excited state (labeled CV(1) in BDF) suffers from spin contamination and thus its excitation energy is often underestimated. X-TDDFT can be used to solve this problem. Considering  $N_2^+$  molecules, the concise computational inputs to X-TDDFT are

```

#! N2+.sh
X-TDDFT/b3lyp/aug-cc-pvtz group=D(2h) charge=1 spinmulti=2 iroot=5

Geometry
N 0.00 0.00 0.00
N 0.00 0.00 1.1164
End geometry

```

Advanced input

```

$compass
#Notice: The unit of molecular coordinate is angstrom
Geometry
N 0.00 0.00 0.00
N 0.00 0.00 1.1164
End geometry
basis
  aug-cc-pvtz
group
  D(2h) # Force to use D2h symmetry
$end

$xuanyuan
$end

$scf
roks # ask for ROKS calculation
dft
  b3lyp
charge
  1
spinmulti
  2
$end

$tdfft
iroot
  5
$end

```

Here, the **SCF** module requires the ROKS method to calculate the ground state, and the **TDDFT** module will default to the **X-TDDFT** calculation.

The excited state output is.

No.	Pair	ExSym	ExEnergies	Wavelengths	f	D<S^2>	Dominant	
↔Excitations			IPA Ova	En-E1				
1	B2u	1 B2u	0.7902 eV	1569.00 nm	0.0017	0.0195	98.6%	CO(0) : B2u( <span style="color:red">↔</span>
↔ 1	)→	Ag( 3 )	3.812 0.605	0.0000				
2	B3u	1 B3u	0.7902 eV	1569.00 nm	0.0017	0.0195	98.6%	CO(0) : B3u( <span style="color:red">↔</span>
↔ 1	)→	Ag( 3 )	3.812 0.605	0.0000				
3	B1u	1 B1u	3.2165 eV	385.46 nm	0.0378	0.3137	82.6%	CO(0) : B1u( <span style="color:red">↔</span>
↔ 2	)→	Ag( 3 )	5.487 0.897	2.4263				
4	B1u	2 B1u	8.2479 eV	150.32 nm	0.0008	0.9514	48.9%	CV(1) : B2u( <span style="color:red">↔</span>
↔ 1	)→	B3g( 1 )	12.415 0.903	7.4577				(continues on next page)

(continued from previous page)

5	Au	1	Au	8.9450 eV	138.61 nm	0.0000	1.2618	49.1%	CV(0) : B2u(	↔
↔ 1	)→	B2g(	1)	12.903 0.574	8.1548					
6	Au	2	Au	9.0519 eV	136.97 nm	0.0000	1.7806	40.1%	CV(1) : B3u(	↔
↔ 1	)→	B3g(	1)	12.415 0.573	8.2617					
7	B1u	3	B1u	9.0519 eV	136.97 nm	0.0000	1.7806	40.1%	CV(1) : B3u(	↔
↔ 1	)→	B2g(	1)	12.415 0.906	8.2617					
8	B2g	1	B2g	9.4442 eV	131.28 nm	0.0000	0.0061	99.0%	OV(0) : Ag(	↔
↔ 3	)→	B2g(	1)	12.174 0.683	8.6540					
9	B3g	1	B3g	9.4442 eV	131.28 nm	0.0000	0.0061	99.0%	OV(0) : Ag(	↔
↔ 3	)→	B3g(	1)	12.174 0.683	8.6540					
10	Au	3	Au	9.5281 eV	130.12 nm	0.0000	0.1268	37.0%	CV(0) : B3u(	↔
↔ 1	)→	B3g(	1)	12.903 0.574	8.7379					
11	B1u	4	B1u	9.5281 eV	130.12 nm	0.0000	0.1267	37.0%	CV(0) : B2u(	↔
↔ 1	)→	B3g(	1)	12.903 0.909	8.7379					
12	Au	4	Au	10.7557 eV	115.27 nm	0.0000	0.7378	49.1%	CV(1) : B3u(	↔
↔ 1	)→	B3g(	1)	12.415 0.575	9.9655					
13	B3u	2	B3u	12.4087 eV	99.92 nm	0.0983	0.1371	70.4%	CV(0) : B1u(	↔
↔ 2	)→	B2g(	1)	15.288 0.793	11.6185					
14	B2u	2	B2u	12.4087 eV	99.92 nm	0.0983	0.1371	70.4%	CV(0) : B1u(	↔
↔ 2	)→	B3g(	1)	15.288 0.793	11.6185					
15	B1u	5	B1u	15.9005 eV	77.98 nm	0.7766	0.7768	32.1%	CV(0) : B3u(	↔
↔ 1	)→	B2g(	1)	12.903 0.742	15.1103					
16	B2u	3	B2u	17.6494 eV	70.25 nm	0.1101	0.4841	92.0%	CV(0) : B2u(	↔
↔ 1	)→	Ag(	4)	19.343 0.343	16.8592					
17	B3u	3	B3u	17.6494 eV	70.25 nm	0.1101	0.4841	92.0%	CV(0) : B3u(	↔
↔ 1	)→	Ag(	4)	19.343 0.343	16.8592					
18	Ag	2	Ag	18.2820 eV	67.82 nm	0.0000	0.0132	85.2%	OV(0) : Ag(	↔
↔ 3	)→	Ag(	4)	19.677 0.382	17.4918					
19	B2u	4	B2u	18.5465 eV	66.85 nm	0.0021	1.5661	77.8%	CV(1) : B2u(	↔
↔ 1	)→	Ag(	4)	19.825 0.401	17.7562					
20	B3u	4	B3u	18.5465 eV	66.85 nm	0.0021	1.5661	77.8%	CV(1) : B3u(	↔
↔ 1	)→	Ag(	4)	19.825 0.401	17.7562					
21	Ag	3	Ag	18.7805 eV	66.02 nm	0.0000	0.2156	40.4%	CV(0) : B3u(	↔
↔ 1	)→	B3u(	2)	20.243 0.337	17.9903					
22	B1g	1	B1g	18.7892 eV	65.99 nm	0.0000	0.2191	40.5%	CV(0) : B2u(	↔
↔ 1	)→	B3u(	2)	20.243 0.213	17.9990					
23	B1g	2	B1g	18.8704 eV	65.70 nm	0.0000	0.2625	41.8%	CV(0) : B3u(	↔
↔ 1	)→	B2u(	2)	20.243 0.213	18.0802					
24	B3g	2	B3g	18.9955 eV	65.27 nm	0.0000	0.2673	83.4%	CV(0) : B2u(	↔
↔ 1	)→	B1u(	3)	20.290 0.230	18.2053					
25	B2g	2	B2g	18.9955 eV	65.27 nm	0.0000	0.2673	83.4%	CV(0) : B3u(	↔
↔ 1	)→	B1u(	3)	20.290 0.230	18.2053					
26	B3u	5	B3u	19.0339 eV	65.14 nm	0.0168	1.6012	66.7%	CV(1) : B1u(	↔
↔ 2	)→	B2g(	1)	20.612 0.715	18.2437					

(continues on next page)

(continued from previous page)

27	B2u	5	B2u	19.0339 eV	65.14 nm	0.0168	1.6012	66.7%	CV(1): B1u(	↔
↔ 2	)→	B3g(	1)	20.612 0.715	18.2437					
28	Ag	4	Ag	19.0387 eV	65.12 nm	0.0000	0.0693	35.9%	CO(0): Ag(	↔
↔ 2	)→	Ag(	3)	21.933 0.437	18.2484					
29	Ag	5	Ag	19.3341 eV	64.13 nm	0.0000	0.1694	44.7%	CO(0): Ag(	↔
↔ 2	)→	Ag(	3)	21.933 0.457	18.5439					
30	Ag	6	Ag	19.8685 eV	62.40 nm	0.0000	1.7807	40.4%	CV(1): B3u(	↔
↔ 1	)→	B3u(	2)	21.084 0.338	19.0783					
31	B1g	3	B1g	19.8695 eV	62.40 nm	0.0000	1.7774	40.5%	CV(1): B2u(	↔
↔ 1	)→	B3u(	2)	21.084 0.213	19.0792					
32	B3g	3	B3g	19.9858 eV	62.04 nm	0.0000	1.6935	80.7%	CV(1): B2u(	↔
↔ 1	)→	B1u(	3)	21.038 0.231	19.1956					
33	B2g	3	B2g	19.9858 eV	62.04 nm	0.0000	1.6935	80.7%	CV(1): B3u(	↔
↔ 1	)→	B1u(	3)	21.038 0.231	19.1956					
34	B1g	4	B1g	19.9988 eV	62.00 nm	0.0000	1.7373	41.8%	CV(1): B3u(	↔
↔ 1	)→	B2u(	2)	21.084 0.213	19.2086					
35	B2g	4	B2g	20.2417 eV	61.25 nm	0.0000	0.2901	81.4%	CV(0): B1u(	↔
↔ 2	)→	B3u(	2)	22.628 0.228	19.4515					
36	B3g	4	B3g	20.2417 eV	61.25 nm	0.0000	0.2901	81.4%	CV(0): B1u(	↔
↔ 2	)→	B2u(	2)	22.628 0.228	19.4515					
37	Au	5	Au	21.2302 eV	58.40 nm	0.0000	0.2173	40.4%	CV(0): B2u(	↔
↔ 1	)→	B2g(	2)	22.471 0.157	20.4400					
38	B2g	5	B2g	22.1001 eV	56.10 nm	0.0000	0.0031	99.2%	OV(0): Ag(	↔
↔ 3	)→	B2g(	2)	23.220 0.204	21.3099					
39	B3g	5	B3g	22.1001 eV	56.10 nm	0.0000	0.0031	99.2%	OV(0): Ag(	↔
↔ 3	)→	B3g(	2)	23.220 0.204	21.3099					
40	B1g	5	B1g	23.4663 eV	52.84 nm	0.0000	0.0027	99.8%	OV(0): Ag(	↔
↔ 3	)→	B1g(	1)	25.135 0.283	22.6761					

Here, the 4th, 6th and 7th excited states are CV(1) states. Note that the  $D\langle S^2 \rangle$  values calculated by SA-TDDFT are based on the U-TDDFT formula, which gives an approximation of the spin contamination of the results if these states were calculated by U-TDDFT, but does not represent the actual spin contamination of these states, since SA-TDDFT guarantees that all excited states are strictly free of spin contamination. Therefore, a large value of  $D\langle S^2 \rangle$  for a state calculated by SA-TDDFT does not indicate that the result for that state is unreliable, but rather indicates that SA-TDDFT is an improvement over U-TDDFT for that state.

## 4.8.4 Calculation of the triplet excited state using the closed-shell singlet state as the reference state

Starting from the ground state of the closed-shell layer of the H<sub>2</sub>O molecule, the triplet excited state can be calculated. The simple input is.

```
#!/ bdf.sh
tdft/b3lyp/cc-pvdz iroot=4 spinflip=1

geometry
O
H 1 R1
H 1 R1 2 109.

R1=1.0      # OH bond length in angstrom
end geometry
```

Note that although the keyword here is named spinflip, this calculation is not a spin-flip TDDFT calculation, since it calculates the  $M_S = 0$  component of the triplet excited state instead of the  $M_S = 1$  component. The corresponding high-level inputs are

```
$compass
#Notice: Coordinate unit is angstrom
geometry
O
H 1 R1
H 1 R1 2 109.

R1=1.0      # OH bond length in angstrom
end geometry
basis
  cc-pvdz
group
  C(1) # Force to use C1 symmetry
$end

$xuanyuan
$end

$scf
rks      # ask for RKS calculation
dft
  b3lyp
$end
```

(continues on next page)

(continued from previous page)

```

$tdfft
isf      # ask for triplet TDDFT calculation
1
iroot
4
$end

```

When the TDDFT calculation is almost finished, there is an output message as follows.

```

*** List of excitations ***

Ground-state spatial symmetry:  A
Ground-state spin: Si=  0.0000

Spin change: isf=  1
D<S^2>_pure=  2.0000 for excited state (Sf=Si+1)
D<S^2>_pure=  0.0000 for excited state (Sf=Si)

Imaginary/complex excitation energies :  0 states
Reversed sign excitation energies :  0 states

No. Pair  ExSym  ExEnergies  Wavelengths  f  D<S^2>  Dominant
↔Excitations      IPA  Ova  En-E1

```

No.	Pair	ExSym	ExEnergies	Wavelengths	f	D<S^2>	Dominant
1	A 1 A	6.4131 eV	193.33 nm	0.0000	2.0000	99.2%	CV(1): A( 5)
↔)	-> A( 6 )	8.853 0.426	0.0000				
2	A 2 A	8.2309 eV	150.63 nm	0.0000	2.0000	97.7%	CV(1): A( 4)
↔)	-> A( 6 )	10.736 0.519	1.8177				
3	A 3 A	8.4793 eV	146.22 nm	0.0000	2.0000	98.9%	CV(1): A( 5)
↔)	-> A( 7 )	10.897 0.357	2.0661				
4	A 4 A	10.1315 eV	122.37 nm	0.0000	2.0000	92.8%	CV(1): A( 4)
↔)	-> A( 7 )	12.779 0.479	3.7184				

```

*** Ground to excited state Transition electric dipole moments (Au) ***
State      X      Y      Z      Osc.
1          0.0000  0.0000  0.0000  0.0000  0.0000
2          0.0000  0.0000  0.0000  0.0000  0.0000
3          0.0000  0.0000  0.0000  0.0000  0.0000
4          0.0000  0.0000  0.0000  0.0000  0.0000

```

where Spin change: isf= 1 suggests that the calculation is for a state with a spin multiplet 2 larger than the ground state (i.e., a triplet state), and since the ground state is a single heavy state and the ground to excited state jump is spin-barred, the oscillator strength and jump dipole moment are both 0.

TDDFT only calculates excited states with the same spin as the reference state by default. For example, the ground state of an H<sub>2</sub>O molecule is a single heavy state, and the TDDFT value calculates the single heavy excited state; to calculate both the single heavy state and the triplet state, the input is

```
#!/ H2OTDDFT.sh
TDDFT/b3lyp/cc-pVDZ iroot=4 spinflip=0,1

geometry
O
H 1 0.9
H 1 0.9 2 109.0
end geometry
```

The system will run TDDFT twice to calculate the single heavy state and triplet state respectively, where the output of the single heavy state is

No.	Pair	ExSym	ExEnergies	Wavelengths	f	D<S^2>	Dominant
↔Excitations			IPA Ova	En-E1			
1	B2	1 B2	8.0968 eV	153.13 nm	0.0292	0.0000 99.9%	CV(0): B2 (
↔1	)→	A1( 4 )	9.705 0.415	0.0000			
2	A2	1 A2	9.9625 eV	124.45 nm	0.0000	0.0000 99.9%	CV(0): B2 (
↔1	)→	B1( 2 )	11.745 0.329	1.8656			
3	A1	2 A1	10.1059 eV	122.69 nm	0.0711	0.0000 99.1%	CV(0): A1 (
↔3	)→	A1( 4 )	11.578 0.442	2.0090			
4	B1	1 B1	12.0826 eV	102.61 nm	0.0421	0.0000 99.5%	CV(0): A1 (
↔3	)→	B1( 2 )	13.618 0.392	3.9857			
5	B1	2 B1	15.1845 eV	81.65 nm	0.2475	0.0000 99.5%	CV(0): B1 (
↔1	)→	A1( 4 )	16.602 0.519	7.0877			
6	A1	3 A1	17.9209 eV	69.18 nm	0.0843	0.0000 95.4%	CV(0): B1 (
↔1	)→	B1( 2 )	18.643 0.585	9.8240			
7	A2	2 A2	22.3252 eV	55.54 nm	0.0000	0.0000 99.8%	CV(0): B2 (
↔1	)→	B1( 3 )	24.716 0.418	14.2284			
...							

The output of the triplet state is

No.	Pair	ExSym	ExEnergies	Wavelengths	f	D<S^2>	Dominant
↔Excitations			IPA Ova	En-E1			
1	B2	1 B2	7.4183 eV	167.13 nm	0.0000	2.0000 99.4%	CV(1): B2 (
↔1	)→	A1( 4 )	9.705 0.415	0.0000			
2	A1	1 A1	9.3311 eV	132.87 nm	0.0000	2.0000 98.9%	CV(1): A1 (
↔3	)→	A1( 4 )	11.578 0.441	1.9128			
3	A2	1 A2	9.5545 eV	129.76 nm	0.0000	2.0000 99.2%	CV(1): B2 (
↔1	)→	B1( 2 )	11.745 0.330	2.1363			

(continues on next page)

(continued from previous page)

4	B1	1	B1	11.3278 eV	109.45 nm	0.0000	2.0000	97.5%	CV(1): A1 (	↵
↵3	)	->	B1 ( 2 )	13.618 0.395	3.9095					
5	B1	2	B1	14.0894 eV	88.00 nm	0.0000	2.0000	97.8%	CV(1): B1 (	↵
↵1	)	->	A1 ( 4 )	16.602 0.520	6.6711					
6	A1	2	A1	15.8648 eV	78.15 nm	0.0000	2.0000	96.8%	CV(1): B1 (	↵
↵1	)	->	B1 ( 2 )	18.643 0.582	8.4465					
7	A2	2	A2	21.8438 eV	56.76 nm	0.0000	2.0000	99.5%	CV(1): B2 (	↵
↵1	)	->	B1 ( 3 )	24.716 0.418	14.4255					
...										

Since the single to triplet state jump is dipole-barred, the oscillator strength  $f=0.0000$ .

## 4.8.5 Spin-flip TDDFT calculation

The BDF can calculate the triplet state not only from a single heavy state, but also from a **2S+1** heavy state with higher spin multiplicity ( $S = 1/2, 1, 3/2, \dots$ ), and upward spin-flip the **2S+3** heavy state; the spin-up **TDDFT/TDA** gives the alpha electron to unoccupied beta orbital lepton state of the double-occupied orbital, labeled as CV(1) excitation. Unlike the case where the ground state is a closed-shell single heavy state, the BDF calculation at this point is for the  $M_S = S + 1$  component of the **2S + 3** heavy state, so when the ground state is not a closed-shell single heavy state, the calculation can be called a spin-flip TDDFT calculation. The input file format for the spin-up TDDFT calculation is exactly the same as when the ground state is a closed-shell single heavy state and the triplet excited state is calculated, e.g., the following input file calculates the quadruplet excited state with the duplex state as the reference state.

```

...
$scf
UKS
...
spinmulti
2
$end

$tddft
...
isf
1
$end

```

In addition, the BDF can start from a triplet state and flip the spin down to calculate a single heavy state, which requires setting `isf` to `-1`. Of course, it is also possible to flip down from a state with a higher spin multiplicity to calculate a state with a spin multiplicity of 2 less. Note that the spin-down **TDDFT/TDA** can only correctly describe electronic states that leap from the alpha orbital occupied by the open-shell layer to the beta orbital occupied by the open-shell layer, labeled as **OO(ab)** leap, while all other leap types of states have spin contamination problems.

Starting from the triplet state and reversing the spin downward to calculate the singlet state, the input is

```

#! H2OTDDFT.sh
TDA/b3lyp/cc-pVDZ spinmulti=3 iroot=-4 spinflip=-1

geometry
O
H 1 0.9
H 1 0.9 2 109.0
end geometry

```

The output is

```

Imaginary/complex excitation energies : 0 states

No. Pair  ExSym  ExEnergies      Wavelengths      f      D<S^2>      Dominant_
↪Excitations      IPA  Ova      En-E1

  1  A   1  A   -8.6059 eV      -144.07 nm      0.0000  -1.9933  99.3% OO(ab) :  A( ↪
↪ 6 )->  A(  5 )  -6.123 0.408  0.0000
  2  A   2  A   -0.0311 eV     -39809.08 nm      0.0000  -0.0034  54.1% OO(ab) :  A( ↪
↪ 5 )->  A(  5 )   7.331 1.000   8.5747
  3  A   3  A    0.5166 eV     2399.85 nm      0.0000  -1.9935  54.0% OO(ab) :  A( ↪
↪ 6 )->  A(  6 )   2.712 0.999   9.1225
  4  A   4  A    2.3121 eV     536.24 nm      0.0000  -0.9994  99.9% OV(ab) :  A( ↪
↪ 6 )->  A(  7 )   4.671 0.872  10.9180

```

Here, the first three states are **OO(ab)** type excited states, where the first and the third states are basically pure singlet states ( $D$  is approximately equal to -2, i.e., the excited state is approximately equal to 0), and the second state is basically pure triplet state ( $D$  is approximately equal to 0); the fourth state is an **OV(ab)** type excited state with spin contamination problem ( $D$  is approximately equal to -1, i.e., the excited state is approximately equal to 1, between singlet and triplet states), and its excitation energy is not reliable.

**Warning:**

- BDF currently only supports spin-flipped TDA, but not spin-flipped TDDFT, but the calculation of triplet excited states with closed-shell singlet states as reference states is not subject to this restriction.

## 4.8.6 Calculation of UV-Vis and XAS spectra by iVI method

The above examples are based on the TDDFT excited states solved by Davidson's method. In order to solve for an excited state with the Davidson method, it is generally necessary to solve for all excited states with lower energies than it. Therefore, when the energy of the target excited state is very high (e.g., when calculating the XAS spectrum), the Davidson method requires too many computational resources to obtain a result with limited computation time and memory. In addition, when using the Davidson method, the user must specify the number of excited states to be solved before the calculation, however, many times the user does not know the number of excited states he needs before the calculation, but only the approximate energy range of the excited states he needs, etc. This makes the user have to go through a series of trial and error, first set a small number of excited states for the calculation, and if he finds that If you find that you do not have the state you need, you can increase the number of excited states and recalculate until you find the state you need. Obviously, this will consume the user's energy and time for no reason.

BDF's iVI method provides a solution to this problem. In the iVI method, the user can specify the excitation energy range of interest (e.g., the entire visible region, or the K-edge region of carbon) without having to estimate how many excited states are in that range; the program can calculate all excited states in that range, without having to calculate excited states at energies lower than that range as in the Davidson method, on the one hand, and ensure that all excited states in that energy range are obtained On the other hand, it ensures that all excited states in the energy range are obtained without missing any. Two examples of calculations are given below.

(1) Calculation of the absorption spectrum of the DDQ radical anion in the 400-700 nm range (X-TDDFT, wB97X/LANL2DZ)

```

$COMPASS
Title
  DDQ radical anion TDDFT
Basis
  LANL2DZ
Geometry # UB3LYP/def2-SVP geometry
C          0.00000000    2.81252550   -0.25536084
C          0.00000000    1.32952185   -2.58630187
C          0.00000000   -1.32952185   -2.58630187
C          0.00000000   -2.81252550   -0.25536084
C          0.00000000   -1.29206304    2.09336443
C          -0.00000000    1.29206304    2.09336443
Cl         0.00000000   -3.02272954    4.89063172
Cl        -0.00000000    3.02272954    4.89063172
C          0.00000000   -2.72722649   -4.89578100
C         -0.00000000    2.72722649   -4.89578100
N          0.00000000   -3.86127688   -6.78015122
N         -0.00000000    3.86127688   -6.78015122
O          0.00000000   -5.15052650   -0.22779097
O         -0.00000000    5.15052650   -0.22779097
End geometry

```

(continues on next page)

(continued from previous page)

```
units
  bohr
mpec+cosx # accelerate the calculation using MPEC+COSX
$end

$XUANYUAN
rs
  0.3 # rs for wB97X
$END

$SCF
roks
dft
  wB97X
charge
  -1
$END

$tdft
iprt # print level
  2
itda
  0
idiag # selects the iVI method
  3
iwindow
  400 700 nm # alternatively the unit can be given as au, eV or cm-1 instead of nm.
              # default is in eV if no unit is given
itest
  1
incorrect
  1
memjkop
  2048
$end
```

Since the molecule belongs to the  $C_{2v}$  point group, there are four integrable representations (A1, A2, B1, B2), and the program solves the TDDFT problem under each of the four integrable representations. Take A1 integrable representation as an example, after convergence of iVI iterations, the program outputs the following information.

```
Root 0, E= 0.1060649560, residual= 0.0002136455
Root 1, E= 0.1827715245, residual= 0.0005375061
Root 2, E= 0.1863919913, residual= 0.0006792424
```

(continues on next page)

(continued from previous page)

```

Root 3, E= 0.2039707800, residual= 0.0008796108
Root 4, E= 0.2188244775, residual= 0.0015619745
Root 5, E= 0.2299349293, residual= 0.0010684879
Root 6, E= 0.2388141752, residual= 0.0618579646
Root 7, E= 0.2609321083, residual= 0.0695001907
Root 8, E= 0.2649984329, residual= 0.0759920121
Root 9, E= 0.2657352154, residual= 0.0548521587
Root 10, E= 0.2743644891, residual= 0.0655238098
Root 11, E= 0.2766959875, residual= 0.0600950472
Root 12, E= 0.2803090818, residual= 0.0587604503
Root 13, E= 0.2958382984, residual= 0.0715968457
Root 14, E= 0.3002756135, residual= 0.0607394762
Root 15, E= 0.3069930238, residual= 0.0720773993
Root 16, E= 0.3099721369, residual= 0.0956453409
Root 17, E= 0.3141986951, residual= 0.0688103843
Excitation energies of roots within the energy window (au):
0.1060649560
Timing Spin analyze :          0.01          0.00          0.00

No.      1      w=      2.8862 eV      -594.3472248862 a.u.      f= 0.0000      D<Pab>= 0.0717  _
↔ Ova= 0.5262
      CO(bb):   A1( 20 )->  A2(  4 )   c_i: -0.9623   Per: 92.6%   IPA:    8.586 eV  _
↔ Oai: 0.5360
      CV(bb):   A1( 20 )->  A2(  5 )   c_i: -0.1121   Per:  1.3%   IPA:   11.748 eV  _
↔ Oai: 0.3581
      CV(bb):   B1( 18 )->  B2(  6 )   c_i:  0.2040   Per:  4.2%   IPA:   13.866 eV  _
↔ Oai: 0.4328

```

It can be seen that the program computes 17 excited states under this integrable representation, but only one of them (excitation energy 0.106 au = 2.89 eV) is within the user-specified wavelength interval (400-700 nm), and thus converges completely (in the form of small residuals); the rest of the excited states are known to be outside the user-interest range far before they converge, so the program does not try to converge anymore. The remaining excited states are known to be out of the user's range of interest well before they converge, and no further attempts are made to converge them (as indicated by the large residuals), thus saving much computational effort.

After all four integrable representations have been calculated, the program summarizes the results of each integrable representation as usual.

No.	Pair	ExSym	ExEnergies	Wavelengths	f	D<S^2>	Dominant	
↔Excitations			IPA	Ova	En-E1			
1	A1	2	A2	2.4184 eV	512.66 nm	0.1339	0.0280	93.0% OV(aa): A2( 4
↔)->			A2( 5 )	7.064 0.781	0.0000			
2	B2	1	B1	2.7725 eV	447.19 nm	0.0000	0.0000	92.5% CO(bb): B1( 18
↔)->			A2( 4 )	8.394 0.543	0.3541			

(continues on next page)

(continued from previous page)

```

  3  A2    1  A1    2.8862 eV   429.58 nm   0.0000   0.0000   92.6% CO(bb) :  A1 ( 20
↔)-> A2 (   4 )    8.586 0.526   0.4677
  4  B1    1  B2    3.0126 eV   411.55 nm   0.0000   0.0000   63.5% CO(bb) :  B2 (  4
↔)-> A2 (   4 )    8.195 0.820   0.5942

```

## (2) Calculation of carbon K-edge XAS spectra of ethylene (sf-X2C, M06-2X/uncontracted def2-TZVP)

```

$COMPASS
Title
  iVI test
Basis
  def2-TZVP
geometry
  C -5.77123022 1.49913343 0.00000000
  H -5.23806647 0.57142851 0.00000000
  H -6.84123022 1.49913343 0.00000000
  C -5.09595591 2.67411072 0.00000000
  H -5.62911966 3.60181564 0.00000000
  H -4.02595591 2.67411072 0.00000000
End geometry
group
  c(1)
uncontract # uncontract the basis set (beneficial for the accuracy of core
↔excitations)
$END

$XUANYUAN
heff
  3 # selects sf-X2C
$END

$SCF
rks
dft
  m062x
$END

$TDDFT
imethod
  1 # R-TDDFT
idiag
  3 # iVI
iwindow
  275 285 # default unit: eV

```

(continues on next page)

(continued from previous page)

Send

The K-edge absorption of carbon is experimentally known to be around 280 eV, so the energy range here is chosen to be 275-285 eV. 15 excited states are calculated in this energy interval.

No.	Pair	ExSym	ExEnergies	Wavelengths	f	D<S^2>	Dominant
↔Excitations	IPA	Ova	En-E1				
1	A 2	A	277.1304 eV	4.47 nm	0.0018	0.0000	97.1% CV(0): A( 5
↔)	-> A( 93 )		281.033 0.650	0.0000			
2	A 3	A	277.1998 eV	4.47 nm	0.0002	0.0000	96.0% CV(0): A( 6
↔)	-> A( 94 )		282.498 0.541	0.0694			
3	A 4	A	277.9273 eV	4.46 nm	0.0045	0.0000	92.8% CV(0): A( 7
↔)	-> A( 94 )		281.169 0.701	0.7969			
4	A 5	A	278.2593 eV	4.46 nm	0.0000	0.0000	100.0% CV(0): A( 8
↔)	-> A( 95 )		283.154 0.250	1.1289			
5	A 6	A	279.2552 eV	4.44 nm	0.0002	0.0000	85.5% CV(0): A( 4
↔)	-> A( 93 )		284.265 0.627	2.1247			
6	A 7	A	280.0107 eV	4.43 nm	0.0000	0.0000	96.6% CV(0): A( 8
↔)	-> A( 96 )		284.941 0.315	2.8803			
7	A 8	A	280.5671 eV	4.42 nm	0.0000	0.0000	97.0% CV(0): A( 5
↔)	-> A( 94 )		284.433 0.642	3.4366			
8	A 9	A	280.8642 eV	4.41 nm	0.1133	0.0000	93.3% CV(0): A( 2
↔)	-> A( 9 )		287.856 0.179	3.7337			
9	A 10	A	280.8973 eV	4.41 nm	0.0000	0.0000	90.1% CV(0): A( 1
↔)	-> A( 9 )		287.884 0.185	3.7668			
10	A 11	A	281.0807 eV	4.41 nm	0.0000	0.0000	66.8% CV(0): A( 6
↔)	-> A( 95 )		287.143 0.564	3.9502			
11	A 12	A	282.6241 eV	4.39 nm	0.0000	0.0000	97.7% CV(0): A( 7
↔)	-> A( 95 )		285.815 0.709	5.4937			
12	A 13	A	283.7528 eV	4.37 nm	0.0000	0.0000	65.1% CV(0): A( 4
↔)	-> A( 94 )		287.666 0.592	6.6223			
13	A 14	A	283.9776 eV	4.37 nm	0.0000	0.0000	92.1% CV(0): A( 6
↔)	-> A( 96 )		288.929 0.523	6.8471			
14	A 15	A	284.1224 eV	4.36 nm	0.0008	0.0000	98.2% CV(0): A( 7
↔)	-> A( 96 )		287.601 0.707	6.9920			
15	A 16	A	284.4174 eV	4.36 nm	0.0000	0.0000	93.7% CV(0): A( 3
↔)	-> A( 93 )		289.434 0.509	7.2869			

However, it can be seen from the composition of the excited states that only the two excited states with excitation energies of 280.8642 eV and 280.8973 eV are excitations from the C 1s to the valence orbitals, while the rest of the excitations are excitations from the valence orbitals to the very high Rydberg orbitals, i.e., background absorption corresponding to valence electron ionization.

## 4.8.7 Plotting of absorption spectra with Gaussian widening

The above calculations yield only the excitation energies and oscillator intensities for each excited state, while the user often needs to obtain the theoretically predicted peak shape of the absorption spectrum, which requires a Gaussian widening of the absorption of each excited state by a certain half-peak width. In BDF, this is done with the Python script `plotspec.py` (located under `$BDFHOME/sbin/`, where `$BDFHOME` is the BDF installation path). For example, suppose we have calculated the TDDFT excited state of the C60 molecule using BDF, and the corresponding output file is `C60.out`, then we can run

```
$BDFHOME/sbin/plotspec.py C60.out
```

or

```
$BDFHOME/sbin/plotspec.py C60
```

The script will output the following on the screen.

```
BDF output file: C60.out
1 TDDFT output block(s) found
Block 1: 10 excited state(s)
- Singlet absorption spectrum, spin-allowed
plotspec.py: exit successfully
```

and produces two files, one for `C60.stick.csv`, containing the absorption wavelengths and molar extinction coefficients for all excited states, which can be used to make stick graphs.

```
TDDFT Singlets 1,,
Wavelength,Extinction coefficient,
nm,L/(mol cm),
342.867139,2899.779319,
307.302300,31192.802393,
237.635960,131840.430395,
211.765024,295.895849,
209.090150,134.498113,
197.019205,179194.526059,
178.561512,145.257962,
176.943322,54837.570677,
164.778366,548.752301,
160.167663,780.089056,
```

The other is `C60.spec.csv`, which contains the absorption spectrum after Gaussian broadening (the default broadening FWHM is 0.5 eV).

```
TDDFT Singlets 1,,
Wavelength,Extinction coefficient,
```

(continues on next page)

(continued from previous page)

```

nm, L/(mol cm),
200.000000, 162720.545118,
201.000000, 151036.824457,
202.000000, 137429.257570,
...
998.000000, 0.000000,
999.000000, 0.000000,
1000.000000, 0.000000,

```

These two files can be opened and plotted with Excel, Origin, and other plotting software.

The command line parameters can be used to control the plotting range, the Gaussian broadening FWHM, etc.

Example.

```

# Plot the spectrum in the range 300-600 nm:
$BDFHOME/sbin/plotspec.py wavelength=300-600nm filename.out

# Plot an X-ray absorption spectrum in the range 200-210 eV,
# using an FWHM of 1 eV:
$BDFHOME/sbin/plotspec.py energy=200-210eV fwhm=1eV filename.out

# Plot a UV-Vis spectrum in the range 10000 cm-1 to 40000 cm-1,
# where the wavenumber is sampled at an interval of 50 cm-1:
$BDFHOME/sbin/plotspec.py wavenumber=10000-40000cm-1 interval=50 filename.out

# Plot an emission spectrum in the range 600-1200 nm, as would be
# given by Kasha's rule (i.e. only the first excited state is considered),
# where the wavelength is sampled at an interval of 5 nm:
$BDFHOME/sbin/plotspec.py -emi wavelength=600-1200nm interval=5 filename.out

```

If you run `$BDFHOME/sbin/plotspec.py` without command line arguments, you can list all command line arguments and their usage, which will not be repeated here.

## 4.8.8 Excited State Structure Optimization

BDF supports not only the calculation of TDDFT single point energies (i.e., excitation energies for a given molecular structure), but also structure optimization of excited states, numerical frequencies, etc. For this purpose, the `$tddft` module is required. For this purpose, a `$resp` module is added after the `$tddft` module to calculate the gradient of the TDDFT energy, and a `$bdfopt` module is added after the `$compass` module to use the TDDFT gradient information for structure optimization and frequency calculation (see [结构优化与频率计算](#) for details).

The following is an example of calculations to optimize the structure of the first excited state of butadiene at the B3LYP/cc-pVDZ level.

```
$COMPASS
Title
  C4H6
Basis
  CC-PVDZ
Geometry # Coordinates in Angstrom. The structure has C(2h) symmetry
  C          -1.85874726   -0.13257980   0.00000000
  H          -1.95342119   -1.19838319   0.00000000
  H          -2.73563916    0.48057645   0.00000000
  C          -0.63203020    0.44338226   0.00000000
  H          -0.53735627    1.50918564   0.00000000
  C          0.63203020   -0.44338226   0.00000000
  H          0.53735627   -1.50918564   0.00000000
  C          1.85874726    0.13257980   0.00000000
  H          1.95342119    1.19838319   0.00000000
  H          2.73563916   -0.48057645   0.00000000
End Geometry
$END

$BDFOPT
solver
  1
$END

$XUANYUAN
$END

$SCF
RKS
dft
  B3lyp
$END

$TDDFT
nroot
# The ordering of irreps of the C(2h) group is: Ag, Au, Bg, Bu
# Thus the following line specifies the calculation of the 1Bu state, which
# happens to be the first excited state for this particular molecule.
  0 0 0 1
istore
  1
# TDDFT gradient requires tighter TDDFT convergence criteria than single-point
# TDDFT calculations, thus we tighten the convergence criteria below.
crit_vec
```

(continues on next page)

(continued from previous page)

```

1.d-6 # default 1.d-5
crit_e
1.d-8 # default 1.d-7
$END

$resp
geom
norder
1 # first-order nuclear derivative
method
2 # TDDFT response properties
nfiles
1 # must be the same number as the number after the istore keyword in $TDDFT
iroot
1 # calculate the gradient of the first root. Can be omitted here since only
  # one root is calculated in the $TDDFT block
$end

```

Note that in the above example, the meaning of the keyword `iroot` in the `$resp` module is different from the meaning of the keyword `iroot` in the `$tddft` module above. The former refers to calculating the gradient of the first excited state, and the latter refers to how many excited states are calculated for each irreducible representation.

After convergence of the structure optimization, the converged structure is output in the main output file.

```

Good Job, Geometry Optimization converged in      5 iterations!

Molecular Cartesian Coordinates (X,Y,Z) in Angstrom :
  C      -1.92180514      0.07448476      0.00000000
  H      -2.21141426     -0.98128927      0.00000000
  H      -2.70870517      0.83126705      0.00000000
  C      -0.54269837      0.45145649      0.00000000
  H      -0.31040658      1.52367715      0.00000000
  C       0.54269837     -0.45145649      0.00000000
  H       0.31040658     -1.52367715      0.00000000
  C       1.92180514     -0.07448476      0.00000000
  H       2.21141426      0.98128927      0.00000000
  H       2.70870517     -0.83126705      0.00000000

          Force-RMS   Force-Max   Step-RMS   Step-Max
Conv. tolerance :  0.2000E-03  0.3000E-03  0.8000E-03  0.1200E-02
Current values  :  0.5550E-04  0.1545E-03  0.3473E-03  0.1127E-02
Geom. converge  :      Yes          Yes          Yes          Yes

```

In addition, the excitation energy in the excited state equilibrium structure can be read from the output of the last TDDFT module in the `.out.tmp` file, as well as the total energy of the excited state and the main components.

```

No.      1      w=      5.1695 eV      -155.6874121542 a.u.      f= 0.6576      D<Pab>= 0.0000
-> Ova= 0.8744
      CV(0):   Ag(  6 )-> Bu( 10 )   c_i:  0.1224   Per:  1.5%   IPA:   17.551 eV
-> Oai: 0.6168
      CV(0):   Bg(  1 )-> Au(  2 )   c_i: -0.9479   Per: 89.9%   IPA:    4.574 eV
-> Oai: 0.9035

...

No. Pair  ExSym  ExEnergies  Wavelengths      f      D<S^2>      Dominant
->Excitations          IPA  Ova      En-E1
      1 Bu    1 Bu    5.1695 eV    239.84 nm    0.6576    0.0000    89.9% CV(0): Bg(
->1 )-> Au(  2 )    4.574 0.874    0.0000

```

The wavelength (240 nm) corresponding to the excitation energy in the excited state equilibrium structure is the fluorescence emission wavelength of butadiene.

#### 4.8.9 Spin-orbit coupling calculation based on sf-X2C-TDDFT-SOC

Relativistic effects include scalar relativity and spin-orbit coupling (SOC). The relativistic calculations require the use of **basis sets optimized for relativistic effects and the selection of a suitable Hamiltonian**. sf-X2C-TDDFT-SOC calculations are supported by BDF for all-electron sf-X2C, where sf-X2C refers to the scalar relativistic effects considered with spinless exact two-component (eXact Two-Component, X2C) Hamiltonians, and TDDFT-SOC refers to the spin-orbit coupling calculations based on TDDFT. TDDFT calculation of spin-orbit coupling. Note that although TDDFT is an excited state method, TDDFT-SOC can be used to calculate not only the contribution of SOC to the excited state energy and properties, but also the contribution of SOC to the ground state energy and properties.

Taking a molecule with a single heavy ground state as an example, the TDDFT calculation module needs to be called three times in sequence to complete the sf-X2C-TDDFT-SOC calculation. Among them, the first execution utilizes R-TDDFT and calculates the single heavy state, the second uses SF-TDDFT to calculate the triplet state, and the last reads in the wave functions of the first two TDDFT calculations and calculates the spin-orbit coupling of these states using the state interaction (SI) method. This can be clearly seen from the advanced input of the sf-X2C-TDDFT-SOC calculation for the CH<sub>2</sub>S molecule below.

```

$COMPASS
Title
  ch2s
Basis # Notice: we use relativistic basis set contracted by DKH2
  cc-pVDZ-DK
Geometry
C      0.000000    0.000000   -1.039839
S      0.000000    0.000000    0.593284

```

(continues on next page)

(continued from previous page)

```
H      0.000000    0.932612   -1.626759
H      0.000000   -0.932612   -1.626759
End geometry
$END

$xuanyuan
heff  # ask for sf-X2C Hamiltonian
  3
hsoc  # set SOC integral as 1e+mf-2e
  2
$end

$scf
RKS
dft
  PBE0
$end

#1st: R-TDDFT, calculate singlets
$tdft
isf
  0
idiag
  1
iroot
  10
itda
  0
istore # save TDDFT wave function in the 1st scratch file
  1
$end

#2nd: spin-flip tddft, use close-shell determinant as reference to calculate triplets
$tdft
isf # notice here: ask for spin-flip up calculation
  1
itda
  0
idiag
  1
iroot
  10
istore # save TDDFT wave function in the 2nd scratch file, must be specified
```

(continues on next page)

```
2
$end

#3rd: tddft-soc calculation
$tddft
isoc
2
nprt # print level
10
nfiles
2
ifgs # whether to include the ground state in the SOC treatment. 0=no, 1=yes
1
imatsoc
8
0 0 0 2 1 1
0 0 0 2 2 1
0 0 0 2 3 1
0 0 0 2 4 1
1 1 1 2 1 1
1 1 1 2 2 1
1 1 1 2 3 1
1 1 1 2 4 1
imatrso
6
1 1
1 2
1 3
1 4
1 5
1 6
idiag # full diagonalization of SO Hamiltonian
2
$end
```

**Warning:**

- The calculations must be performed in the order `isf=0, isf=1`. When SOC processing does not consider the ground state (i.e., `ifgs=0`), the more excited states `iroot`, the more accurate the result; when considering the ground state (i.e., `ifgs=1`), too many `iroot` will reduce the accuracy, which is reflected in the underestimation of the ground state energy, and there is no fixed rule for the selection of `iroot`.

The keyword `imatsoc` controls which SOC matrix elements  $\langle A|H_{so}|B\rangle$  are to be printed.

- `- 8` means that the SOC matrix elements between 8 sets of spin states are to be printed, and an array of 8 integer rows is entered sequentially below.
- The input format for each line is `fileA symA stateA fileB symB stateB`, representing the matrix element  $\langle \text{fileA}, \text{symA}, \text{stateA} | H_{so} | \text{fileB}, \text{symB}, \text{stateB} \rangle$ , where
- `fileA symA stateA` represents the `stateA` root of the integrable representation of `symA` in `fileA`; for example, `1 1 1` represents the first root of the integrable representation of the first TDDFT calculation.
- `0 0 0 0` indicates the ground state

The printout of the coupling matrix element is as follows.

```
[tddft_soc_matsoc]

Print selected matrix elements of [Hsoc]

SocPairNo. = 1   SOCmat = < 0 0 0 |Hso| 2 1 1 >   Dim = 1 3
  mi/mj      ReHso (au)      cm^-1      ImHso (au)      cm^-1
0.0 -1.0     0.0000000000     0.0000000000     0.0000000000     0.0000000000
0.0  0.0     0.0000000000     0.0000000000     0.0000000000     0.0000000000
0.0  1.0     0.0000000000     0.0000000000     0.0000000000     0.0000000000

SocPairNo. = 2   SOCmat = < 0 0 0 |Hso| 2 2 1 >   Dim = 1 3
  mi/mj      ReHso (au)      cm^-1      ImHso (au)      cm^-1
0.0 -1.0     0.0000000000     0.0000000000     0.0000000000     0.0000000000
0.0  0.0     0.0000000000     0.0000000000     0.0007155424     157.0434003237
0.0  1.0     0.0000000000     0.0000000000     -0.0000000000     -0.0000000000

SocPairNo. = 3   SOCmat = < 0 0 0 |Hso| 2 3 1 >   Dim = 1 3
  mi/mj      ReHso (au)      cm^-1      ImHso (au)      cm^-1
0.0 -1.0     -0.0003065905     -67.2888361761     0.0000000000     0.0000000000
0.0  0.0     0.0000000000     0.0000000000     -0.0000000000     -0.0000000000
0.0  1.0     -0.0003065905     -67.2888361761     -0.0000000000     -0.0000000000
```

Here,  $\langle 0 0 0 | H_{so} | 2 2 1 \rangle$  denotes the matrix element  $\langle S_0 | H_{so} | T_1 \rangle$ , which gives its real part `ReHso` and imaginary part `ImHso`, respectively. Since  $S_0$  has only one component,  $m_i$  is 1.  $T_1$  (spin  $S=1$ ) has 3 components ( $M_s=-1,0,1$ ), and the 3 components are numbered by  $m_j$ . The imaginary part of the coupling matrix element between the component with  $M_s=0$  and the ground state is `0.0007155424 au`.

**Warning:** When comparing the results of different programs, note that the so-called spherical tensor is given here instead of cartesian tensor, i.e.,  $T_1$  is  $T_{-1}, T_0, T_1$ , not  $T_x, T_y, T_z$ , and there are you-transformations between them.

The SOC calculation results in that

Total No. of States:		161	Print:	10			
No.	1	w=	-0.0006 eV				
		Spin:  Gs,1>	1-th Spatial: A1;	OmegaSF=	0.0000eV	Cr= 0.0000	Ci= 0.
	↔9999	Per:100.0%					
		SumPer: 100.0%					
No.	2	w=	1.5481 eV				
		Spin:  S+,1>	1-th Spatial: A2;	OmegaSF=	1.5485eV	Cr= 0.9998	Ci= -0.
	↔0000	Per:100.0%					
		SumPer: 100.0%					
No.	3	w=	1.5482 eV				
		Spin:  S+,3>	1-th Spatial: A2;	OmegaSF=	1.5485eV	Cr= 0.9998	Ci= 0.
	↔0000	Per:100.0%					
		SumPer: 100.0%					
No.	4	w=	1.5486 eV				
		Spin:  S+,2>	1-th Spatial: A2;	OmegaSF=	1.5485eV	Cr= 0.9999	Ci= 0.
	↔0000	Per:100.0%					
		SumPer: 100.0%					
No.	5	w=	2.2106 eV				
		Spin:  So,1>	1-th Spatial: A2;	OmegaSF=	2.2117eV	Cr= -0.9985	Ci= 0.
	↔0000	Per: 99.7%					
		SumPer: 99.7%					
No.	6	w=	2.5233 eV				
		Spin:  S+,1>	1-th Spatial: A1;	OmegaSF=	2.5232eV	Cr= 0.9998	Ci= 0.
	↔0000	Per:100.0%					
		SumPer: 100.0%					
No.	7	w=	2.5234 eV				
		Spin:  S+,3>	1-th Spatial: A1;	OmegaSF=	2.5232eV	Cr= 0.9998	Ci= -0.
	↔0000	Per:100.0%					
		SumPer: 100.0%					
No.	8	w=	2.5240 eV				
		Spin:  S+,2>	1-th Spatial: A1;	OmegaSF=	2.5232eV	Cr= 0.0000	Ci= -0.
	↔9985	Per: 99.7%					
		SumPer: 99.7%					
No.	9	w=	5.5113 eV				

(continues on next page)

(continued from previous page)

```

Spin: |S+,1> 1-th Spatial: B2; OmegaSF= 5.5115eV Cr= -0.7070 Ci= -0.
↪0000 Per: 50.0%
Spin: |S+,3> 1-th Spatial: B2; OmegaSF= 5.5115eV Cr= 0.7070 Ci= 0.
↪0000 Per: 50.0%
SumPer: 100.0%

No. 10 w= 5.5116 eV
Spin: |S+,1> 1-th Spatial: B2; OmegaSF= 5.5115eV Cr= -0.5011 Ci= -0.
↪0063 Per: 25.1%
Spin: |S+,2> 1-th Spatial: B2; OmegaSF= 5.5115eV Cr= 0.7055 Ci= 0.
↪0000 Per: 49.8%
Spin: |S+,3> 1-th Spatial: B2; OmegaSF= 5.5115eV Cr= -0.5011 Ci= -0.
↪0063 Per: 25.1%
SumPer: 100.0%

*** List of SOC-SI results ***

No. ExEnergies Dominant Excitations Esf dE ̄
↪Eex (eV) (cm^-1)

1 -0.0006 eV 100.0% Spin: |Gs,1> 0-th A1 0.0000 -0.0006 0.
↪0000 0.00
2 1.5481 eV 100.0% Spin: |S+,1> 1-th A2 1.5485 -0.0004 1.
↪5487 12491.27
3 1.5482 eV 100.0% Spin: |S+,3> 1-th A2 1.5485 -0.0004 1.
↪5487 12491.38
4 1.5486 eV 100.0% Spin: |S+,2> 1-th A2 1.5485 0.0001 1.
↪5492 12494.98
5 2.2106 eV 99.7% Spin: |So,1> 1-th A2 2.2117 -0.0011 2.
↪2112 17834.44
6 2.5233 eV 100.0% Spin: |S+,1> 1-th A1 2.5232 0.0002 2.
↪5239 20356.82
7 2.5234 eV 100.0% Spin: |S+,3> 1-th A1 2.5232 0.0002 2.
↪5239 20356.99
8 2.5240 eV 99.7% Spin: |S+,2> 1-th A1 2.5232 0.0008 2.
↪5246 20362.08
9 5.5113 eV 50.0% Spin: |S+,1> 1-th B2 5.5115 -0.0002 5.
↪5119 44456.48
10 5.5116 eV 49.8% Spin: |S+,2> 1-th B2 5.5115 0.0001 5.
↪5122 44458.63

```

Here the output has two parts, the first part gives the energy and composition of each SOC-SI state relative to the S0 state, for example

- No. 10 w= 5.5116 eV means that the energy of the 10th SOC-SI state is 5.5116 eV, note that this is

the energy relative to the S0 state;

The following three lines show the components of this state.

- Spin:  $|S+,1\rangle$  1-th Spatial: B2; represents the first triplet state with symmetry B2 (spin +1 with respect to the S state, and therefore S+); and and therefore S+);
- $\Omega_{SF} = 5.5115\text{eV}$  is the energy relative to the first spin state.
- $C_r = -0.5011$   $C_i = -0.0063$  is the real and imaginary part of the wave function of this component in the spinor state, with a percentage of 25.1%.

The second part summarizes the results of the SOC-SI state calculations.

- $E_{x\text{Energies}}$  are the excitation energies when SOC is taken into account, and  $E_{sf}$  is the original excitation energy without SOC;
- The excited states are denoted by Spin:  $|S,M\rangle$  n-th sym, and spin  $|G_s,l\rangle$ , the nth state with spatial symmetry sym. For example, the  $|G_s,l\rangle$  represents the ground state,  $|S_o,l\rangle$  represents the excited state with the same total spin as the ground state, and  $|S+,2\rangle$  represents the excited state with total spin plus 1. M is the first component of the spin projection (in total  $2S+1$ ).

The keyword **imatrso** specifies which sets of jump dipole moments between the spin states are

- 1 1 indicates the intrinsic dipole moment of the ground state.
- 1 2 indicates the dipole moments between the first and second spin states.

The output of the leap dipole moments is as follows.

```
[tddft_soc_matrso]: Print selected matrix elements of [dpl]
```

No.	( I , J )	$ r_{ij} ^2$	$E_J-E_I$	fosc	rate(s <sup>-1</sup> )
1	1 1	0.472E+00	0.000000000	0.000000000	0.000E+00
Details of transition dipole moment <b>with</b> SOC (in a.u.):					
		$\langle I X J\rangle$	$\langle I Y J\rangle$	$\langle I Z J\rangle$	(also in debye)
	Real=	-0.113E-15	-0.828E-18	0.687E+00	-0.0000 -0.0000 1.7471
	Imag=	-0.203E-35	0.948E-35	0.737E-35	-0.0000 0.0000 0.0000
	Norm=	0.113E-15	0.828E-18	0.687E+00	
No.	( I , J )	$ r_{ij} ^2$	$E_J-E_I$	fosc	rate(s <sup>-1</sup> )
2	1 2	0.249E-05	1.548720567	0.000000095	0.985E+01
Details of transition dipole moment <b>with</b> SOC (in a.u.):					
		$\langle I X J\rangle$	$\langle I Y J\rangle$	$\langle I Z J\rangle$	(also in debye)
	Real=	-0.589E-03	0.207E-07	-0.177E-15	-0.0015 0.0000 -0.0000

(continues on next page)

(continued from previous page)

Imag=	-0.835E-08	0.147E-02	-0.198E-16	-0.0000	0.0037	-0.0000
Norm=	0.589E-03	0.147E-02	0.178E-15			

**Hint:**

- `imatSOC` set to `-1` specifies printing of all coupling matrix elements;
- The default is not to print the leap dipole moments, set `imatRSO` to `-1` to print the leap dipole moments between all spin states, and `imatRSO` to `-2` to print the leap dipole moments between all ground state spin states and all excited state spin states.
- The reference state for SOC calculations must be either RHF/RKS or ROHF/ROKS, UHF/UKS is not supported.
- When the reference state for SOC calculations is ROHF/ROKS, TDDFT calculations with `isf=0` must use X-TDA (i.e., `itest=1`, `icorrect=1`, `isf=0`, `itda=1`; full X-TDDFT is not supported) and TDDFT calculations with `isf=1` must use SF-TDA (i.e. `isf=1`, `itda=1`; full SF-TDDFT is not supported).

#### 4.8.10 SOECP-TDDFT-SOC based spin-orbit coupling calculations

In addition to the `sf-X2C` all-electron scalar relativistic Hamiltonian, the spin-orbit coupling pseudopotential (SOECP) can also be used for TDDFT-SOC spin-orbit coupling calculations by selecting the appropriate 旋軌耦合 赝势基组 and setting `hsoc` to 0 in the `xuanyuan` module (if other values are written, they are treated as 0). The other inputs are similar to or identical to the `sf-X2C-TDDFT-SOC` inputs (e.g. the core electrons are subtracted when specifying the orbital occupation in `scf`).

In the following example, the closed-shell layer ground state  $X^1\Sigma^+$  (A1) and three excited states  $^3\Pi$  (B1+B2),  $^1\Pi$  (B1+B2),  $^3\Sigma^+$  (A1) are calculated under the  $C_{2v}$  point group symmetry for the InBr molecule, where the first two  $\Lambda$ -S states are bound states that have been extensively studied experimentally and the last two  $\Lambda$ -S states are repulsive states that are of little experimental interest. In the input, the energy of the  $\Lambda$ -S state is first calculated at the TDDFT level (using the Tamm-Dancoff approximation here) and the wave function is stored, and then the energy of the  $\Omega$  state after spin-orbit coupling is calculated.

```
$COMPASS
Title
  soecp test: InBr
Basis-block
  cc-pVTZ-PP
end basis
Geometry
  In  0.0  0.0  0.0
  Br  0.0  0.0  2.45
END geometry
```

(continues on next page)

(continued from previous page)

```
group
  C(2v)      # Abelian symmetry must be used for SOC
$END

$XUANYUAN
  hsoc
  10
$END

$scf
  rks
  dft
  pbe0
$end

$TDDFT
ISF
  0
ITDA
  1
istore
  1
# 1Pi state: A1, A2, B1, B2
nroot
  0 0 1 1
$END

$TDDFT
ISF
  1
ITDA
  1
istore
  2
# 3Sigma+ and 3Pi states: A1, A2, B1, B2
nroot
  1 0 1 1
$END

$TDDFT
isoc
  2
nfiles
```

(continues on next page)

(continued from previous page)

```

2
ifgs
1
idiag
2
$END

```

The computational output of SOECP-TDDFT-SOC is similar to that of sf-X2C-TDDFT-SOC. The results are summarized below and compared with those of EOM-CCSD/SOC.

Table 4.2: InBr 分子的垂直激发能：SOECP/TDDFT-SOC 与二分量 EOM-CCSD。能量单位： $\text{cm}^{-1}$

$\Lambda$ -S 态	TDDFT	$\Omega$ 态	TDDFT-SOC	分裂	二分量 EOM-CCSD	分裂
$X^1\Sigma^+$	0	0+	0		0	
$^3\Pi$	25731	0-	24884		24516	
		0+	24959	75	24588	72
		1	25718	759	25363	775
		2	26666	948	26347	984
$^1\Pi$	35400	1	35404		36389	
$^3\Sigma^+$	38251	0-	38325			
		1	38423	98		

In addition to the SOECP basis set, the above calculation can also be done using the scalar ECP basis set in combination with 有效核电荷近似 (Zeff). As a test, first delete the SO pseudopotential part in the Br basis set and redo the above calculation, but you will find that the result is poor: The split of  $^3\Pi_2$  and  $^3\Pi_1$  is only  $850 \text{ cm}^{-1}$ , while the split of  $^3\Sigma^+$  almost zero. This is because the ECP basis set for Br with 10 core electrons has no specially optimized effective nuclear charge, and the program can only take the actual nuclear charge number of 35:

```

SO-1e[BP]
      Zeff for Wso
-----
IAtm   ZA   NCore   Zeff
-----
  1     49    28     SOECP
  2     35    10     N.A.
-----

```

For Br in the above example, it is possible to use the scalar ECP basis set cc-pVTZ-ccECP with 28 core electrons instead. The input part of the basis set is modified as follows:

```
Basis-block
```

(continues on next page)

(continued from previous page)

```
cc-pvtz-pp
Br=cc-pvtz-ccecp
end basis
```

At the beginning of the TDDFT-SOC calculation output can be seen

```
SO-1e[BP]
      Zeff for Wso
-----
IAtm   ZA   NCore   Zeff
-----
   1    49    28     SOECP
   2    35    28    1435.000
-----
```

This shows that in the single-electron spin-orbit integration of Br, the default nuclear charge number 35 is replaced with the optimized 1435.000 (in general, the larger the ECP core electron number NCore, the larger the effective nuclear charge Zeff), The SOECP integral is still calculated for the In atom. The calculation results are as follows, and it can be seen that the orbital splitting has been significantly improved:

Table 4.3: InBr 分子的 TDDFT-SOC 垂直激发能: In:SOECP, Br:SOECP 与 Br:ECP。能量单位:  $\text{cm}^{-1}$

$\Lambda$ -S 态	TDDFT	$\Omega$ 态	Br:SOECP	分裂	Br:ECP	分裂
$X^1\Sigma^+$	0	0+	0		0	
$^3\Pi$	25731	0-	24884		25019	
		0+	24959	75	25084	65
		1	25718	759	25856	772
		2	26666	948	26808	952
$^1\Pi$	35400	1	35404		35729	
$^3\Sigma^+$	38251	0-	38325		38788	
		1	38423	98	38853	65

Finally, TDDFT-SOC calculations can also be combined with the SOECP (or scalar ECP) basis set with the all-electron non-relativistic basis set. The BDF program has optimized Zeff for main group elements prior to Xe (except heavier noble gas elements). For example, continuing to use cc-pVTZ-PP for In, and using the all-electron non-relativistic basis set cc-pVTZ for Br, yields similar results to SOECP/TDDFT-SOC. Detailed results are omitted. Finally, TDDFT-SOC calculations can also be combined with the SOECP (or scalar ECP) basis set with the all-electron non-relativistic basis set. The BDF program has optimized Zeff for main group elements prior to Xe (except heavier noble gas elements).

**Attention:**

1. Precautions when using the effective nuclear charge method for TDDFT-SOC calculation: You must use 优化好的有效核电荷 to ensure accuracy. To do this, check the  $Z_{\text{eff}}$  value printed in the output file, try not to have N.A., this is especially important for ECP basis sets.
2. When SOECP or scalar ECP is combined with all-electron basis set, note about all-electron basis set: Atoms using all-electron basis set do not consider scalar relativistic correspondence, so they cannot be heavy atoms, and must use non-relativistic basis set.

#### 4.8.11 Calculation of first-order non-adiabatic coupling matrix elements (fo-NACME)

As mentioned before, the (first-order) non-adiabatic coupling matrix element is of great importance in the non-radiative leap process. In BDF, the input files of NACME between the ground state and excited state, and between the excited state and excited state are written with some differences, which are described below.

(1) NACME between ground state and excited state: D0/D1 NACME of  $\text{NO}_3$  radical (GB3LYP/cc-pVDZ)

```

$COMPASS
Title
  NO3 radical NAC, 1st excited state
Basis
  cc-pvdz
Geometry
N          0.0000000000      0.0000000000      -0.1945736441
O         -2.0700698389      0.0000000000      -1.1615808530
O          2.0700698389     -0.0000000000      -1.1615808530
O         -0.0000000000      0.0000000000       2.4934136445
End geometry
unit
  bohr
$END

$XUANYUAN
$END

$SCF
UKS
dft
  GB3LYP
spinmulti
  2

```

(continues on next page)

```
$END

$tdft
iroot
  1 # One root for each irrep
istore
  1 # File number, to be used later in $resp
crit_vec
  1.d-6
crit_e
  1.d-8
gridtol
  1.d-7 # tighten the tolerance value of XC grid generation. This helps to
        # reduce numerical error, and is recommended for open-shell molecules
$end

$resp
iprt
  1
QUAD # quadratic response
FNAC # first-order NACME
single # calculation of properties from single residues (ground state-excited
        # state fo-NACMEs belong to this kind of properties)
norder
  1
method
  2
nfiles
  1 # must be the same as the istore value in the $TDFFT block
states
  1 # Number of excited states for which NAC is requested.
# First number 1: read TDDFT results from file No. 1
# Second number 2: the second irrep, in this case A2
# (note: this is the pair symmetry of the particle-hole pair, not
# the excited state symmetry. One must bear this in mind because the
# ground state of radicals frequently does not belong to the totally
# symmetric irrep)
# If no symmetry is used, simply use 1.
# Third number 1: the 1st excited state that belongs to this irrep
  1 2 1
$end
```

Note that the integrable representation specified in the `$resp` module is pair irrep (i.e., the direct product of the integrable representations of the occupied and empty orbitals involved in the leap; for the Abelian point group, pair irrep

can be obtained from the direct product of the ground state integrable representation and the excited state integrable representation), not the irrep of the excited state. the ground state (D0) of this molecule belongs to the B1 integrable representation, and the first two-state excited state (D1) belongs to the B1 integrable representation. The pair irrep of the D1 state is therefore the direct product of B1 and B2, i.e., A2. Pair irrep can also be read from the output of the TDDFT module, i.e., the Pair column of the following output section.

No.	Pair	ExSym	ExEnergies	Wavelengths	f	D<S^2>	Dominant
↔Excitations			IPA	Ova	En-E1		
1	A2	1 B2	0.8005 eV	1548.84 nm	0.0000	0.0186	98.2% CO(bb) : B2 ( 2 )
↔	->	B1 ( 5 )	3.992 0.622	0.0000			
2	B1	1 A1	1.9700 eV	629.35 nm	0.0011	0.0399	92.2% CO(bb) : A1 ( 8 )
↔	->	B1 ( 5 )	3.958 0.667	1.1695			
3	B2	1 A2	2.5146 eV	493.06 nm	0.0000	0.0384	98.4% CO(bb) : A2 ( 1 )
↔	->	B1 ( 5 )	4.159 0.319	1.7141			
4	A1	2 B1	2.6054 eV	475.87 nm	0.0171	0.0154	87.7% CO(bb) : B1 ( 4 )
↔	->	B1 ( 5 )	3.984 0.746	1.8049			

After the calculation is completed, the result of the NACME calculation can be seen at the end of the output section of the \$resp module.

Gradient contribution from Final-NAC(R)-Escaled			
1	0.0000000000	-0.0000000000	0.0000000000
2	-0.0000000000	-0.1902838724	0.0000000000
3	-0.0000000000	0.1902838724	0.0000000000
4	-0.0000000000	0.0000000000	0.0000000000

Note that this result does not include the contribution of the electron translation factor (ETF). For some molecules, NACME without ETF may not have translation invariance, which may lead to errors in subsequent kinetic simulations. In this case, it is necessary to use a NACME that takes into account the ETF, which can be read later in the output file at the following location.

Gradient contribution from Final-NAC(S)-Escaled			
1	0.0000000000	-0.0000000000	0.0000000000
2	-0.0000000000	-0.1920053581	0.0000000000
3	-0.0000000000	0.1920053581	0.0000000000
4	-0.0000000000	0.0000000000	-0.0000000000

The program will also output vectors named dpq-R, Final-NAC(R), dpq-S, Final-NAC(S), etc. These quantities are intermediate variables that are only used to monitor the computational process, not the final NACME, and the user can generally ignore these outputs.

(2) NACME between excited and excited states: T1/T2 NACME of acetophenone (BH&HLYP/def2-SVP)

```
$compass
title
  PhCOMe
basis
  def2-SVP
geometry
  C      -0.3657620861      4.8928163606      0.0000770328
  C      -2.4915224786      3.3493223987     -0.0001063823
  C      -2.2618953860      0.7463412225     -0.0001958732
  C       0.1436118499     -0.3999193588     -0.0000964543
  C       2.2879147462      1.1871091769      0.0000824391
  C       2.0183382809      3.7824607425      0.0001740921
  H      -0.5627800515      6.9313968857      0.0001389666
  H      -4.3630645857      4.1868310874     -0.0002094148
  H      -3.9523568496     -0.4075513123     -0.0003833263
  H       4.1604797959      0.3598389310      0.0001836001
  H       3.6948496439      4.9629708946      0.0003304312
  C       0.3897478526     -3.0915327760     -0.0002927344
  O       2.5733215239     -4.1533492423     -0.0002053903
  C      -1.8017552120     -4.9131221777      0.0003595831
  H      -2.9771560760     -4.6352720097      1.6803279168
  H      -2.9780678476     -4.6353463569     -1.6789597597
  H      -1.1205416224     -6.8569277129      0.0002044899
end geometry
unit
  bohr
nosymm
$end

$XUANYUAN
$END

$SCF
rks
dft
  bhhlyp
$END

$tdft
isf # request for triplets (spin flip up)
  1
ialda # use collinear kernel (NAC only supports collinear kernel)
  4
iroot
```

(continues on next page)

(continued from previous page)

```

2 # calculate T1 and T2 states
crit_vec
1.d-6
crit_e
1.d-8
istore
1
iprt
2
$end

$resp
iprt
1
QUAD
FNAC
double # calculation of properties from double residues (excited state-excited
        # state fo-NACMEs belong to this kind of properties)
norder
1
method
2
nfiles
1
pairs
1 # Number of pairs of excited states for which NAC is requested.
1 1 1 1 1 2
noresp # do not include the quadratic response contributions (recommended)
$end

```

NACME was calculated for the T1 and T2 states.

Gradient contribution from Final-NAC(R)-Escale			
1	0.0005655253	0.0005095355	-0.2407937116
2	-0.0006501682	-0.0005568029	0.5339003311
3	0.0009640605	0.0003767996	-2.6530192038
4	-0.0013429266	-0.0034063171	1.6760344312
5	0.0010446538	0.0006384285	-0.8024123329
6	-0.0001081722	-0.0006245719	-0.0487310115
7	-0.0000001499	0.0000176176	-0.0730900968
8	-0.0000214634	0.0000165092	0.3841606239
9	0.0000026057	-0.0000025322	-0.2553378323
10	-0.0002028358	-0.0000591642	0.5800987974
11	-0.0000166820	0.0000105734	0.2713836450

(continues on next page)

(continued from previous page)

12	-0.0023404123	0.0052038311	3.5121827769
13	0.0021749503	-0.0012164868	-2.7480141157
14	0.0000433873	-0.0011202812	0.2896243729
15	0.1407516324	0.1432264573	-0.1655701318
16	-0.1407399684	-0.1429881941	-0.1657943551
17	-0.0000034197	0.0004577563	-0.0833951446

Similar to the case of the ground state, the

#### 4.8.12 Definitization of the excited states

```

$COMPASS
Basis
  cc-pvdz
Geometry
  C      0.000000    0.000000    0.000000
  C      1.332000    0.000000    0.000000
  H     -0.574301   -0.928785    0.000000
  H     -0.574301    0.928785    0.000000
  H      1.906301    0.928785    0.000000
  H      1.906301   -0.928785    0.000000
  C     -0.000000    0.000000    3.5000
  C      1.332000   -0.000000    3.5000
  H     -0.574301    0.928785    3.50000
  H     -0.574301   -0.928785    3.50000
  H      1.906301   -0.928785    3.50000
  H      1.906301    0.928785    3.50000
End geometry
Group
  C(1)
Nfragment # must input: number of fragment, should be 1
  1
$END

$xuanyuan
$end

$scf
rks
dft
  B3lyp
$end

```

(continues on next page)

(continued from previous page)

```

$TDDFT
ITDA
  1
IDIAG
  1
istore
  1
iroot
  4
crit_e # set a small threshold for TDDFT energy convergence
  1.d-8
$END

# calculate local excited states (LOCALES)
$elecoup
locales
  1
$END

&database
fragment 1 12 # first fragment with 12 atoms, next line gives the atom list
  1 2 3 4 5 6 7 8 9 10 11 12
&end

```

The TDA calculates 4 excited states and the output is as follows,

No.	Pair	ExSym	ExEnergies	Wavelengths	f	D<S^2>	Dominant
↔Excitations			IPA	Ova	En-E1		
1	A 2	A	7.4870 eV	165.60 nm	0.0000	0.0000	82.6% CV(0): A( 16 )-
↔>	A( 17 )		13.476 0.820	0.0000			
2	A 3	A	8.6807 eV	142.83 nm	0.0673	0.0000	79.6% CV(0): A( 16 )-
↔>	A( 18 )		14.553 0.375	1.1937			
3	A 4	A	9.0292 eV	137.31 nm	0.0000	0.0000	62.4% CV(0): A( 16 )-
↔>	A( 20 )		15.353 0.398	1.5422			
4	A 5	A	9.0663 eV	136.75 nm	0.0000	0.0000	50.4% CV(0): A( 15 )-
↔>	A( 18 )		15.688 0.390	1.5793			

The process of domainization and the excited states in the fixed domain are,

```

No. 8 iteration
Pair States : 1 2
aij,bij,abij -.25659893E-01 -.48045653E-11 0.25659893E-01
cos4a,sin4a 0.10000000E+01 -.18724027E-09

```

(continues on next page)

(continued from previous page)

```
cosa,sina 0.10000000E+01 0.00000000E+00
Pair States : 1 3
aij,bij,abij -.40325646E-02 0.35638586E-11 0.40325646E-02
cos4a,sin4a 0.10000000E+01 0.88376974E-09
cosa,sina 0.10000000E+01 0.00000000E+00
Pair States : 1 4
aij,bij,abij -.25679319E-01 -.28753641E-08 0.25679319E-01
cos4a,sin4a 0.10000000E+01 -.11197197E-06
cosa,sina 0.10000000E+01 0.27877520E-07
Pair States : 2 3
aij,bij,abij -.39851115E-02 -.27118892E-05 0.39851124E-02
cos4a,sin4a 0.99999977E+00 -.68050506E-03
cosa,sina 0.99999999E+00 0.17012628E-03
Pair States : 2 4
aij,bij,abij -.42686102E-02 -.95914926E-06 0.42686103E-02
cos4a,sin4a 0.99999997E+00 -.22469825E-03
cosa,sina 0.10000000E+01 0.56174562E-04
Pair States : 3 4
aij,bij,abij -.67873307E-01 -.47952471E-02 0.68042488E-01
cos4a,sin4a 0.99751360E+00 -.70474305E-01
cosa,sina 0.99984454E+00 0.17632279E-01
Sum= 0.13608498 Max Delta= 0.00531009
```

```
No. 9 iteration
Pair States : 1 2
aij,bij,abij -.40325638E-02 0.35621782E-11 0.40325638E-02
cos4a,sin4a 0.10000000E+01 0.88335323E-09
cosa,sina 0.10000000E+01 0.00000000E+00
Pair States : 1 3
aij,bij,abij -.25690755E-01 -.11200070E-08 0.25690755E-01
cos4a,sin4a 0.10000000E+01 -.43595721E-07
cosa,sina 0.10000000E+01 0.10536712E-07
Pair States : 1 4
aij,bij,abij -.25690755E-01 -.10900573E-11 0.25690755E-01
cos4a,sin4a 0.10000000E+01 -.42429944E-10
cosa,sina 0.10000000E+01 0.00000000E+00
Pair States : 2 3
aij,bij,abij -.41480079E-02 -.83549288E-06 0.41480080E-02
cos4a,sin4a 0.99999998E+00 -.20142027E-03
cosa,sina 0.10000000E+01 0.50355067E-04
Pair States : 2 4
aij,bij,abij -.41480100E-02 0.17462423E-06 0.41480100E-02
cos4a,sin4a 0.10000000E+01 0.42098314E-04
```

(continues on next page)

(continued from previous page)

```

cosa,sina 0.10000000E+01 0.10524580E-04
Pair States :    3    4
aij,bij,abij -.68042492E-01 0.19389042E-08 0.68042492E-01
cos4a,sin4a 0.10000000E+01 0.28495490E-07
cosa,sina 0.10000000E+01 0.74505806E-08
Sum=      0.13608498 Max Delta=      0.00000000

***** Diabatic Hamiltonian matrix *****
      State1      State2      State3      State4
State1  7.486977    0.000000    0.000000    0.000000
State2  0.000000    9.029214   -0.000020    0.000021
State3  0.000000   -0.000020    8.873501    0.192803
State4  0.000000    0.000021    0.192803    8.
↪873501*****

```

其中，对角元为定域激发态的能量，非对角元为两个定域态之间的耦合，这里的能量单位是 eV。where the diagonal element is the energy of the fixed-domain excited state and the non-diagonal element is the coupling between the two fixed-domain states, where the unit of energy is eV.

## 4.9 Nuclear magnetic resonance shielding constants

BDF supports the restricted Hartree-Fock (RHF) and restricted Kohn-Sham (RKS) methods for NMR shielding constant (NMR) calculations, where the problem of the outer vector potential canonical origin can be handled using the Common gauge method and GIAO (gauge-including atomic orbitals).

**Warning:** Since the libcint library is required for NMR calculation, a line needs to be added in the calculation script: export USE\_LIBCINT=yes

### 4.9.1 NMR calculation example

The following is the input file for the NMR shielding constant calculation of methane molecules:

```

$COMPASS # Molecular coordinate input and symmetry judgment
Title
CH4 Molecule NR-DFT-NMR      # job title
Basis
CC-PVQZ                       # basis set
Geometry
C 0.000 00000 0.000 # molecule geometry
H 1.196 1.196 1.196

```

(continues on next page)

```
H -1.196 -1.196 1.196
H -1.196 1.196 -1.196
H 1.196 -1.196 -1.196
END geometry
nosymm # NMR module does not support symmetry
UNIT
  BOHR # input molecule geometry in bohr
$END

$xuanyuan # Single and double electron integral correlation setting and calculation
$end

$SCF # Self consistent field calculation module
RKS # Restrict Kohn-Sham
DFT
  b3lyp
$END

$NMR # NMR shielding constant calculation module
icg
  1 # can enter 0 or 1. 0 means no common gauge calculation and 1 means common
  ↳gauge calculation. The default value is 0.
igiao
  1 # can enter 0 or 1. 0 means no GIAO calculation and 1 means GIAO
  ↳calculation. The default value is 0.
$END
```

The four modules `compass`, `xuanyuan`, `scf` and `nmr` are called sequentially to complete the calculation. The `scf` module performs the RKS calculation. Based on the results of the RKS calculation, subsequent NMR calculations are performed, where the NMR calculation is followed by the COMMON GAUGE calculation and the GIAO calculation, which gives the isotropic and anisotropic NMR shielding constants for all atoms.

## 4.9.2 COMMON GAUGE

The NMR calculation of COMMON GAUGE can be controlled by the keyword `icg`:

```
$NMR
icg
  1
$END
```

You can enter 0 or 1. The default value is 0, which means that no COMMON GAUGE calculation is performed, and 1, which means that a COMMON GAUGE calculation is performed.

In the COMMON GAUGE calculation, the canonical origin is located at the origin of the coordinates (0, 0, 0) by default. The canonical origin can be specified at an atom by using the keyword `igatom`, or it can be set to a specified location in space by using `cgcoord`, as follows.:

```
$NMR
icg
  1
igatom
  3          # Specify the gauge origin on atom 3, enter an integer number,
↳ranging from 0 to the number of molecular atoms,
          # If you enter a value of 0, the specification origin is specified at
↳the coordinate origin
cgcoord
  1.0 0.0 0.0 # enter 3 real numbers, and place the origin of the specification on
↳the point with the spatial coordinates of (1.0, 0.0, 0.0)
cgunit
  angstrom   # The unit of cgcoord coordinate. The default value is atomic unit.
↳When the input is Angstrom, the input standard origin coordinate
          # In angstroms; For other inputs (such as Bohr, AU), the coordinate
↳unit is atomic unit, and the input is case insensitive
$END
```

When both `igatom` and `cgcoord` are present in the input, the latter input prevails. For example, in the above example, the final canonical origin is set at the spatial coordinates (1.0, 0.0, 0.0) (in angstroms). If both parameters `igatom` and `cgcoord` are not entered, the NMR value of COMMON GAUGE is calculated with the normative origin at the coordinate origin, i.e. at the position (0.0, 0.0, 0.0)

The Common gauge calculation in the output file starts at `[nmr_nr_cg]`, as follows:

```
[nmr_nr_cg]
Doing nonrelativistic-CG-DFT nmr...

[nmr_set_common_gauge]
set the common gauge origin as the coordinate origin(default)
  0.000000000000    0.000000000000    0.000000000000
```

Omitting the middle part of the output, the final result is output as follows:

```
Isotropic/anisotropic constant by atom type:
atom-C
  186.194036    0.000003
atom-H
  31.028177    9.317141
  31.028176    9.317141
  31.028177    9.317141
```

(continues on next page)

```
31.028177    9.317141
```

The NMR shielding constants in ppm for C and H atoms, respectively, are shown in the first column as isotropic shielding constants and in the second column as anisotropic shielding constants.

### 4.9.3 GIAO

The NMR calculation of GIAO can be controlled by the keyword `igiao`:

```
$NMR
igiao
  1
$END
```

You can enter either 0 or 1. The default value is 0, i.e., no GIAO calculation is performed, and when entered as 1, GIAO calculation is performed.

**Warning:** In the NMR module, `icg` and `igiao` can be entered as either 1, which sets either calculation to be performed, or both (i.e. both calculations), but not both or 0, otherwise the NMR module will not produce any NMR shielding constant values.

The GIAO calculation in the output file starts at `[nmr_nr_giao]`, as follows:

```
[nmr_nr_giao]
Doing nonrelativistic-GIAO-DFT nmr

[set_para_for_giao_eri]

[nmr_int]
Doing nmr integral of operators resulting from the response of B10...

No. of pGTOs and cGTOs:    196    196

giao integrals...
```

Omitting the middle part of the output, the final result is output as follows:

```
Isotropic/anisotropic constant by atom type:
atom-C
  186.461988    0.000019
atom-H
  31.204947    9.070916
```

(continues on next page)

(continued from previous page)

31.204944	9.070916
31.204947	9.070921
31.204946	9.070920

As in the case of COMMON GAUGE, the above results are the GIAO shielding constants in ppm for C and H atoms, respectively, with the first column being the isotropic shielding constant and the second column being the anisotropic shielding constant.

**Warning:** The keyword `Isotropic/anisotropic constant by atom type` in the output is exactly the same for GIAO and COMMON GAUGE, when reading the result, you should pay attention to whether it is after `[nmr_nr_cg]` or after `[nmr_nr_giao]` to distinguish between COMMON GAUGE result or GIAO result.

## 4.10 Relativistic effects

The relativistic effect mainly includes two parts: scalar relativistic effect and spin-orbit coupling effect. Relativistic effects are important in the study of organic light-emitting mechanisms, such as inter-system scattering of electronic states due to spin-track coupling, and changes in the spin of transition states and products. The innermost core electrons of the atom are most strongly affected by relativistic effects on the one hand, and insensitive to chemical changes on the other hand, so there are different ways to deal with them, mainly all-electron relativistic and effective core potential (ECP) two types of methods.

1. **All-electron methods** all the inner layer electrons are made variational treatment. The common all-electron relativistic Hamiltonians are: the zero-order regular approximation (ZORA), the second-order Douglas-Kroll-Hess (DKH2), the exact dichotomy (X2C) proposed by Wenjian Liu et al. ZORA and DKH2 have no advantages in terms of accuracy and efficiency and are not recommended.
2. The core electrons of **ECP** heavy atoms are replaced by a pre-fitted effective potential function. When there are more heavy atoms, ECP can significantly reduce the variational degrees of freedom and improve the computational efficiency. The ECP is divided into two categories, pseudopotential (PP) and model core potential (MCP), depending on whether the valence electron wave function has nodes in the core layer or not, where MCP has to combine a large number of Gaussian functions in order to reproduce the valence electron wave function nodes correctly, leading to an insignificant improvement in computational efficiency, and is therefore less used. ECP in BDF programs refers to PP.

The BDF basis set library provides a large number of all-electron relativistic basis sets and ECP basis sets .

**Warning:**

1. Do not mix X2C Hamiltonian and ECP basis sets

2. X2C relativistic calculations must use non-shrinking basis sets or specially optimized shrinking basis sets, but the first 18 elements are not mandatory.

### 4.10.1 Scalar relativistic effects

- **All-electron methods**

The BDF can consider scalar relativistic effects through the spinless X2C Hamiltonian (sf-X2C) and its local approximation variants sf-X2C-AXR, sf-X2C-AU. For example :

```
$xuanyuan
heff
  23
nuclear
  1
$end
```

In the above input, `heff` calls scalar relativistic Hamiltonians such as sf-X2C (3, 4, or 21), sf-X2C-AXR (atomic X-matrix approximation, 22), and sf-X2C-AU (atomic U-transformation approximation, 23), where 21, 22, 23 have analytic derivatives. For molecular systems not involving heavy element-heavy element bonding above 5d, sf-X2C-AU has the highest efficiency without loss of precision and is the recommended method. Otherwise, sf-X2C-AXR or sf-X2C is used.

The `nuclear` option set to 1 indicates the use of a finite nuclear model, which is generally not required. However, some relativistic shrinkage basis groups take into account nucleus size effects, or when calculating the electronic properties near the nucleus, in these cases the finite nuclear model is used.

The types of calculations supported by sf-X2C and its local variants are: single point energies, structure optimization, and some first-order single electron properties . Analytic Hessian and second-order single-electron properties are under development.

- **ECP**

ECP must be combined with a non-relativistic Hamiltonian, where relativistic effects are implicit in the pseudopotential parameters. The types of ECP calculations supported by the BDF are: single-point energy and some single electron properties . Gradient and Hessian for ECP are under development.

## 4.10.2 Spin-orbit coupling interactions

BDF can handle spin-orbit coupling between different spin multiplet electronic states in TDDFT single-point calculations by means of the state interaction (SI) method. It needs to be specified in the `xuanyuan` module via the `hsoc` keyword how to calculate the spin-orbit integrals. See the TDDFT section for an example.

Depending on the adopted Hamiltonian, spin-orbit coupling can also be divided into two categories: all-electron and ECP.

- **All-electron methods**

Although the contribution of the two-electron spin-orbit integral is smaller than that of the single-electron spin-orbit integral, the effect on the spin-orbit coupling effect may reach 1/5~1/3, and therefore cannot be neglected. A singleelectron spin-orbit integral + a molecular mean-field two-electron spin-orbit integral with a single-center approximation (`so1e + SOMF-1c; hsoc = 2`) is suggested. It can be combined with the `sf-X2C` scalar relativistic Hamiltonian and, for the light element system, with the non-relativistic Hamiltonian. In addition, it is possible to correct the single-electron spin-orbit integrals for shielded nuclei [66, 67] or effective nuclear charges [68], approximating the effect of the twoelectron spin-orbit integrals, but this may cause unpredictable errors on the core-layer orbital properties.

- **ECP**

It includes two treatments:

1. the spin-orbit coupling pseudopotential, which requires the addition of an additional SO potential function to the scalar ECP (SOECP; see the spin-orbit coupling pseudopotential basis set in the basis set library)
2. the effective nuclear charge [68, 69] .

Since the effect of the two-electron spin-orbit interaction is already included in the fitting parameters of the SO potential or in the empirical parameters of the effective nuclear charge, it is sufficient to calculate the single-electron spin-orbit integral. In the BDF, the atoms described by SOECP and the atoms described by scalar ECP or all-electron non-relativity can be used separately, by setting `hsoc` to 10 in the `xuanyuan` module.

It is important to note that the effective nuclear charge supports a limited number of elemental and base group types. For all-electron base groups, only the main group elements before Xe are supported, with the exception of the heavier noble gas elements Ne, Ar, and Kr. For scalar ECP base groups, although more elements are supported, the number of core electrons must be consistent, see the following table:

Table 4.4: The number of electrons and atoms in the core of the scalar ECP base group supported by the effective nuclear charge

Atom	ZA	NCore
Li-F	3- 9	2
Na-Cl, Sc-Cu, Zn, Ga	11-17, 21-29, 30, 31	10
K -Ca	19-20	18
Ge-Br, Y -Ag, Cd, In	32-35, 39-47, 48, 49	28
Rb-Sr	37-38	36
Sn-I, La	50-53, 57	46
Cs-Ba	55-56	54
Hf-Au, Hg, Tl	72-79, 80, 81	60
Pb-At	82-85	78

For more details, such as  $Z_{\text{eff}}$  parameters, references, etc., see `soint_util/zefflib.F90`. If the effective nuclear charge method is used for unsupported elements or groups, the results of the spin-orbit coupling calculations are unreliable.

## 4.11 QM/MM Combination Approach

The QM/MM combination method generally divides the system into two zones, the QM zone and the MM zone. The total energy of the system is written as:

$$E_{QM/MM}(S) = E_{MM}(O) + E_{QM}(I + L) + E_{QM/MM}(I, O)$$

where,  $S$  denotes the system,  $I$  denotes the QM layer,  $O$  denotes the MM layer, and  $L$  denotes the linking atom.  $E_{MM}(O)$  is calculated using molecular mechanics force fields, and  $E_{QM/MM}(I, O)$  includes two terms:

$$E_{QM/MM}(I, O) = E_{nuc-MM} + V_{elec-MM}$$

$E_{nuc-MM} + V_{elec-MM}$  in the BDF by adding an external point charge to the QM region.

So the total energy of the whole system consists of two parts,  $E_{MM}$  is calculated using molecular mechanics method,  $E_{QM}$  和  $E_{QM/MM}$  are calculated using quantum chemical method. Meanwhile, the interactions between QM and MM regions also include VDW interactions, etc., which will not be discussed here. For the broken bonds in the QM and MM regions, the linked-atom model is generally used to describe them.

The BDF program mainly performs the quantum chemistry calculations, while the rest of the calculations are performed by the `pdynamo2.0` program package modified by the group. The rest is done by the `pdynamo2.0` package, which has been modified by the group. See the relevant examples:

---

**Note:** The installation of the `pdynamo` program is described in the package. Detailed features of the package can be found in the Help file in the package. This manual only provides instructions and examples for QM/MM calculations using BDF.

For reference to QM/MM computing environment installation and configuration, see *QM/MM computing environment configuration*

### 4.11.1 Input File Preparation

In general, before QM/MM calculations, molecular dynamics simulations of the target system are required to obtain a suitable initial conformation. Different molecular dynamics software has different output files. pDynamo-2 currently supports Amber, CHARMM, Gromacs and other force fields, and also supports PDB, MOL2, xyz and other formats to read in molecular coordinates.

**Note:** When using PDB, MOL2, or xyz files as input, the pDynamo program only supports the OPLS force field, which is not recommended for small molecules and non-standard amino acid force field parameters are incomplete. It is recommended to prefer the Amber program, which enters the force field parameters through the topology file. If you only do QM calculations, various input methods are available.

In Amber, for example, the structure of interest is extracted from the kinetic simulation trajectory and stored in a `crd` file, which together with the corresponding topology file `.prmtop` can be used as the starting point for QM/MM calculations. the Python script is as follows.

```
from pBabel import AmberCrdFile_ToCoordinates3, AmberTopologyFile_ToSystem
# Read input information
molecule = AmberTopologyFile_ToSystem(Topfile)
molecule.coordinates3 = AmberCrdFile_ToCoordinates3(CRDfile)
```

At this point, the molecule information is stored in the `molecule` structure. In the specific QM/MM calculation, energy calculation and geometric configuration optimization of the system are required. Also, the active region can be defined in the MM region to accelerate the calculation.

### 4.11.2 Total energy calculation

Take a 10 Å water box as an example, after the molecular dynamics simulation, the files are extracted as `wat.prmtop`, `wat.crd`, and the full quantum chemical calculation can be performed for the system with the following code:

```
import glob, math, os
from pBabel import AmberCrdFile_ToCoordinates3, AmberTopologyFile_ToSystem
from pCore import logFile
from pMolecule import QCModelBDF, System
# Read the coordinates and topology information of the water box
molecule = AmberTopologyFile_ToSystem ("wat.prmtop")
```

(continues on next page)

(continued from previous page)

```

molecule.coordinates3 = AmberCrdFile_ToCoordinates3("wat.crd")
# Define the energy calculation mode, which is the density functional calculation of
↳the whole system, GB3LYP:6-31g
model = QCModelBDF("GB3LYP:6-31g")
molecule.DefineQCModel(model)
molecule.Summary() #Output system calculation setting information
# Calculate total energy
energy = molecule.Energy()

```

Methods and basis groups can be defined in the `QCModelBDF` class `GB3LYP:6-31g`, with `:` partitioning between methods and basis groups. In the above example, it is also possible to select the molecule of interest (e.g., the fifth water molecule) for the QM/MM calculation, using the QM method for the fifth water molecule and the MM (in this case, the amber force field) for the rest. Since periodic boundary conditions are used in the MD calculation, and the QM/MM method does not support the use of periodic boundary conditions, an option is added to the script to turn off periodic boundary conditions.

```

molecule.DefineSymmetry( crystalClass = None )

```

The class `Selection` is defined in `pDynamo` and can be used to select specific QM atoms, as described in the usage notes. The script for selecting QM atoms in script is as follows:

```

qm_area = Selection.FromIterable(range(12, 15))
#12, 13 and 14 are atomic list index values (this value = Atomic serial number - 1),
↳which is equal to selecting No. 15 water molecule
molecule.DefineQCModel(qcModel, qcSelection = qm_area)

```

Overall, the script for the combined QM/MM energy calculation is as follows :

```

import glob, math, os
from pBabel import AmberCrdFile_ToCoordinates3, AmberTopologyFile_ToSystem
from pCore import logFile, Selection
from pMolecule import NBModelORCA, QCModelBDF, System
# . Define the energy models.
nbModel = NBModelORCA()
qcModel = QCModelBDF("GB3LYP:6-31g")
# . Read the data.
molecule = AmberTopologyFile_ToSystem("wat.prmtop")
molecule.coordinates3 = AmberCrdFile_ToCoordinates3("wat.crd")
# .Close symmetry to a system
molecule.DefineSymmetry(crystalClass = None) # QM/MM need Close the symmetry.
# .Selection qm area
qm_area = Selection.FromIterable(range (12, 15)) # Select WAT 5 as the QM area.
# . Define the energy model.

```

(continues on next page)

(continued from previous page)

```

molecule.DefineQCModel (qcModel, qcSelection = qm_area)
molecule.DefineNBModel (nbModel)
molecule.Summary()
# . Calculate
energy = molecule.Energy()

```

**Note:**

- QM/MM calculation supports two input modes. For simple examples, it can be used as parameter input in QC-ModelBDF class.
- relatively complex examples can be input in the form of calculation template.

### 4.11.3 Geometry Optimization

QM/MM geometry optimization is generally not easy to converge and requires a lot of skills in practice. It is common to fix the MM zone and optimize the QM zone, and then fix the QM zone and optimize the MM zone. After several cycles, the QM and MM regions are optimized at the same time. The convergence of the optimization depends on the choice of the QM zone and the presence of charged atoms at the QM/MM boundary. In order to speed up the optimization, it is possible to fix the MM region during the calculation and select only a suitable region close to the QM region as the active region, where the coordinates can change during the optimization. The following arithmetic example for the optimization of geometric configurations is given.

```

import glob, math, os.path

from pBabel import AmberCrdFile_ToCoordinates3, \
                    AmberTopologyFile_ToSystem , \
                    SystemGeometryTrajectory , \
                    AmberCrdFile_FromSystem , \
                    PDBFile_FromSystem , \
                    XYZFile_FromSystem

from pCore import Clone, logFile, Selection

from pMolecule import NBModelORCA, QCModelBDF, System

from pMoleculeScripts import ConjugateGradientMinimize_SystemGeometry

# Defines the Opt interface
def opt_ConjugateGradientMinimize(molecule, selection):
    molecule.DefineFixedAtoms(selection) # Define fixed atoms

```

(continues on next page)

```
#Define optimization methods
ConjugateGradientMinimize_SystemGeometry(
    molecule,
    maximumIterations = 4, # Maximum number of optimization steps
    rmsGradientTolerance = 0.1, #Optimize convergence control
    trajectories = [(trajectory, 1)]
) # Defines the frequency at which the track is saved
# . Define the energy models.
nbModel = NBModelORCA()
qcModel = QCModelBDF("GB3LYP:6-31g")
# . Read the data.
molecule = AmberTopologyFile_ToSystem ("wat.prmtop")
molecule.coordinates3 = AmberCrdFile_ToCoordinates3("wat.crd")
# . Close symmetry to a system
molecule.DefineSymmetry(crystalClass = None) # QM/MM need Close the symmetry.
# . Define Atoms List
natoms = len(molecule.atoms) # The total number of atoms in the
↪system
qm_list = range(12, 15) # QM region atoms
activate_list = range(6, 12) + range(24, 27) # MM region active atom (can be moved
↪during optimization)
#Defines the MM region atom
mm_list = range(natoms)
for i in qm_list:
    mm_list.remove(i) # MM deletes the QM atom
mm_inactivate_list = mm_list[:]
for i in activate_list :
    mm_inactivate_list.remove(i)
# Enter the QM region atom
qmmtest_qc = Selection.FromIterable(qm_list) # Select WAT 5 as the QM area.
# Define each selection zone
selection_qm_mm_inactivate = Selection.FromIterable(qm_list + mm_inactivate_list)
selection_mm = Selection.FromIterable(mm_list)
selection_mm_inactivate = Selection.FromIterable(mm_inactivate_list)
# . Define the energy model.
molecule.DefineQCModel(qcModel, qcSelection = qmmtest_qc)
molecule.DefineNBModel(nbModel)
molecule.Summary()
#Calculate the total energy at the start of the optimization
eStart = molecule.Energy()
#Defines the output file
outlabel = 'opt_watbox_bdf'
if os.path.exists(outlabel):
```

(continues on next page)

(continued from previous page)

```

    pass
else:
    os.mkdir (outlabel)
outlabel = outlabel + '/' + outlabel
# Define the output trajectory
trajectory = SystemGeometryTrajectory (outlabel + ".trj" , molecule, mode = "w")
# Start the first phase of optimization
# Define two steps to optimization
iterations = 2
# Sequentially fix the QM area and mm area for optimization
for i in range(iterations):
    opt_ConjugateGradientMinimize(molecule, selection_qm_mm_inactivate) #Fixed QM_
    ↪region optimization
    opt_ConjugateGradientMinimize(molecule, selection_mm) #Fixed MM_
    ↪region optimization
# Start the second phase of optimization
# The QM area and mm area are optimized at the same time
opt_ConjugateGradientMinimize(molecule, selection_mm_inactivate)
#Output optimized total energy
eStop = molecule.Energy()
#Save optimized coordinates, which can be xyz/crd/pdb, etc.
XYZFile_FromSystem(outlabel + ".xyz", molecule)
AmberCrdFile_FromSystem(outlabel + ".crd" , molecule)
PDBFile_FromSystem(outlabel + ".pdb" , molecule)

```

#### 4.11.4 QM/MM-TDDFT algorithm example

After the geometry optimization, the TDDFT calculation can be performed based on the base state calculated by QM/MM. The BDF program interface is designed with a calculation template function that updates the system coordinates based on the `.inp` file given by the user. At the same time, different QM regions can be selected as needed for the geometry optimization and excited state calculation. For example, in order to consider solvation effects, the first hydrated layer of the molecule of interest can be added to the QM region for QM/MM-TDDFT calculations. Taking the example of the calculation done in the previous section, the calculation can be continued by adding the following code.

```

#Continue with the geometry optimization code from the previous section.
#Start the TDDFT calculation. Use a template file as input.
qcModel = QCModelBDF_template(template = 'head_bdf_nosymm.inp')
# Adjust the QM region atoms
tdtest = Selection.FromIterable(qm_list + activate_list) # Redefine the QM_
    ↪region.
molecule.DefineQCModel(qcModel, qcSelection = tdtest)
molecule.DefineNBModel(nbModel)

```

(continues on next page)

(continued from previous page)

```
molecule.Summary()  
#Energy calculation using the method in the template, (which can be TDDFT)  
energy = molecule.Energy()
```

In the above code, the template chosen is the input file of the BDF with the following file contents:

```
$COMPASS  
Title  
  cla_head_bdf  
Basis  
  6-31g  
Geometry  
  H 100.723 207.273 61.172  
  MG 92.917 204.348 68.063  
  C 95.652 206.390 67.185  
END geometry  
Extcharge  
  point  
nosymm  
$END  
  
$XUANYUAN  
RSOMEGA  
  0.33  
$END  
  
$SCF  
RKS  
DFT  
  cam-B3LYP  
$END  
  
$tdft #TDDFT calculation control  
iprt  
  3  
iroot  
  5  
$end
```

## 4.12 Structural Optimization and Frequency Calculation

The purpose of structural optimization is to find the minimum point of the potential energy surface of the system. The minimum point that can be found depends on the initial structure provided in the input file. The closer to the minima, the easier it is to converge to the minima.

Structural optimization is mathematically equivalent to the problem of finding the extrema of a multivariate function :

$$F_i = -\frac{\partial E(R_1, R_2, \dots, R_N)}{\partial R_i} = 0, i = 1, 2, \dots, N$$

Commonly used algorithms for structural optimization are as follows :

1. Steepest descent: The steepest descent method is a line search along the direction of the negative gradient, which is very efficient for structures far from the minima, but slow to converge near the minima and easy to oscillate.
2. Conjugate gradient: Conjugate gradient method is a modification of the most rapid descent method, the optimization direction of each step and the previous step of the combination of optimization direction, can somewhat alleviate the problem of oscillation
3. Newton method: The idea of Newton method is to expand the function with respect to the current position in the Taylor series. Newton method converges quickly, for the quadratic function can go to the minimum in one step. However, the Newton method requires solving the Hessian matrix, which is very expensive to compute, and the quasi-Newton method is generally used in geometric optimization.
4. Quasi-Newton method: The quasi-Newton method constructs the Hessian matrix by approximation, and the Hessian matrix of the current step is obtained based on the forces of the current step and the Hessian matrix of the previous step. There are various approaches, the most commonly used is the BFGS method, in addition to DFP, MS, PSB, etc. Since the Hessian of the quasi-Newton method is constructed approximately, the accuracy of each optimization step is lower than that of the Newton method, and the number of steps required to reach convergence is more than that of the Newton method. However, the total optimization time is still significantly reduced because each step takes much less time.

The structural optimization of the BDF is implemented by the BDFOPT module, which supports the Newtonian and quasi-Newtonian based methods for minima and transition. The BDFOPT module supports Newtonian and quasi-Newtonian methods for the optimization of minima and transition structures, and supports restricted structure optimization. The following is an example of the input file format of the BDFOPT module, and an explanation of the output file.

### 4.12.1 Base state optimization: Optimization of monochloromethane (CH<sub>3</sub>Cl) at the B3LYP/def2-SV(P) level

```
$compass
title
  CH3Cl geomopt
```

(continues on next page)

```
basis
  def2-SV(P)
geometry
C          2.67184328    0.03549756   -3.40353093
H          2.05038141   -0.21545378   -2.56943947
H          2.80438882    1.09651909   -3.44309081
H          3.62454948   -0.43911916   -3.29403269
Cl         1.90897396   -0.51627638   -4.89053325
end geometry
$end

$bdfopt
solver
  1
$end

$xuanyuan
$end

$scf
rks
dft
  b3lyp
$end

$resp
geom
$end
```

The RESP module is responsible for calculating the DFT gradient. Unlike most other tasks, in the structure optimization task, the program does not sequentially and single, linear invocation of the modules, but rather, the modules are invoked several times iteratively. The specific sequence of calls is as follows:

1. run COMPASS, which reads the molecular structure and other information;
2. run BDFOPT to initialize the intermediate quantities needed for structure optimization;
3. BDFOPT starts a separate BDF process to calculate the energy and gradient under the current structure, which only executes COMPASS, XUANYUAN, SCF, and RESP modules and skips BDFOPT. i.e., most of the time the user will find two BDF processes running independently of each other, one of which is the process to which BDFOPT belongs and is waiting. is in the waiting state, while the other process is performing energy and gradient calculations. To avoid cluttering the output file, the output of the latter process is automatically redirected to a file with the `.out.tmp` suffix, thus separating it from the output of the BDFOPT module (which is typically redirected by the user to an `.out` file)

4. when the latter process is finished, BDFOPT aggregates the energy and gradient information of the current structure and adjusts the molecular structure accordingly with a view to reducing the energy of the system;
5. BDFOPT determines whether the structure converges according to the gradient of the current structure and the size of the current geometry step, if it converges, or if the structure optimization reaches the maximum number of iterations, the program ends; if it does not converge, the program jumps to step 3.

Therefore, the `.out` file contains only the output of COMPASS and BDFOPT modules, which can be used to monitor the process of structural optimization, but does not contain information on SCF iterations, Buju analysis, etc., which needs to be viewed in the `.out.tmp` file

Taking the structure optimization task of  $\text{CH}_3\text{Cl}$  above as an example, the output of the BDFOPT module in the `.out` file can be seen as follows:

```

Geometry Optimization step :    1

Single Point SCF for geometry optimization, also get force.

### [bdf_single_point] ### nstate= 1
Allow rotation to standard orientation.

BDFOPT run - details of gradient calculations will be written
into .out.tmp file.

...

### JOB TYPE = SCF ###
E_tot= -499.84154693
Converge= YES

### JOB TYPE = RESP_GSGRAD ###
Energy= -499.841546925072
  1      0.0016714972      0.0041574983      -0.0000013445
  2     -0.0002556962     -0.0006880567      0.0000402277
  3     -0.0002218807     -0.0006861734     -0.0000225761
  4     -0.0003229876     -0.0006350885     -0.0000059774
  5     -0.0008670369     -0.0021403962     -0.0000084046

```

It can be seen that BDFOPT calls the BDF program itself to calculate the SCF energy and gradient of the molecule under the initial guess structure. the detailed output of the SCF and gradient calculations is in the `.out.tmp` file, while the `.out` file only extracts the energy and gradient values, as well as information on whether the SCF converges or not. The unit of energy is Hartree and the unit of gradient is Hartree/Bohr.

Since BDFOPT is a structure optimized in redundant internal coordinates (`solver = 1`), in order to generate the molecular structure for the next step, the redundant internal coordinates of the molecule must be generated first. Therefore,

in the first step of the structure optimization, the output file also gives the definition of the individual redundant internal coordinates (i.e. the atomic numbers involved in the formation of the corresponding bonds, bond angles and dihedral angles), as well as their values (bond lengths in angstroms, bond angles in degrees).

```

*****|
Redundant internal coordinates on Angstrom/Degree

Name      Definition      Value      Constraint
R1        1  2                1.0700    No
R2        1  3                1.0700    No
R3        1  4                1.0700    No
R4        1  5                1.7600    No
A1        2  1  3              109.47    No
A2        2  1  4              109.47    No
A3        2  1  5              109.47    No
A4        3  1  4              109.47    No
A5        3  1  5              109.47    No
A6        4  1  5              109.47    No
D1        4  1  3  2           -120.00   No
D2        5  1  3  2           120.00   No
D3        2  1  4  3           -120.00   No
D4        3  1  4  2           120.00   No
D5        5  1  4  2           -120.00   No
D6        5  1  4  3           120.00   No
D7        2  1  5  3           120.00   No
D8        2  1  5  4           -120.00   No
D9        3  1  5  2           -120.00   No
D10       3  1  5  4           120.00   No
D11       4  1  5  2           120.00   No
D12       4  1  5  3           -120.00   No

*****|

```

After the molecular structure has been updated, the program calculates the gradient as well as the size of the geometric step and determines whether the structural optimization converges :

	Force-RMS	Force-Max	Step-RMS	Step-Max
Conv. tolerance :	0.2000E-03	0.3000E-03	0.8000E-03	0.1200E-02
Current values :	0.8833E-02	0.2235E-01	0.2445E-01	0.5934E-01
Geom. converge :	No	No	No	No

The program considers the convergence of the structural optimization only when the current values of Force-RMS, Force-Max, Step-RMS, and Step-Max are less than the corresponding convergence limits (i.e., Geom. converge column is Yes). For this example, the structural optimization converges at step 5, when the output message not only contains the values of the convergence criteria, but also explicitly informs the user that the geometry optimization has

converged, and prints the converged molecular structure in Cartesian and internal coordinates, respectively.

```

Good Job, Geometry Optimization converged in      5 iterations!

Molecular Cartesian Coordinates (X,Y,Z) in Angstrom :
  C      -0.93557703      0.15971089      0.58828595
  H      -1.71170348     -0.52644336      0.21665897
  H      -1.26240747      1.20299703      0.46170050
  H      -0.72835075     -0.04452039      1.64971607
  Cl     0.56770184      -0.09691413     -0.35697029

                Force-RMS   Force-Max   Step-RMS   Step-Max
Conv. tolerance : 0.2000E-03  0.3000E-03  0.8000E-03  0.1200E-02
Current values  : 0.1736E-05  0.4355E-05  0.3555E-04  0.6607E-04
Geom. converge  :      Yes           Yes           Yes           Yes

Print Redundant internal coordinates of the converged geometry

|*****|
Redundant internal coordinates on Angstrom/Degree

Name      Definition      Value      Constraint
R1         1  2                1.1006     No
R2         1  3                1.1006     No
R3         1  4                1.1006     No
R4         1  5                1.7942     No
A1         2  1  3             110.04     No
A2         2  1  4             110.04     No
A3         2  1  5             108.89     No
A4         3  1  4             110.04     No
A5         3  1  5             108.89     No
A6         4  1  5             108.89     No
D1         4  1  3  2          -121.43     No
D2         5  1  3  2          119.28     No
D3         2  1  4  3          -121.43     No
D4         3  1  4  2          121.43     No
D5         5  1  4  2          -119.28     No
D6         5  1  4  3          119.29     No
D7         2  1  5  3          120.00     No
D8         2  1  5  4          -120.00     No
D9         3  1  5  2          -120.00     No
D10        3  1  5  4          120.00     No
D11        4  1  5  2          120.00     No
D12        4  1  5  3          -120.00     No

```

(continues on next page)

(continued from previous page)

```
*****|
```

Note that the convergence limits for the root-mean-square force and the rootmean-square step can be set here using the `tolgrad` and `tolstep` keywords, respectively, and the program automatically adjusts the convergence limits for the maximum force and the maximum step according to the set values; when using the DL-FIND library (see later), the energy convergence limit can also be specified by `tolene`. However, it is generally not recommended to adjust the convergence limits by the user.

At the same time, the program generates files with the suffix `.optgeom`, which contain the Cartesian coordinates of the converged molecular structure, but in Bohr units instead of Angstrom:

```
GEOM
  C      -0.7303234729      -2.0107211546      -0.0000057534
  H      -0.5801408002      -2.7816264533       1.9257943885
  H       0.4173171420      -3.1440530286      -1.3130342173
  H      -2.7178161476      -2.0052051760      -0.6126883555
  Cl     0.4272106261       1.1761889168      -0.0000021938
```

The `.optgeom` file can be converted to xyz format using the tool `optgeom2xyz.py` under `$BDFHOME/sbin/`, so that the optimized molecular structure can be viewed in any visualization software that supports xyz format. For example, if the file to be converted is named `filename.optgeom`, execute the following command line: (note that you must first set the environment variable `$BDFHOME`, or manually replace `$BDFHOME` in the following command with the path to the BDF folder)

```
$BDFHOME/sbin/optgeom2xyz.py filename
```

You can get `filename.xyz` in the current directory.

#### 4.12.2 Frequency calculation: Resonant frequencies and thermochemical quantities of CH<sub>3</sub>Cl in the equilibrium structure

After convergence of the structure optimization, the frequency analysis can be performed. Prepare the following input file:

```
$compass
title
  CH3Cl freq
basis
  def2-SV(P)
geometry
  C      -0.93557703      0.15971089      0.58828595
  H      -1.71170348      -0.52644336      0.21665897
```

(continues on next page)

(continued from previous page)

```

H          -1.26240747      1.20299703      0.46170050
H          -0.72835075     -0.04452039     1.64971607
Cl         0.56770184     -0.09691413    -0.35697029
end geometry
$end

$bdfopt
hess
  only
$end

$xuanyuan
$end

$scf
rks
dft
  b3lyp
$end

$resp
geom
$end

```

where the molecular structure is the converged structure obtained from the above structure optimization task. Note that we have added `hess only` to the BDFOPT module, where `hess` stands for computed (numerical) Hessian, and the meaning of `only` will be described in detail in the subsequent sections. The program perturbs each atom in the molecule in the positive x-axis, negative x-axis, positive y-axis, negative y-axis, positive z-axis, and negative z-axis directions, and calculates the gradient under the perturbed structure, e.g.

```

Displacing atom    1 (+x)...

### [bdf_single_point] ### nstate= 1
Do not allow rotation to standard orientation.

BDFOPT run - details of gradient calculations will be written
into .out.tmp file.

...

### JOB TYPE = SCF ###
E_tot= -499.84157717
Converge= YES

```

(continues on next page)

(continued from previous page)

```

### JOB TYPE = RESP_GSGRAD ###
Energy= -499.841577166026
  1      0.0005433780      -0.0000683370      -0.0000066851
  2     -0.0000516384      0.0000136326     -0.0000206081
  3     -0.0001360377      0.0000872513      0.0000990006
  4     -0.0003058645      0.0000115926     -0.0000775624
  5     -0.0000498284     -0.0000354732      0.0000023346

```

If the atomic number of the system is  $N$ , then a total of  $6N$  gradients have to be calculated. However, in practice the program also calculates the gradients of the unperturbed structure in order to allow the user to check whether the aforementioned structural optimization has indeed converged, so that the program actually calculates a total of  $6N+1$  gradients. Finally, the program obtains the Hessian of the system by the finite difference method. :

```

-----|
      Molecular Hessian - Numerical Hessian (BDFOPT)
      1          2          3          4          5  ↵
↵      6
  1  0.5443095266 -0.0744293569 -0.0000240515 -0.0527420800  0.0127361607  -
↵0.0209022664
  2 -0.0744293569  0.3693301504 -0.0000259750  0.0124150102 -0.0755387479  ↵
↵0.0935518380
  3 -0.0000240515 -0.0000259750  0.5717632089 -0.0213157291  0.0924260912  -
↵0.2929392390
  4 -0.0527420800  0.0124150102 -0.0213157291  0.0479752418 -0.0069459473  ↵
↵0.0239610358
  5  0.0127361607 -0.0755387479  0.0924260912 -0.0069459473  0.0867377886  -
↵0.0978524147
  6 -0.0209022664  0.0935518380 -0.2929392390  0.0239610358 -0.0978524147  ↵
↵0.3068416997
  7 -0.1367366097  0.0869338594  0.0987840786  0.0031968314 -0.0034098009  -
↵0.0016497426
  8  0.0869913627 -0.1185605401 -0.0945336434 -0.0070787068  0.0099076105  ↵
↵0.0045621064
  9  0.0986508197 -0.0953400774 -0.1659434327  0.0163191407 -0.0140134399  -
↵0.0166739137
 10 -0.3054590932  0.0111756577 -0.0774713107  0.0016297078  0.0019657599  -
↵0.0021771884
 11  0.0112823039 -0.0407134661  0.0021058508  0.0106623780  0.0018506067  ↵
↵0.0005120364
 12 -0.0775840113  0.0018141942 -0.0759448618 -0.0275602878  0.0006820252  -
↵0.0059830018

```

(continues on next page)

(continued from previous page)

13	-0.0486857506	-0.0362556088	0.0000641125	-0.0000787206	-0.0045253276	↵
↵0.0011289985						
14	-0.0360823429	-0.1334063062	0.0000148321	-0.0091074064	-0.0228930763	-
↵0.0010993076						
15	0.0001686252	0.0004961854	-0.0352553706	0.0084860406	0.0189117305	↵
↵0.0079690194						
	7	8	9	10	11	↵
↵	12					
1	-0.1367366097	0.0869913627	0.0986508197	-0.3054590932	0.0112823039	-
↵0.0775840113						
2	0.0869338594	-0.1185605401	-0.0953400774	0.0111756577	-0.0407134661	↵
↵0.0018141942						
3	0.0987840786	-0.0945336434	-0.1659434327	-0.0774713107	0.0021058508	-
↵0.0759448618						
4	0.0031968314	-0.0070787068	0.0163191407	0.0016297078	0.0106623780	-
↵0.0275602878						
5	-0.0034098009	0.0099076105	-0.0140134399	0.0019657599	0.0018506067	↵
↵0.0006820252						
6	-0.0016497426	0.0045621064	-0.0166739137	-0.0021771884	0.0005120364	-
↵0.0059830018						
7	0.1402213115	-0.0861503922	-0.1081442631	-0.0130805143	0.0143574755	↵
↵0.0192323598						
8	-0.0861503922	0.1322736798	0.1009922720	0.0016534140	0.0024111759	↵
↵0.0011733340						
9	-0.1081442631	0.1009922720	0.1688786678	-0.0038440081	0.0072277457	↵
↵0.0091535975						
10	-0.0130805143	0.0016534140	-0.0038440081	0.3186765202	-0.0079165663	↵
↵0.0838593213						
11	0.0143574755	0.0024111759	0.0072277457	-0.0079165663	0.0509206668	-
↵0.0029665370						
12	0.0192323598	0.0011733340	0.0091535975	0.0838593213	-0.0029665370	↵
↵0.0707430980						
13	0.0064620333	0.0044161973	-0.0031236007	-0.0026369496	-0.0283860480	↵
↵0.0017966445						
14	-0.0119743475	-0.0258901434	0.0013817613	-0.0066143965	-0.0145372292	-
↵0.0006143935						
15	-0.0078330845	-0.0126024853	0.0040383425	-0.0008566397	-0.0068931757	↵
↵0.0018028482						
	13	14	15			
1	-0.0486857506	-0.0360823429	0.0001686252			
2	-0.0362556088	-0.1334063062	0.0004961854			

(continues on next page)

(continued from previous page)

```

3  0.0000641125  0.0000148321 -0.0352553706
4 -0.0000787206 -0.0091074064  0.0084860406
5 -0.0045253276 -0.0228930763  0.0189117305
6  0.0011289985 -0.0010993076  0.0079690194
7  0.0064620333 -0.0119743475 -0.0078330845
8  0.0044161973 -0.0258901434 -0.0126024853
9 -0.0031236007  0.0013817613  0.0040383425
10 -0.0026369496 -0.0066143965 -0.0008566397
11 -0.0283860480 -0.0145372292 -0.0068931757
12  0.0017966445 -0.0006143935  0.0018028482
13  0.0450796910  0.0642866688  0.0000350066
14  0.0642866688  0.1954779468  0.0000894464
15  0.0000350066  0.0000894464  0.0213253497

```

where the 3N+1 (3N+2, 3N+3) rows correspond to the x (y, z) coordinates of the Nth atom and the 3N+1 (3N+2, 3N+3) columns do the same.

Next, the BDF calls the UniMoVib program for the calculation of frequencies and thermodynamic quantities. First are the results for the integrable representation to which the vibration belongs, the vibrational frequency, the approximate mass, the force constants and the simple positive modes :

```

*****
***  Properties of Normal Modes  ***
*****

Results of vibrations:
Normal frequencies (cm^-1), reduced masses (AMU), force constants (mDyn/A)

      1                               2
  ↪                               3                               4
      Irreps                        A1                               E
  ↪                               E
      Frequencies                    733.9170                    1020.5018
  ↪                               1021.2363
      Reduced masses                  7.2079                    1.1701
  ↪                               1.1699
      Force constants                 2.2875                    0.7179
  ↪                               0.7189
      Atom  ZA                       X           Y           Z           X           Y           Z
  ↪           X           Y           Z
      1  6          -0.21108  -0.57499  -0.00106  -0.04882  0.01679  0.10300
  ↪  0.09664  -0.03546  0.05161

```

(continues on next page)

(continued from previous page)

	2	1	-0.13918	-0.40351	0.04884	-0.06700	-0.59986	-0.13376
↔	-0.37214	-0.36766	-0.03443					
	3	1	-0.11370	-0.42014	-0.03047	0.26496	0.65294	-0.15254
↔	-0.28591	-0.18743	-0.15504					
	4	1	-0.19549	-0.38777	-0.01079	0.05490	-0.14087	-0.24770
↔	0.15594	0.73490	-0.07808					
	5	17	0.08533	0.23216	0.00014	0.00947	-0.00323	-0.01995
↔	-0.01869	0.00699	-0.01000					

Where each vibration mode is arranged in the order of vibration frequencies from smallest to largest, and the imaginary frequencies are ranked before all real frequencies, so only the first few frequencies need to be checked to know the number of imaginary frequencies. Next, the thermochemical analysis results are printed:

```

*****
***   Thermal Contributions to Energies   ***
*****

Molecular mass      :      49.987388   AMU
Electronic total energy :    -499.841576   Hartree
Scaling factor of Freq. :      1.000000
Tolerance of scaling :      0.000000   cm^-1
Rotational symmetry number:      3
The C3v point group is used to calculate rotational entropy.

Principal axes and moments of inertia in atomic units:

      Eigenvalues --
      X              1              2              3
      Y              11.700793     137.571621     137.571665
      Z              0.345094       0.938568      -0.000000
      Z              0.938568      -0.345094     -0.000000
      Z              0.000000       0.000000       1.000000

Rotational temperatures
↔ Kelvin              7.402388         0.629591         0.629591
Rot. constants A, B, C
↔ cm^-1              5.144924         0.437588         0.437588
↔ GHz                154.240933        13.118557        13.118553

# 1   Temperature =      298.15000 Kelvin      Pressure =      1.00000 Atm
=====
Thermal correction energies                                Hartree      kcal/mol

```

(continues on next page)

(continued from previous page)

Zero-point Energy	:	0.037519	23.543449
Thermal correction to Energy	:	0.040539	25.438450
Thermal correction to Enthalpy	:	0.041483	26.030936
Thermal correction to Gibbs Free Energy	:	0.014881	9.338203
Sum of electronic <b>and</b> zero-point Energies	:	-499.804057	
Sum of electronic <b>and</b> thermal Energies	:	-499.801038	
Sum of electronic <b>and</b> thermal Enthalpies	:	-499.800093	
Sum of electronic <b>and</b> thermal Free Energies:		-499.826695	
=====			

The user can read the zero-point energy, enthalpy, Gibbs free energy, etc. as needed. Note that all of the above thermodynamic quantities are obtained under each of the following assumptions:

1. a frequency correction factor of 1.0;
2. a temperature of 298.15 K;
3. a pressure of 1 atm;
4. the simplicity of the electronic state is 1.

If the user's calculation does not fall under the above scenario, it can be specified by a series of keywords, such as the following, which represents a frequency correction factor of 0.98, a temperature of 373.15 K, a pressure of 2 atm, and a simplicity of 2 for the electronic state.

```
$bdfopt
hess
  only
scale
  0.98
temp
  373.15
press
  2.0
ndeg
  2
$end
```

Of particular note is the simplicity of the electronic state, which is equal to the spin multiplet  $(2S+1)$  for non-relativistic or scalar relativistic calculations and where the electronic state does not have spatial simplicity; for electronic states with spatial simplicity, the spatial simplicity of the electronic state should also be multiplied by the spatial simplicity of the electronic state, which is the number of dimensions of the incommensurable representation to which the spatial part of the electronic wave function belongs. As for the relativistic calculations considering the spin-orbit coupling (e.g., TDDFT-SOC calculations), the spin multiplicity should be replaced by the simplicity of the corresponding spin state  $(2J+1)$ .

Sometimes, the frequency calculation is interrupted due to SCF non-convergence or other external reasons, when the calculation time can be saved by adding the `restarthess` keyword to the BDFOPT module for breakpoint continuation, e.g.

```
$bdfopt
hess
  only
restarthess
$end
```

It is also worth noting that structural optimization and frequency analysis (the so-called opt+freq calculation) can be implemented sequentially in the same BDF task, without the need to write two separate input files. For this purpose it is sufficient to change the input of the BDFOPT module to:

```
$bdfopt
solver
  1
hess
  final
$end
```

where `final` means that the numerical Hessian calculation is performed only after the successful completion of the structural optimization; if the structural optimization does not converge, the program simply quits with an error and does not perform the Hessian and frequency and thermodynamic quantities calculations. If the structural optimization does not converge, the program quits without performing the Hessian, frequency, and thermodynamic quantities calculations.

### 4.12.3 Transition State Optimization: Transition State Optimization and Frequency Calculation for HCN/HNC Heterogeneous Reactions

Prepare the following input file:

```
$compass
title
  HCN <-> HNC transition state
basis
  def2-SVP
geometry
C          0.00000000    0.00000000    0.00000000
N          0.00000000    0.00000000    1.14838000
H          1.58536000    0.00000000    1.14838000
end geometry
$end

$bdfopt
```

(continues on next page)

```
solver
  1
hess
  init+final
iopt
  10
$end

$xuanyuan
$end

$scf
rks
dft
  b3lyp
$end

$resp
geom
$end
```

where `iopt 10` indicates the optimized transition state.

Whether optimizing the minima structure or the transition state, the program must generate an initial Hessian prior to the first structural optimization step for use in subsequent structural optimization steps. In general, the initial Hessian should qualitatively match the exact Hessian under the initial structure, and in particular, the number of imaginary frequencies must be the same. This requirement is easily satisfied for the optimization of very small value points, and even the molecular mechanics level Hessian (the so-called “model Hessian”) can be made to match the exact Hessian qualitatively, so the program uses the model Hessian as the initial Hessian without calculating the exact Hessian. However, for transition state optimization, the model Hessian generally does not have an imaginary frequency, so the exact Hessian must be generated as the initial Hessian. `hess init+final` in the above input file means that both the initial Hessian is generated for the transition state optimization (this Hessian is not calculated on a structure with gradient 0), and the frequency and thermochemical quantities are not calculated on a gradient 0 structure. The `hess init+final` in the above input file means that the initial Hessian is generated for the transition state optimization (this Hessian is not calculated on the structure with gradient 0, the frequency and thermochemical quantities have no clear physical meaning, so only the Hessian is calculated without frequency analysis), and the Hessian is calculated again after the structure optimization converges to obtain the frequency analysis results. It is also possible to replace `init+final` with `init`, i.e., to generate only the initial Hessian and not to calculate the Hessian again after convergence of the structural optimization, but it is not recommended to omit the final keyword since transition state optimization (and indeed all structural optimization tasks) generally requires checking the number of virtual frequencies of the final converged structure.

The computed output is similar to that of the optimized minimal value point structure. The final frequency analysis shows that the converged structure has one and only one imaginary frequency ( $-1104\text{ cm}^{-1}$ ):

Results of vibrations:

Normal frequencies ( $\text{cm}^{-1}$ ), reduced masses (AMU), force constants (mDyn/A)

		1			2			
Irreps		A'			A'			
→	Frequencies		-1104.1414			2092.7239		
→	Reduced masses	2631.2601		1.1680			11.9757	
→	Force constants			-0.8389			30.9012	
→		4.3205						
	Atom	ZA	X	Y	Z	X	Y	Z
→		X	Y	Z				
→	1	6	0.04309	0.07860	0.00000	0.71560	0.09001	0.00000
→			-0.00274	-0.06631	0.00000			
→	2	7	0.03452	-0.06617	0.00000	-0.62958	-0.08802	0.00000
→			0.00688	-0.01481	0.00000			
→	3	1	-0.99304	-0.01621	0.00000	0.22954	0.15167	0.00000
→			-0.06313	0.99566	0.00000			

This means that the transition state is indeed found.

In the above calculation, the theoretical level of the initial Hessian is the same as the theoretical level of the transition state optimization. Since the initial Hessian only needs to be qualitatively correct, the initial Hessian can be calculated at a lower level and then the transition state can be optimized at a higher theoretical level. Taking the above example, if we want to calculate the initial Hessian at the HF/STO-3G level and optimize the transition state at the B3LYP/Def2-SVP level, we can follow the following steps.

(1) Prepare the following input file named `HCN-inithess.inp` :

```
$compass
title
  HCN <-> HNC transition state, initial Hessian
basis
  STO-3G
geometry
C          0.00000000    0.00000000    0.00000000
N          0.00000000    0.00000000    1.14838000
H          1.58536000    0.00000000    1.14838000
end geometry
$end

$bdfopt
```

(continues on next page)

(continued from previous page)

```
hess
  only
$end

$xuanyuan
$end

$scf
rhf
$end

$resp
geom
$end
```

- (2) Run the input file with BDF to obtain the Hessian file `HCN-inithess.hess` ;
- (3) Copy or rename `HCN-inithess.hess` to `HCN-optTS.hess` ;
- (4) Prepare the following input file, named `HCN-optTS.inp`:

```
$compass
title
  HCN <-> HNC transition state
basis
  def2-SVP
geometry
  C          0.00000000    0.00000000    0.00000000
  N          0.00000000    0.00000000    1.14838000
  H          1.58536000    0.00000000    1.14838000
end geometry
$end

$bdfopt
solver
  1
hess
  init+final
iopt
  10
readhess
$end

$xuanyuan
$end
```

(continues on next page)

(continued from previous page)

```
$scf
rks
dft
  b3lyp
$end

$resp
geom
$end
```

Where the keyword `readhess` means to read a hess file with the same name as the input file (i.e. HCN-optTS.hess) as the initial Hessian. Note that although this input file does not recalculate the initial Hessian, you still need to write `hess init+final` instead of `hess final`.

- (5) Just run the input file.

#### 4.12.4 Restricted Structural Optimization

BDF also supports restricting the value of one or more internal coordinates in structure optimization by adding the `constrain` keyword to the BDFOPT module. The first line after the `constrain` keyword is an integer (hereafter called N) indicating the total number of restrictions; lines 2 through N+1 define each restriction. For example, the following input indicates the distance between the 2nd atom and the 5th atom (these two atoms do not necessarily need to be chemically bonded to each other) to be restricted during structure optimization:

```
$bdfopt
solver
  1
constrain
  1
  2 5
$end
```

The following input indicates that the distance between the 1st atom and the 2nd atom is restricted during structure optimization, and also the bond angles formed by the 2nd, 5th and 10th atoms (again, no chemical bond is required between the 2nd and 5th atoms, or the 5th and 10th atoms):

```
$bdfopt
solver
  1
constrain
  2
  1 2
```

(continues on next page)

(continued from previous page)

```
2 5 10
$end
```

The following input indicates that the dihedral angles between the 5th, 10th, 15th, and 20th atoms are restricted during structure optimization, and also between the 10th, 15th, 20th, and 25th atoms:

```
$bdfopt
solver
1
constrain
2
5 10 15 20
10 15 20 25
$end
```

#### 4.12.5 Optimization of the conical intersection (CI) and the minimum energy intersection point (MECP)

The optimization of CIs and MECPs requires calling the DL-FIND external library, for which the following keywords are added to the input of the BDFOPT module.

```
solver
0
```

Accordingly, `solver 1` in the previous examples means that the optimization is performed using the BDF's own structural optimization code instead of DL-FIND. In principle, the optimization of minima and transition states can also be done with DL-FIND, but it is generally not as efficient as the BDF's own code, so DL-FIND should be called only for tasks that are not supported by the BDF's own code, such as CI and MECP optimization.

The following is an example input for CI optimization, which computes the tapered intersection of the T1 and T2 states of the ethylene:

```
#-----
# Gradient projection method for CI between T1 and T2 by TDFT BHHLYP/6-31G
#
$COMPASS
Title
  C2H4 Molecule test run
Basis
  6-31G
Geometry
```

(continues on next page)

(continued from previous page)

```
C          0.00107880  -0.00318153   1.43425054
C          0.00066030   0.00195132  -1.43437339
H          0.05960990  -0.89114967   0.84012371
H         -0.05830329   0.95445870   0.96064844
H          0.05950228   0.89180839  -0.84311032
H         -0.06267534  -0.95390169  -0.95768311
END geometry
nosym
$END

$bdfopt
imulti          #Optimize CI
  2
maxcycle        #Maximum number of optimization steps
  50
tolgrad         #Convergence criterion for root mean square gradients
  1.d-4
tolstep         #Convergence criterion for root mean square steps
  5.d-3
$end

$xuanyuan
$end

$SCF
RKS
charge
  0
spinmulti
  1
atomorb
DFT
  BHHLYP
$END

$tdft
imethod
  1
isf
  1
itda
  1
nroot
```

(continues on next page)

(continued from previous page)

```
5
idiag
1
istore
1
crit_e
1.d-8
crit_vec
1.d-6
lefteig
ialda
4
$end

$resp
geom
norder
1
method
2
iroot
1
nfiles
1
$end

$resp
geom
norder
1
method
2
iroot
2
nfiles
1
$end

$resp
iprt
1
QUAD
FNAC
```

(continues on next page)

(continued from previous page)

```
double
norder
  1
method
  2
nfiles
  1
pairs
  1
  1 1 1 1 1 2
$end
```

Note that this task requires not only the calculation of the gradients of the T1 and T2 states, but also the calculation of the non-adiabatic coupling vector between the T1 and T2 states (done by the last RESP module), see `tddft` for the relevant keywords `tddft`, which are not repeated here. In the input of the BDFOPT module, `imulti 2` represents the optimization CI. similar to the normal structural optimization task, the CI optimization outputs the gradient and step size convergence for each step, along with the energy convergence. For example, the output of the last optimization step of the above example is

```
Testing convergence  in cycle    6
  Energy  0.0000E+00 Target: 1.0000E-06 converged?  yes
  Max step 9.0855E-04 Target: 5.0000E-03 converged?  yes component    4
  RMS step 5.6602E-04 Target: 3.3333E-03 converged?  yes
  Max grad 5.5511E-05 Target: 1.0000E-04 converged?  yes component    1
  RMS grad 2.7645E-05 Target: 6.6667E-05 converged?  yes
Converged!
converged
```

Similar to the previous optimization tasks, the convergent CI structure is saved in In the `.optgeom` file, the coordinate unit is Bohr. Note that the value in the row of energy is always displayed as 0, which does not mean that the system energy remains unchanged during CI optimization, but because the optimization CI will not use the convergence of energy to judge whether it converges. For the same reason, the `tolene` keyword has no effect on CI optimization (and MECP optimization below).

The following is an example input file for optimizing MECP:

```
#-----
# Gradient projection method for ISC between S0 and T1 by BHHLYP/6-31G
#
$COMPASS
Title
  C2H4 Molecule test run
Basis
```

(continues on next page)

```
6-31G
Geometry
C      -0.00000141      0.00000353      0.72393424
C      0.00000417      -0.00000109     -0.72393515
H      0.73780975      -0.54421247      1.29907106
H     -0.73778145      0.54421417      1.29907329
H      0.73777374      0.54421576     -1.29907129
H     -0.73779427     -0.54423609     -1.29906321
END geometry
nosym
$END

$bdfopt
imulti
 2
maxcycle
 50
tolgrad
 1.d-4
tolstep
 5.d-3
noncouple
$end

$xuanyuan
$end

$SCF
RKS
charge
 0
spinmulti
 1
atomorb
DFT
BHLYP
$END

$resp
geom
norder
 1
method
```

(continues on next page)

(continued from previous page)

```
1
$end

$SCF
UKS
charge
0
spinmulti
3
atomorb
DFT
BHLLYP
$END

$resp
geom
norder
1
method
1
$end
```

where the `imulti 2` and `noncouple` keywords are specified to perform MECP optimization. Note that the MECP optimization task requires the calculation of only two states (here  $S_0$  and  $T$ ). The output of the MECP optimization task is similar to the CI optimization task and is not described here.

## 4.12.6 Geometric Optimization Frequently Asked Questions

### False frequency problem

Geometric optimization requires not only convergence of the structure (i.e., gradient and step size meet the convergence limits), but also the number of imaginary frequencies of the resulting structure to meet the expected value, i.e., 0 when optimizing the structure of the minima, 1 when optimizing the transition state, and higher order saddle points if the number of imaginary frequencies is greater than 1. When the actual number of virtual frequencies calculated does not match the expected value, the structure needs to be adjusted and re-optimized.

- When the actual calculated number of imaginary frequencies is less than the expected value, i.e., when the optimized transition state gets a structure with the number of imaginary frequencies of 0: this generally means that the obtained transition state structure is wrongly characterized, and the initial guess structure needs to be prepared again according to the common sense of chemistry.
- When the actual number of false frequencies is greater than the expected value, there are two possible cases: (1) the false frequencies are caused by the numerical error of the calculation, not the real existence. In this case, it can be

solved by increasing the grid point, decreasing the integration truncation threshold, decreasing various convergence thresholds (such as SCF convergence threshold, structural optimization convergence threshold, etc.), etc. (2) The system does have a false frequency. In this case, we should check the simple positive mode corresponding to the false frequency from the output file, and perturb the converged structure along the direction of the simple positive mode, and then use the perturbed structure as the first guess to re-optimize the structure.

- Note that it is impossible to determine whether a certain imaginary frequency is caused by numerical error from the frequency calculation results alone, but in general, the smaller the absolute value of the imaginary frequency, the more likely it is caused by numerical error, and vice versa, the more likely it is real.

## Symmetry problem

When the initial structure has a point group symmetry above group  $C_1$ , the structure optimization may break the point group symmetry, e.g., when optimizing an ammonia molecule with a planar structure with initial structure symmetry  $D_{3h}$ , the structure optimization may result in a conical structure with symmetry  $C_{3v}$ . By default the BDF forces the molecular point group symmetry to be maintained unless the system has a first order Jahn-Teller effect. If the user wants the BDF to break the symmetry of the molecules, one of the following approaches can be taken:

- Still optimize at high symmetry until convergence, and then calculate the frequencies. If false frequencies are present, perturb the molecular structure as in the previous subsection to eliminate them. If the molecule can be further reduced in energy by breaking the symmetry, then the perturbed molecular structure should be found to have reduced symmetry at this point, and the optimization should continue with that structure as the initial structure.
- If a subgroup of the molecular point group is specified in the COMPASS module, the program will only keep the subgroup symmetry unbroken. If a  $C_1$  group is specified, the program allows breaking the molecular symmetry in any way, maximizing the probability of obtaining a low-energy structure at the cost of not being able to use the point group symmetry to speed up the computation, resulting in increased computational effort.

## Geometric optimization does not converge

There are many factors that lead to the non-convergence of geometric optimization, including:

- The presence of numerical noise in the energy, gradients;
- The potential energy surface is too flat;
- The molecule has more than one stable wave function, and the wave function jumps back and forth between the various stable solutions during structural optimization, and does not converge stably and consistently to the the same solution;
- unreasonable molecular structure, e.g. wrong units of coordinates (e.g. the unit of coordinates is supposed to be Bohr, but the unit specified in the input file is Angstrom or vice versa), overdrawing or missing atoms, too close distances between non-bonded atoms, etc.

If the geometric optimization does not converge, or if there is no trend of convergence even though the maximum number of convergences has not been reached, after repeatedly checking that the three-dimensional structure of the

molecule is correct and reasonable, and that the wave function is not too close to the atom, then the geometry of the molecule should be optimized. After repeatedly checking that the three-dimensional structure of the molecule is correct and reasonable, and that the wave function converges normally, the following methods can be tried in turn:

- Use the last frame of the task that does not converge as the initial structure and start the optimization again. In addition to manually copying the structure coordinates of the last frame into the input file, a simpler way is to add the `restart` keyword to the COMPASS module, e.g.

```
$compass
title
  CH3Cl geomopt
basis
  def2-SV(P)
geometry
  C          2.67184328    0.03549756   -3.40353093
  H          2.05038141   -0.21545378   -2.56943947
  H          2.80438882    1.09651909   -3.44309081
  H          3.62454948   -0.43911916   -3.29403269
  Cl         1.90897396   -0.51627638   -4.89053325
end geometry
restart
$end
```

Suppose the input file is named `CH3Cl-opt.inp`, then the program automatically reads the coordinates in `CH3Cl-opt.optgeom` as the initial structure at this point (note that the program does not use the molecular coordinates in the `geometry` field at this point, but the molecular coordinates cannot be deleted). At first glance, this may seem to be the same as simply increasing the maximum number of iterations for geometry optimization, but in fact it often works better than simply increasing the maximum number of iterations, e.g., if the structure is reread after 100 steps of optimization and then re-optimized for 50 steps, the convergence probability is often higher than if the structure is re-read for 150 consecutive steps. This is because the program regenerates the initial Hessian when the structure is re-read to continue the optimization, thus avoiding the error accumulated by the quasi-Newton method of constructing the Hessian in multiple successive steps.

- Decreasing the optimization step length, or trust radius. This is done by using the `trust` keyword, e.g.

```
$bdfopt
solver
  1
trust
  0.05
$end
```

The default confidence radius is 0.3, so the new confidence radius should be less than 0.3. Note that the program will dynamically increase the confidence radius if it detects that the confidence radius is too small. To avoid this behavior,

the confidence radius can be set to a negative value, e.g.

```
$bdfopt
solver
  1
trust
  -0.05
$end
```

To avoid this behavior, the confidence radius can be set to a negative value, e.g., the initial confidence radius is set to 0.05, and the confidence radius is forbidden to exceed 0.05 during the entire structural optimization process.

- For transition state optimization, the `recalchess` keyword can be used to specify that the exact Hessian is recalculated at several steps.

```
$bdfopt
solver
  1
iopt
  10
hess
  init
recalchess
  10
$end
```

It indicates that the exact Hessian is recalculated every 10 steps of structural optimization, in addition to the exact Hessian calculated before structural optimization.

- The lattice points are increased and the convergence thresholds of the integration truncation and SCF, etc., are decreased to reduce the numerical errors. Note that this method only works when the structural optimization is almost convergence but not full convergence.

## 4.13 Solvation models

Solvation models are used to calculate the interaction between solutes and solvents, and are generally divided into two types: implicit solvent models (continuous medium models) and explicit solvent models. In BDF, for the continuous solvent model, we use ddCOSMO (domain-decomposition COSMO solvation model), and for the display solvent model we use the QM/MM method in combination with the pDynamo2.0 program package for the calculation.

### 4.13.1 Calculation of solvation effects

BDF currently supports the calculation of base-state solvation effects, including HF and DFT methods. The following are the input files for the solvation effect calculation of formaldehyde molecules file:

```
$COMPASS
Title
  ch2o Molecule test run
Basis
  6-31g
Geometry
  C    0.00000000    0.00000000   -0.54200000
  O    0.00000000    0.00000000    0.67700000
  H    0.00000000    0.93500000   -1.08200000
  H    0.00000000   -0.93500000   -1.08200000
END geometry
nosym
unit
  ang
$END

$xuanyuan
$END

$SCF
rks
dft
  b3lyp
solvent  #Solvation calculation
  water  #Specify the solvent
grid
  medium
$END
```

Where the solvent keyword is added to the SCF to indicate that a solvent effect calculation is to be performed, followed by a line where the solvent type, in this case `water`, can be entered. The list of solvent types supported in BDF is as follows:

Water	ε =78.3553	Butanone	ε =18.246
Acetonitrile	ε =35.688	ButanoNitrile	ε =24.291
Methanol	ε =32.613	ButylAmine	ε =4.6178
Ethanol	ε =24.852	ButylEthanoate	ε =4.9941
IsoQuinoline	ε =11.00	CarbonDiSulfide	ε =2.6105
Quinoline	ε =9.16	Cis-1,2-DiMethylCycloHexane	ε =2.06
Chloroform	ε =4.7113	Cis-Decalin	ε =2.2139
DiethylEther	ε =4.24	CycloHexanone	ε =15.619
Dichloromethane	ε =8.93	CycloPentane	ε =1.9608
DiChloroEthane	ε =10.125	CycloPentanol	ε =16.989
CarbonTetraChloride	ε =2.2280	CycloPentanone	ε =13.58
Benzene	ε =2.2706	Decalin-mixture	ε =2.196
Toluene	ε =2.3741	DiBromomEthane	ε =7.2273
ChloroBenzene	ε =5.6968	DiButylEther	ε =3.0473
NitroMethane	ε =36.562	DiEthylAmine	ε =3.5766
Heptane	ε =1.9113	DiEthylSulfide	ε =5.723
CycloHexane	ε =2.0165	DiIodoMethane	ε =5.32
Aniline	ε =6.8882	DiIsoPropylEther	ε =3.38
Acetone	ε =20.493	DiMethylDiSulfide	ε =9.6
TetraHydroFuran	ε =7.4257	DiPhenylEther	ε =3.73
DiMethylSulfoxide	ε =46.826	DiPropylAmine	ε =2.9112
Argon	ε =1.430	e-1,2-DiChloroEthene	ε =2.14
Krypton	ε =1.519	e-2-Pentene	ε =2.051
Xenon	ε =1.706	EthaneThiol	ε =6.667
n-Octanol	ε =9.8629	EthylBenzene	ε =2.4339
1,1,1-TriChloroEthane	ε =7.0826	EthylEthanoate	ε =5.9867
1,1,2-TriChloroEthane	ε =7.1937	EthylMethanoate	ε =8.3310
1,2,4-TriMethylBenzene	ε =2.3653	EthylPhenylEther	ε =4.1797
1,2-DiBromoEthane	ε =4.9313	FluoroBenzene	ε =5.42
1,2-EthaneDiol	ε =40.245	Formamide	ε =108.94
1,4-Dioxane	ε =2.2099	FormicAcid	ε =51.1
1-Bromo-2-MethylPropane	ε =7.7792	HexanoicAcid	ε =2.6
1-BromoOctane	ε =5.0244	IodoBenzene	ε =4.5470
1-BromoPentane	ε =6.269	IodoEthane	ε =7.6177
1-BromoPropane	ε =8.0496	IodoMethane	ε =6.8650
1-Butanol	ε =17.332	IsoPropylBenzene	ε =2.3712
1-ChloroHexane	ε =5.9491	m-Cresol	ε =12.44
1-ChloroPentane	ε =6.5022	Mesitylene	ε =2.2650
1-ChloroPropane	ε =8.3548	MethylBenzoate	ε =6.7367

continues on next page

Table 4.5 – continued from previous page

1-Decanol	$\epsilon = 7.5305$	MethylButanoate	$\epsilon = 5.5607$
1-FluoroOctane	$\epsilon = 3.89$	MethylCycloHexane	$\epsilon = 2.024$
1-Heptanol	$\epsilon = 11.321$	MethylEthanoate	$\epsilon = 6.8615$
1-Hexanol	$\epsilon = 12.51$	MethylMethanoate	$\epsilon = 8.8377$
1-Hexene	$\epsilon = 2.0717$	MethylPropanoate	$\epsilon = 6.0777$
1-Hexyne	$\epsilon = 2.615$	m-Xylene	$\epsilon = 2.3478$
1-IodoButane	$\epsilon = 6.173$	n-ButylBenzene	$\epsilon = 2.36$
1-IodoHexaDecane	$\epsilon = 3.5338$	n-Decane	$\epsilon = 1.9846$
1-IodoPentane	$\epsilon = 5.6973$	n-Dodecane	$\epsilon = 2.0060$
1-IodoPropane	$\epsilon = 6.9626$	n-Hexadecane	$\epsilon = 2.0402$
1-NitroPropane	$\epsilon = 23.73$	n-Hexane	$\epsilon = 1.8819$
1-Nonanol	$\epsilon = 8.5991$	NitroBenzene	$\epsilon = 34.809$
1-Pentanol	$\epsilon = 15.13$	NitroEthane	$\epsilon = 28.29$
1-Pentene	$\epsilon = 1.9905$	n-MethylAniline	$\epsilon = 5.96$
1-Propanol	$\epsilon = 20.524$	n-MethylFormamide-mixture	$\epsilon = 181.56$
2,2,2-TriFluoroEthanol	$\epsilon = 26.726$	n,n-DiMethylAcetamide	$\epsilon = 37.781$
2,2,4-TriMethylPentane	$\epsilon = 1.9358$	n,n-DiMethylFormamide	$\epsilon = 37.219$
2,4-DiMethylPentane	$\epsilon = 1.8939$	n-Nonane	$\epsilon = 1.9605$
2,4-DiMethylPyridine	$\epsilon = 9.4176$	n-Octane	$\epsilon = 1.9406$
2,6-DiMethylPyridine	$\epsilon = 7.1735$	n-Pentadecane	$\epsilon = 2.0333$
2-BromoPropane	$\epsilon = 9.3610$	n-Pentane	$\epsilon = 1.8371$
2-Butanol	$\epsilon = 15.944$	n-Undecane	$\epsilon = 1.9910$
2-ChloroButane	$\epsilon = 8.3930$	o-ChloroToluene	$\epsilon = 4.6331$
2-Heptanone	$\epsilon = 11.658$	o-Cresol	$\epsilon = 6.76$
2-Hexanone	$\epsilon = 14.136$	o-DiChloroBenzene	$\epsilon = 9.9949$
2-MethoxyEthanol	$\epsilon = 17.2$	o-NitroToluene	$\epsilon = 25.669$
2-Methyl-1-Propanol	$\epsilon = 16.777$	o-Xylene	$\epsilon = 2.5454$
2-Methyl-2-Propanol	$\epsilon = 12.47$	Pentanal	$\epsilon = 10.0$
2-MethylPentane	$\epsilon = 1.89$	PentanoicAcid	$\epsilon = 2.6924$
2-MethylPyridine	$\epsilon = 9.9533$	PentylAmine	$\epsilon = 4.2010$
2-NitroPropane	$\epsilon = 25.654$	PentylEthanoate	$\epsilon = 4.7297$
2-Octanone	$\epsilon = 9.4678$	PerFluoroBenzene	$\epsilon = 2.029$
2-Pentanone	$\epsilon = 15.2$	p-IsoPropylToluene	$\epsilon = 2.2322$
2-Propanol	$\epsilon = 19.264$	Propanal	$\epsilon = 18.5$
2-Propen-1-ol	$\epsilon = 19.011$	PropanoicAcid	$\epsilon = 3.44$
3-MethylPyridine	$\epsilon = 11.645$	PropanoNitrile	$\epsilon = 29.324$
3-Pentanone	$\epsilon = 16.78$	PropylAmine	$\epsilon = 4.9912$
4-Heptanone	$\epsilon = 12.257$	PropylEthanoate	$\epsilon = 5.5205$

continues on next page

Table 4.5 – continued from previous page

4-Methyl-2-Pentanone	$\epsilon = 12.887$	p-Xylene	$\epsilon = 2.2705$
4-MethylPyridine	$\epsilon = 11.957$	Pyridine	$\epsilon = 12.978$
5-Nonanone	$\epsilon = 10.6$	sec-ButylBenzene	$\epsilon = 2.3446$
AceticAcid	$\epsilon = 6.2528$	tert-ButylBenzene	$\epsilon = 2.3447$
AcetoPhenone	$\epsilon = 17.44$	TetraChloroEthene	$\epsilon = 2.268$
a-ChloroToluene	$\epsilon = 6.7175$	TetraHydroThiophene-s,s-dioxide	$\epsilon = 43.962$
Anisole	$\epsilon = 4.2247$	Tetralin	$\epsilon = 2.771$
Benzaldehyde	$\epsilon = 18.220$	Thiophene	$\epsilon = 2.7270$
BenzoNitrile	$\epsilon = 25.592$	Thiophenol	$\epsilon = 4.2728$
BenzylAlcohol	$\epsilon = 12.457$	trans-Decalin	$\epsilon = 2.1781$
BromoBenzene	$\epsilon = 5.3954$	TriButylPhosphate	$\epsilon = 8.1781$
BromoEthane	$\epsilon = 9.01$	TriChloroEthene	$\epsilon = 3.422$
Bromoform	$\epsilon = 4.2488$	TriEthylAmine	$\epsilon = 2.3832$
Butanal	$\epsilon = 13.45$	Xylene-mixture	$\epsilon = 2.3879$
ButanoicAcid	$\epsilon = 2.9931$	z-1,2-DiChloroEthene	$\epsilon = 9.2$

### 4.13.2 Entering the dielectric constant

For solvents that are not in the table, you can enter the dielectric constant. The format is as follows:

```
solvent
  user  #User-specified
dielectric
  78.3553  #Enter the dielectric constant
```

**Note:** Solvation effects are currently only supported for energy calculations, and gradient calculations will be completed in the near future.

### 4.13.3 Excited State Solvation Effect

The excited state solvation effect can be calculated using a combination of explicit and implicit solvents. Take an aqueous solution as an example, due to the solute molecule's HOMO and LUMO orbitals may diffuse into the first hydration layer, so the excited state calculation can include the water molecules in the first hydration layer in the TDDFT calculation region, while the rest is treated with an implicit solvent.

Take sinapic acid as an example. To determine the first hydration layer of the solute molecule, the Amber program can be used to simulate the molecular dynamics of the mustard acid molecule in a small box of water. After the system is equilibrated, the distribution of water molecules around the solute molecule can be analyzed to determine the first hydration layer. Of course, it is also possible to select a multi-frame structure for calculation and then take an average.

The selection of molecules in the hydration layer can be done using the VMD program. Assuming that the input is a pdb file, the first hydration layer molecule can be selected on the command line and saved as a pdb file. The command is as follows:

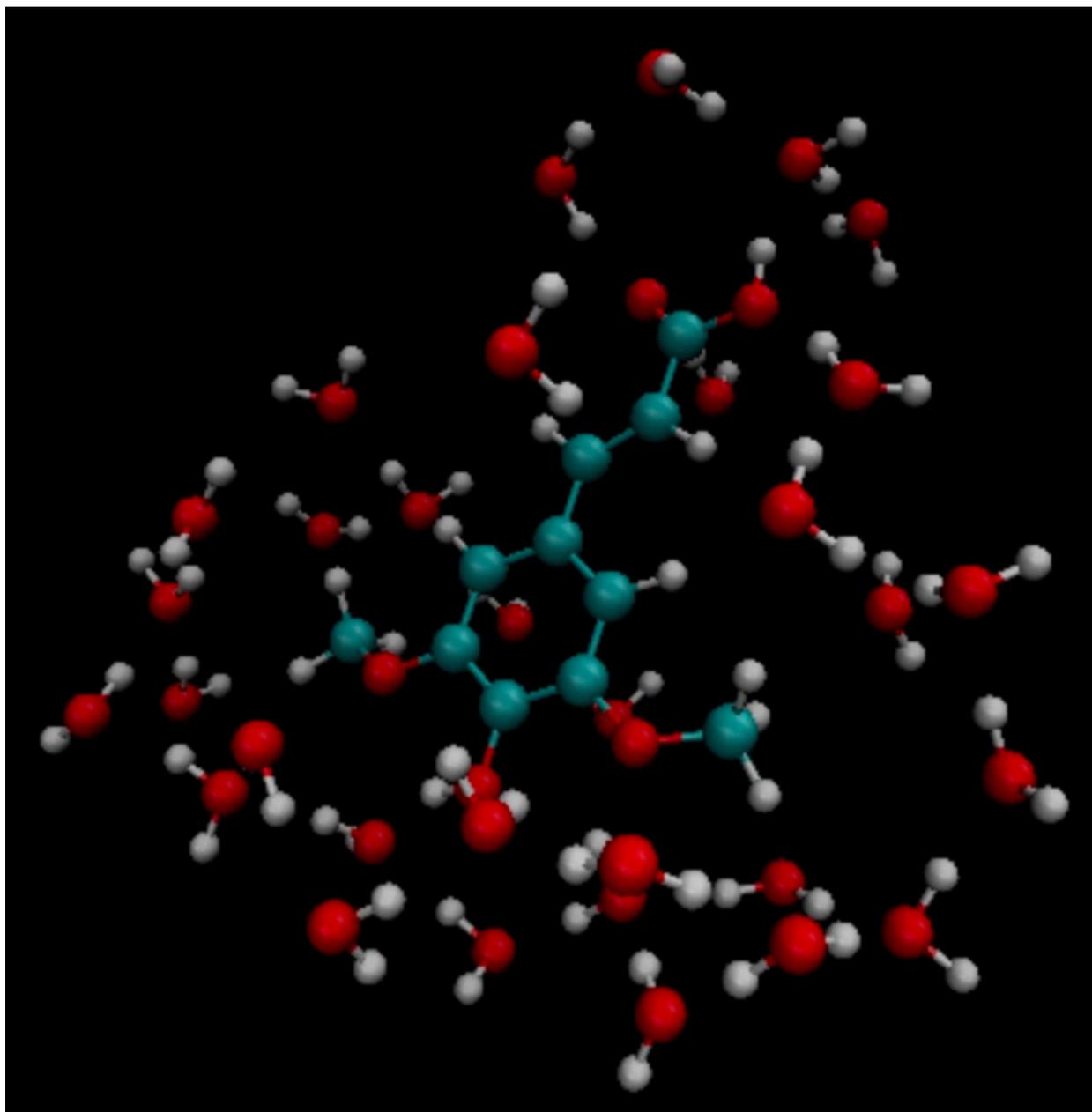
```
atomselect top "same resid as (within 3.5 of not water)" # Select the first_
↪hydration layer
atomselect0 writepdb sa.pdb #Solute molecules and first hydrated_
↪layers are preserved in pdb files
```

In the above example, all water molecules within 3.5 Å of the solute molecule are selected, and the entire molecule is selected as long as one of the three atoms of the water molecule is within the truncation range. The selection results are shown in the figure:

Based on the coordinate information in the sa.pdb file, the TDDFT calculation is performed with the following input file:

```
$COMPASS
Title
  SA Molecule test run
Basis
  6-31g
Geometry
C      14.983  14.539  6.274
C      14.515  14.183  7.629
C      13.251  14.233  8.118
C      12.774  13.868  9.480
C      11.429  14.087  9.838
C      10.961  13.725  11.118
O       9.666  13.973  11.525
C       8.553  14.050  10.621
C      11.836  13.125  12.041
O      11.364  12.722  13.262
C      13.184  12.919  11.700
O      14.021  12.342  12.636
C      15.284  11.744  12.293
C      13.648  13.297  10.427
O      14.270  14.853  5.341
O      16.307  14.468  6.130
H      15.310  13.847  8.286
H      12.474  14.613  7.454
H      10.754  14.550  9.127
H       7.627  14.202  11.188
H       8.673  14.888  9.924
H       8.457  13.118  10.054
H      10.366  12.712  13.206
```

(continues on next page)



(continued from previous page)

H	15.725	11.272	13.177
H	15.144	10.973	11.525
H	15.985	12.500	11.922
H	14.687	13.129	10.174
H	16.438	14.756	5.181
O	18.736	9.803	12.472
H	18.779	10.597	11.888
H	19.417	10.074	13.139
O	18.022	14.021	8.274
H	17.547	14.250	7.452
H	18.614	13.310	7.941
O	8.888	16.439	7.042
H	9.682	16.973	6.797
H	8.217	17.162	7.048
O	4.019	14.176	11.140
H	4.032	13.572	10.360
H	4.752	14.783	10.885
O	16.970	8.986	14.331
H	17.578	9.273	13.606
H	17.497	8.225	14.676
O	8.133	17.541	10.454
H	8.419	17.716	11.386
H	8.936	17.880	9.990
O	8.639	12.198	13.660
H	7.777	11.857	13.323
H	8.413	13.155	13.731
O	13.766	11.972	4.742
H	13.858	12.934	4.618
H	13.712	11.679	3.799
O	10.264	16.103	14.305
H	9.444	15.605	14.054
H	10.527	15.554	15.084
O	13.269	16.802	3.701
H	13.513	16.077	4.325
H	14.141	17.264	3.657
O	13.286	14.138	14.908
H	13.185	14.974	14.393
H	13.003	13.492	14.228
O	16.694	11.449	15.608
H	15.780	11.262	15.969
H	16.838	10.579	15.161
O	7.858	14.828	14.050
H	7.208	15.473	13.691

(continues on next page)

(continued from previous page)

H	7.322	14.462	14.795
O	15.961	17.544	3.706
H	16.342	16.631	3.627
H	16.502	17.866	4.462
O	10.940	14.245	16.302
H	10.828	13.277	16.477
H	11.870	14.226	15.967
O	12.686	10.250	14.079
H	11.731	10.151	14.318
H	12.629	11.070	13.541
O	9.429	11.239	8.483
H	8.927	10.817	7.750
H	9.237	12.182	8.295
O	17.151	15.141	3.699
H	17.124	14.305	3.168
H	18.133	15.245	3.766
O	17.065	10.633	9.634
H	16.918	10.557	8.674
H	17.024	9.698	9.909
O	17.536	14.457	10.874
H	18.014	13.627	11.089
H	17.683	14.460	9.890
O	5.836	16.609	13.299
H	4.877	16.500	13.549
H	5.760	16.376	12.342
O	19.014	12.008	10.822
H	18.249	11.634	10.308
H	19.749	11.655	10.256
O	15.861	14.137	15.750
H	14.900	13.990	15.574
H	16.185	13.214	15.645
O	11.084	9.639	10.009
H	11.641	9.480	9.213
H	10.452	10.296	9.627
O	14.234	10.787	16.235
H	13.668	10.623	15.444
H	13.663	10.376	16.925
O	14.488	8.506	13.105
H	13.870	9.136	13.550
H	15.301	8.683	13.628
O	14.899	17.658	9.746
H	15.674	18.005	9.236
H	15.210	16.754	9.926

(continues on next page)

(continued from previous page)

```
O      8.725  13.791  7.422
H      9.237  13.488  6.631
H      8.845  14.770  7.309
O     10.084  10.156  14.803
H      9.498  10.821  14.366
H     10.215  10.613  15.669
O      5.806  16.161  10.582
H      5.389  16.831  9.993
H      6.747  16.470  10.509
O      6.028  13.931  7.206
H      5.971  14.900  7.257
H      6.999  13.804  7.336
O     17.072  12.787  2.438
H     16.281  12.594  1.885
H     17.062  11.978  3.013

END geometry
nosym
mpec+cosx
$END

$xuanyuan
$end

$SCF
rks
dft
  b3lyp
solvent
  water
grid
  medium
$END

# input for tddft
$tddft
iroot  # Calculate 1 root for each irrep. By default, 10 roots are calculated
  1    # for each irrep
memjkop # maxium memeory for Coulomb and Exchange operator. 1024 MW (Mega Words)
  1024
$end
```

## 4.14 Point charge model

The BDF supports the calculation of atomic charges in the MM region as point charge input. The point charge is given as input to the \$BDFTASK.extcharge file with the same name as the calculation task as input, in the following format:

```
$COMPASS
Title
water molecule in background of external charges
Basis
  6-31g
Geometry
O   0.000000   0.000000   0.106830
H   0.000000   0.785178  -0.427319
H   0.000000  -0.785178  -0.427319
End Geometry
Extcharge  #Indicates that an input point charge is required
  point    #Indicates that the input charge type is a point charge
$END

$XUANYUAN
$END

$SCF
RHF
$END
```

The point charge input file (file name h2o.extcharge) is as follows:

```
External charge, Point charge  #The first line is the title and description line
6                               #The number of point charges to enter
C1   -0.732879   0.000000   5.000000   0.114039
C2    0.366440   0.000000   5.780843  -0.456155
C3    0.366440   0.000000   4.219157  -0.456155
C4   -0.732879   0.000000  10.000000   0.114039
C5    0.366440   0.000000  10.78084   -0.456155
C6    0.366440   0.000000   9.219157  -0.456155
```

The default input format for point charges is: atomic label, charge, and coordinates (x y z); the coordinates are in angstroms by default. Coordinate input units can also be Bohr, and the input format is as follows

```
External charge, Point charge  # title line
6   Bohr                       # Unit: Bohr
C1   -0.732879   0.000000   5.000000   0.114039
```

(continues on next page)

(continued from previous page)

# ... #

## 4.15 Wave function analysis and single-electron properties

The wave function analyses supported by the BDF are: Mulliken and Lowdin Boolean analysis, including net atomic charge and spin density.

The single-electron properties supported by the BDF are:

- SCF: dipole moment, polarizability\*, hyperpolarizability\*, Musburger spectrum (effective contact density), NMR\*\*
- TDDFT: dipole moment of excited states\*, vibronic intensity of fluorescence absorption spectra, vibronic intensity of phosphorescence absorption spectra

\* Calculated in `resp` module. \*\* Calculated in `nmr` module

More wave function analysis and single electron properties can be done by generating data files in molden format in the `scf` module with a third party program This is done. Example input.

```
$scf
rks
dft
  b3lyp
molden
$end
```

The standard molden format only supports spdfg-type Gaussian basis functions, but has been extended to h-functions in BDF

### 4.15.1 Effective contact density

The calculation of the effective contact density (ED) requires consideration of both relativistic effects (X2C Hamiltonian in BDF, identified by `heff = 21, 22, or 23`) and finite size nuclei (`nuclear = 1`). An example input at the `sf-X2CAU/B3LYP` level is as follows

```
$xuanyuan
heff
  23
nuclear
  1
$end
```

(continues on next page)

```
$scf
rks
dft
  b3lyp
grid
  sg1
reled
  20
$end
```

where `reled` calls the calculation of the relativistic property ED. 20 means that the ED is not calculated for light elements with atomic number less than 20, thus saving calculation time. For density generalization calculations, the value of ED is sensitive to the integration grid point and `ultra fine` or more precise `sg1` is recommended.

The ED requires a special treatment of the basis group, see 穆斯堡尔谱.

### 4.15.2 molden2aim

Download: <https://github.com/zorkzou/Molden2AIM>

The purpose of `molden2aim` is to convert molden files generated by BDF into `wfn`, `wfx`, or NBO-47 format data files for various various analyses. It supports `spdfg` type Gaussian basis functions as well as ECP.

### 4.15.3 Multiwfn

Download: <http://sobereva.com/multiwfn/>

`Multiwfn` is a powerful wave function analysis program. Molden data files (supporting `spdfg` basis functions and pseudopotentials) generated by BDF or `wfn`, `wfx` data files transformed by `molden2aim`, `Multiwfn` can be used for various wave function analyses, such as electron density topology analysis (also known as *quantum theory of atoms in molecules*; QTAIM), electron localization functions (ELF), booster analysis, bond level analysis, atomic charge analysis, etc., and Images of molecular orbitals, electron density and various real space functions can also be plotted. See the `Multiwfn` user manual for details.

### 4.15.4 NBO analysis

BDF does not currently contain an interface to NBO (<https://nbo7.chem.wisc.edu/>), but it is possible to use `molden2aim` to convert BDF-generated molden files (supporting `spdfg` basis functions and pseudopotentials) into NBO-47 format data files, and then use the NBO standalone program `gennbo.exe` to perform NBO analysis.

For wave functions of the RHF/RKS and UHF/UKS types (i.e., the MO occupation number can only be 0, 1, and 2), NBO can calculate the “Second Order Perturbation Theory Analysis”, which requires the presence of the Fock matrix in the 47 file. To do this, set `nbopro=1` in the `molden2aim` configuration file `m2a.ini`.

## **SIMPLE INPUT**

This chapter will introduce BDF simple input control keywords.

### **5.1 简洁输入关键词**

#### **5.1.1 方法/泛函/基组、泛函/基组、方法/基组必选参数**

简洁输入模式的必选参数用于设定计算方法、DFT/TDDFT 计算的泛函和基组等，当前支持的计算方法有：

计算方法	功能
HF	Hatree-Fock
RHF	Restricted Hatree-Fock
UHF	Unrestricted Hartree-Fock
ROHF	Restricted open-shell Hatree-Fock
KS	Kohn-Sham
RKS	Restricted Kohn-Sham
UKS	Unrestricted Kohn-Sham
TDDFT	Time-dependent density functional theory
TDA	Tamm-Dancoff Approximation
X-TDDFT	Extended spin-adapted TDDFT
X-TDA	Extended spin-adapted TDA
TDDFT-SOC	TDDFT with spin-orbit coupling
TDA-SOC	TDA with spin-orbit coupling
X-TDDFT-SOC	Extended spin-adapted TDDFT with spin-orbit coupling
X-TDA-SOC	Extended TDA with SOC
TDDFT-NAC	TDDFT with non-adiabatic coupling
TDA-NAC	TDA with non-adiabatic coupling
X-TDDFT-NAC	X-TDDFT with non-adiabatic coupling
X-TDA-NAC	X-TDA with non-adiabatic coupling
MP2	Moller-Plesset second order perturbation theory
RI-MP2	MP2 using Resolution of Identity

## 哈密顿和自旋轨道耦合

### 5.1.2 *hamilton* 参数类型: 字符串, 可选参数

设定计算的相对论哈密顿

默认值: nonrel, 使用相对论基组时默认为 sf-X2C

可选值: sf-X2C, sf-X2C-AXR, sf-X2C-AU

### 5.1.3 SOC 参数类型: Bool, 可选参数

要求进行自旋轨道耦合 (SOC) 计算, 并设置相应的 SOC 算符。如果计算方法是 TDDFT, 进行基于 TDDFT 的 SOC 计算; 方法是 TDA, 则进行基于 TDA 的 SOC 计算。

默认值: DKH1e+mf1c

可选值: DKH1e+mf1c, DKH1e, BP; 全电子用 DKH1e+mf1c, 相对论有效势用 BP 算符。

**Note:**

- 默认原则: 如果指定哈密顿, BDF 将根据基函数选择合适的哈密顿。对于考虑了相对论效应的全电子基组或非相对论全电子基组, 标量项采用 **sf-X2C** 哈密顿, 自旋轨道耦合算符采用 **DHK1e+mf1c**, 用户可以强行设置为 **DHK1e**, 对于轻元素会有较大的误差。对于相对论有效势及基组, 有效势已经考虑了相对论效应, 无需设置哈密顿, SOC 算符默认为 BP。
- 如果用户的输入为 *TDDFT/泛函/基组 SOC*, 使用了 SOC 关键词, 等价于在方法中设置 *X-TDDFT/泛函/基组*, 哈密顿和 SOC 算符将按照默认原则设置。

坐标单位, 电荷和自旋多重度

**5.1.4 unit 参数类型: 字符串, 可选参数**

原子坐标单位

默认值: angstrom

可选值: angstrom, Bohr

**5.1.5 spinmulti 参数类型: 整数, 可选参数**

自旋多重度,  $2S+1$

默认值: 偶数电子体系, 1; 奇数电子体系, 2

**5.1.6 charge 参数类型: 整数, 可选参数**

电荷数

默认值: 0

自旋匹配的 TDDFT 和 TDA

**5.1.7 SpinAdapt**

对自旋匹配的 TDDFT 或 TDA 进行设置。*TDDFT/泛函/基组 SpinAdapt* 等价于 *X-TDDFT/泛函/基组* 或 *X-TDA*。只对开壳层体系有意义,

非绝热耦合

### 5.1.8 *NAC* 参数类型: Bool, 可选参数

基于含时密度泛函 (TDDFT) 的非绝热耦合计算 (NAC)

默认值: False

势能面与结构优化

### 5.1.9 *opt* 参数类型: Bool, 可选参数

稳定点分子几何结构优化。

默认值: False

### 5.1.10 *opt+freq* 参数类型: Bool, 可选参数

稳定点分子几何结构优化, 随后进行频率计算。

默认值: False

### 5.1.11 *ts+freq* 参数类型: Bool, 可选参数

过渡态优化, 随后进行频率计算。

默认值: False

### 5.1.12 *freq* 参数类型: Bool, 可选参数

频率计算。

默认值: False

### 5.1.13 *scan* 参数类型: Bool, 可选参数

分子势能面扫描, 需配合内坐标输入使用。

默认值: False

#### 5.1.14 *scan+opt* 参数类型: Bool, 可选参数

分子势能面柔性扫描, 即固定某些内坐标参量, 优化其它坐标参量, 需配合内坐标输入使用。

默认值: False

加速算法

#### 5.1.15 *MPEC+COSX* 参数类型: Bool, 可选参数

利用 *Multipole expansion of Coulomb potential* (MPEC) 及 *Chain-Of-Sphere Exchange* (COSX) 加速 SCF、TDDFT 能量及梯度计算。

默认值: False

#### 5.1.16 *RI* 参数类型: Bool, 可选参数

利用 RI 加速 SCF、TDDFT 或 MP2 计算, 需要配合 RI 基组使用。

默认值: False

---

#### Tip:

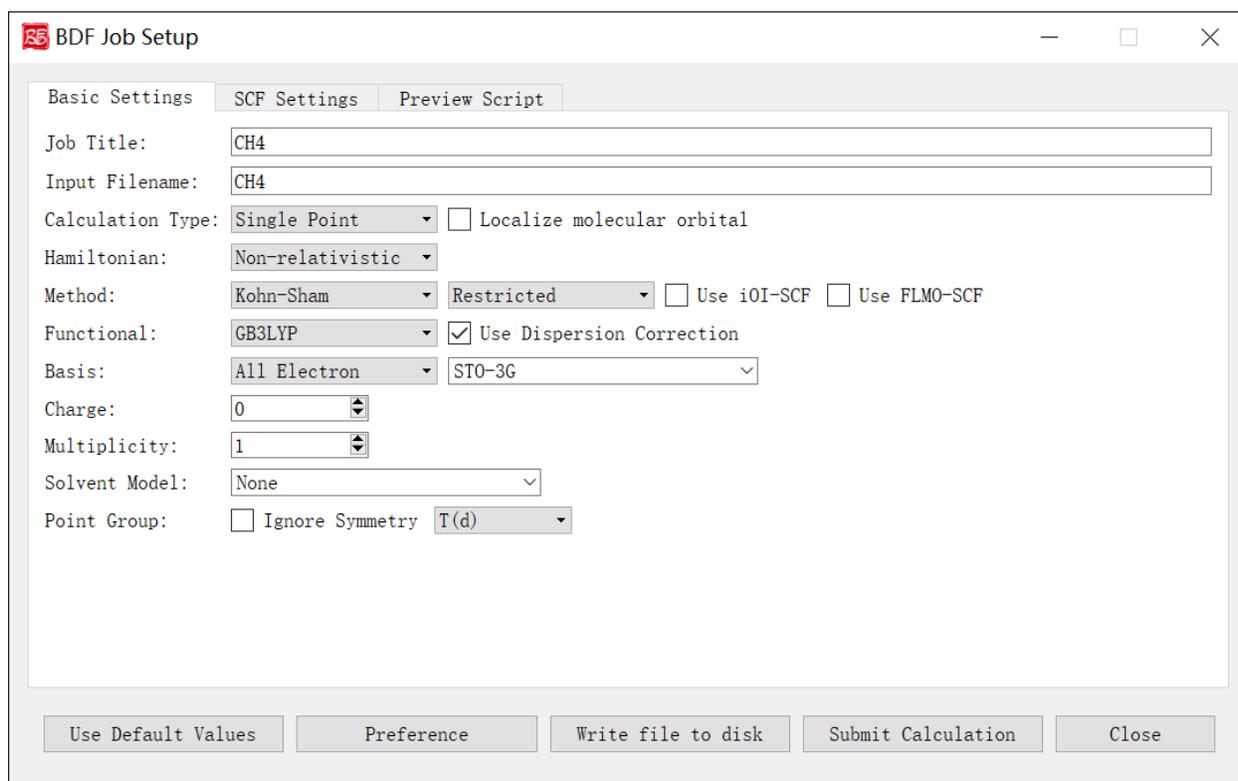
- RI 在 BDF 中主要用于加速 MP2 计算, SCF 和 TDDFT 均可用 MPEC+COSX 方法, 该方法是 BDF 特有的加速算法, 与 RI 算法的精度相近, 但不需要辅助基组。
-



## 图形界面

本章将介绍 BDF 图形界面的功能和使用的注意事项。

## 6.1 初始参数界面



上图是启动 BDF 任务提交界面的初始参数界面，这儿我们以甲烷分子  $\text{CH}_4$  作为计算对象。

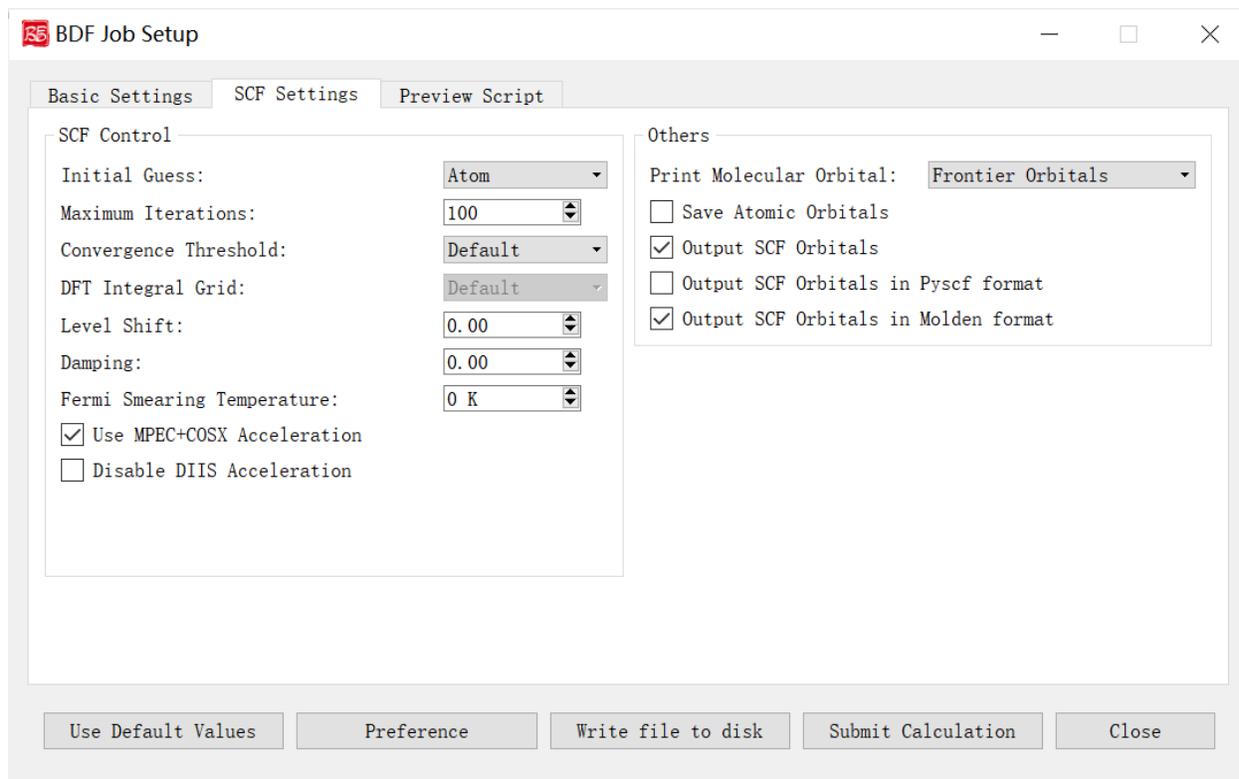
当我们导入分子结构之后，程序首先会识别分子的化学式、电荷数、对称点群等信息。并且根据这些信息来完成一些默认的设置，如 Job Title 输入框的内容、Input Filename 的输入框内容、Multiplicity 的输入框内容、Point Group 的下拉框选项等。

下面我们对上图的图形界面的控件及其功能一一说明：

1. **Job Title**: 计算任务的标题说明, 一般用于备注当前计算的任务类型、分子体系和参数等等。这部分内容并不影响 BDF 的计算, 但是为了养成良好的习惯, 也方便将来查询, 用户可根据自己的习惯进行一定的备注。例如对于单点能计算, 笔者习惯将其备注成 SP\_CH4\_B3LYP-D3/6-31g\*\*。
2. **Input Filename**: 计算任务的文件名, 该输入框限制只能输入数字、字母、下划线 ()、加号 (+) 和减号 (-)。例如当前输入为 CH4, 则软件会自动保存输入文件为 CH4.inp。
3. **Calculation Type**: 计算任务的类型, 当前的图形界面支持 **Single Point (单点能计算)**、**Optimization (结构优化)**、**Frequency (频率计算)**、**Opt+Freq (结构优化 + 频率计算)**、**TDDFT (激发态计算)**、**TDDFT-SOC (自旋轨道耦合计算)**、**TDDFT-NAC (非绝热耦合计算)**、**TDDFT-OPT (激发态结构优化)**、**TDDFT-Freq (激发态频率计算)**、**TDDFT-OPT+Freq (激发态结构优化 + 频率计算)**、**NMR (核磁共振)** 共计 11 种计算类型。**Localize molecular orbital** 复选框表示设置分子轨道局域化参数。每一种计算类型都有相关的设置参数, 选择对应的计算任务类型之后, 程序会激活相应的控制模块。
4. **Hamiltonian**: 计算任务考虑相对论效应时采用的哈密顿量, BDF 支持多种相对论效应的计算, 具体细节参见 *BDF 软件简介*, 考虑到大部分用户的使用习惯和理论基础, 当前的图形界面只提供了 **Non-relativistic (非相对论)**、**ECP (赝势)**、**sf-X2C** 和 **X2C** 四种选项。其中 **X2C** 尚未在当前发布的 BDF 版本中支持, 故界面上不允许用户选择, 在未来 BDF 软件更新相关功能之后才会解锁。**Non-relativistic** 选项表示不考虑相对论效应, 此时的哈密顿量也就是我们在教科书上见到的薛定谔方程的形式。**ECP** 选项表示通过赝势基组的形式考虑相对论效应, **sf-X2C** 表示用无自旋的 X2C 二分量子哈密顿考虑标量相对论效应。
5. **Method** 后面有多个界面控件, 第一个下拉选框是计算方法, 支持 **Hartree-Fock**、**Korn-Sharm** 和 **MP2** 三种方法, 其中 **MP2** 方法当前只支持单点能的计算。第二个下拉选框是计算体系的壳层设置, 包括了 **Restricted (闭壳层)**、**Unrestricted (开壳层)** 和 **Restricted Open (限制性开壳层)** 三种选择, 当体系的自旋多重度等于 1 的时候, 默认选择 **Restricted**, 如果用户要做自旋极化单重态的计算, 可以选择 **Unrestricted**, 并读取三重态计算的波函数作为初猜。**Use iOI-SCF** 和 **Use FLMO-SCF** 这两个复选框可以选择 BDF 独有的通过分子分片方法进行大体系自洽场计算的方法, 具体的技术细节可以参考 *FLMOMethod*。由于 BDF 软件现在只支持用 iOI 或者 FLMO 方法做单点的自洽场计算, 如果用户要做大体系的结构优化, 可以先采用 iOI 或者 FLMO 方法先做自洽场计算, 然后再读取这一步得到的波函数作为初猜, 用常规的 HF 方法或者 DFT 方法进行后续的梯度和结构优化计算。
6. **Functional**: Korn-Sharm 计算采用的具体的泛函类型, 只有在前面的 Method 下拉选项为 Korn-Sharm 的时候才会显示这个界面选项。由于 BDF 内置的泛函对梯度、激发态响应等支持程度各不相同, 为了防止用户选择的泛函和具体的计算任务不匹配, 软件会根据用户前面选择的任务类型, 只列出支持对应计算的泛函选项。如果该泛函支持 D3 色散校正, 则还会出现 **Use Dispersion Correction** 的复选框, 如果勾选该复选框, 则表示使用 D3 色散校正。
7. **Basis**: 计算任务的基组, 软件支持全电子基组、赝势基组和自定义混合基组三种选择。如果前面的相对论效应选择的了 sf-X2C 等全电子相对论计算方法, 则此处只能选择专门为相对论计算优化的收缩基组或这些基组的混合形式。**PS: def2** 系列的基组对于轻元素采用的是全电子基组, 而对于重元素采用的是赝势基组, 这儿根据用户的使用习惯将其放在了非相对论的全电子基组类型中。
8. **Charge**: 计算体系的净电荷数目。
9. **Multiplicity**: 计算体系的自旋多重度 (2S+1)。

10. Solvent Model: 溶剂化模型。
11. Point Group: 分子的对称点群。

## 6.2 自洽场计算参数界面



上图是启动 BDF 任务提交界面的自洽场计算参数界面。

下面我们对上图的图形界面的控件及其功能一一说明：

1. Initial Guess: 指定自洽场计算的初始猜测的类型。下拉框支持 **Atom** (利用原子密度矩阵组合分子密度矩阵猜测), **Huckel** (半经验 Huckel 方法猜测), **Hcore** (对角化单电子哈密顿猜测), **Read** (读入分子轨道做为初始猜测) 四种类型, 一般情况下 **atom** 较 **Hcore**、**Huckel** 好, 因此正常情况下无需选择 **Hcore** 或 **Huckel**。
2. Maximum Iterations: 定义 SCF 计算的最大迭代次数。
3. Convergence Threshold: 同时指定 SCF 收敛的能量和密度矩阵阈值。下拉框支持 **Very Tight**, **Tight**, **Default**, **Loose**, **Very Loose** 五种。**Default** 表示  $1.0D-7$   $5.0D-5$ , **Very Tight** 表示  $1.0D-10$   $5.0D-8$ , **Tight** 表示  $1.0D-9$   $5.0D-7$ , **Loose** 表示  $1.0D-7$   $5.0D-5$ , **Very Loose** 表示  $1.0D-6$   $5.0D-4$ 。
4. DFT Integral Grid: 指定 DFT 计算的格点类型。下拉框支持 **Default**, **Ultra Coarse**, **Coarse**, **Medium**, **Fine**, **Ultra Fine** 六种。仅当初始参数界面-Method 选择 Kohn-Sham 计算之后, 才会显示这个界面选项。

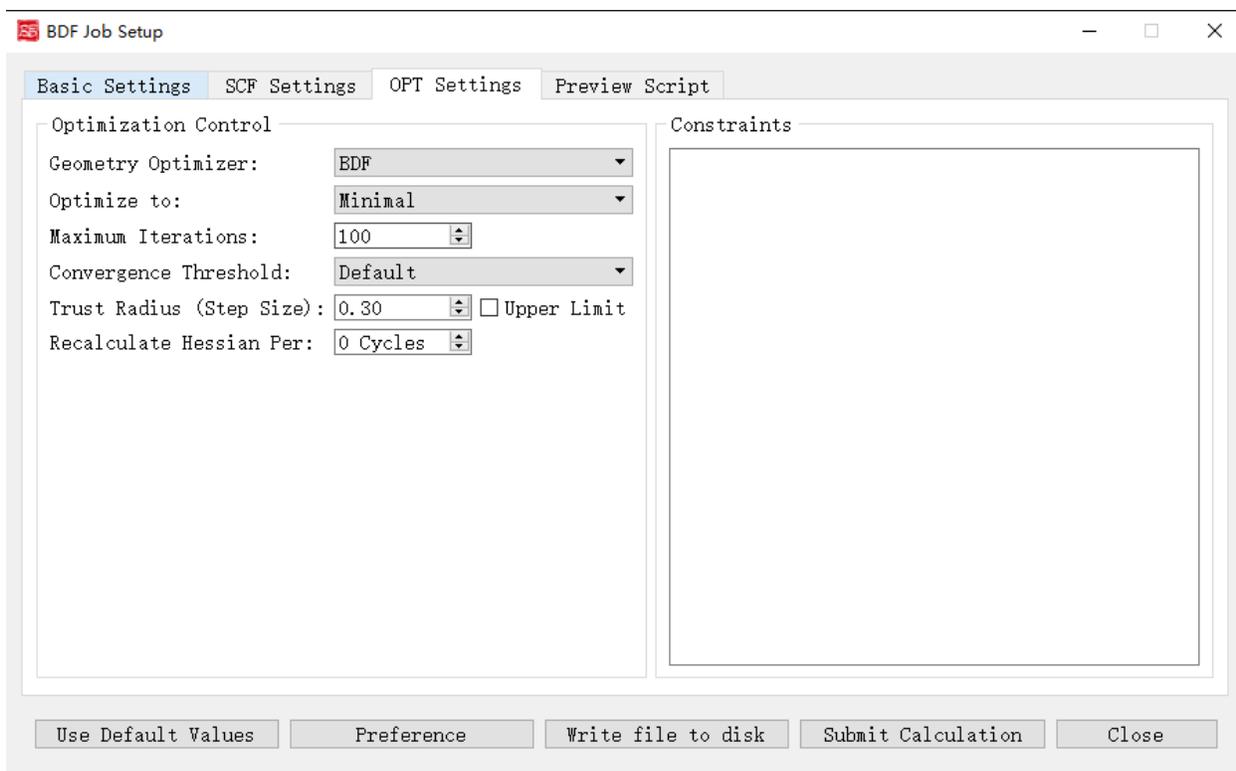
5. **Level Shift**: 指定分子轨道能级移动值, 对应的 scf 模块的 Vshift 参数。人为地将虚轨道能量加上用户指定数值, 以加大 HOMO-LUMO 能隙, 加速收敛。Vshift 值越大, 收敛过程越不容易出现振荡, 但 Vshift 值太大会导致收敛变慢。一般只有在分子的 HOMO-LUMO 能隙较小 (如小于 2 eV), 且 SCF 迭代时能量非单调降低时, 才需要设置 Vshift。
6. **Damping**: 指定本次 SCF 迭代与上次迭代的密度矩阵以一定比例混合 ( $P(i)=(1-C)*P(i)+C*P(i-1)$ ), 从而加速 SCF 收敛, 对应 scf 模块的 Damp 参数。Damp 值越大, 收敛过程越不容易出现振荡, 但 Damp 值太大会导致收敛变慢。一般只有在 SCF 迭代能量非单调降低的时候, 才需要设置 Damp。
7. **Fermi Smearing Temperature**: 指定体系的电子温度, 也即通过费米展宽 (Fermi Smearing) 方法改变前线轨道的占据数。该界面选项受前面的 Level Shift (即 Vshift 参数) 控制, 当 Vshift>0 时, 冻结该界面选项, 且该界面选项值为 0; 当 Vshift=0 时才激活该界面选项。此外, 该界面选项也不可在大分子体系的 FLMO 或 iOI 计算中使用。
8. **Use MPEC+COSX Acceleration**: 指定利用多级展开库伦势 (Multipole expansion of Coulomb potential, MPEC) 方法计算 J 矩阵, COSX (Chain-of-sphere exchange) 方法计算 K 矩阵。此外, 该方法适合计算大分子体系的, 对于小于 20 个原子体系, MPEC+COSX 不推荐使用。
9. **Disable DIIS Acceleration**: 指定不使用 DIIS 加速 SCF 收敛。一般只有在 SCF 能量以较大幅度 (> 1.d-5) 振荡不收敛, 且 scf 模块的 Damp 和 Vshift 参数效果不明显时, 才需要指定该界面复选框。
10. **Print Molecular Orbital**: 控制是否打印分子轨道系数。下拉框支持 **Frontier Orbitals(不打印分子轨道)**, **Energy & Occupation(打印前线轨道 (每个不可约表示的 HOMO-5 到 LUMO+5) 的占据数、能量、系数)**, **All Information(打印所有轨道的占据数、能量、系数)** 三种。
11. **Save Atomic Orbitals**: 计算并存储原子轨道。
12. **Output SCF Orbitals**: 输出 SCF 收敛轨道, 不勾选该界面复选框, 则表示强制不将分子轨道存入.scforb 文件。
13. **Output SCF Orbitals in Pyscf format**: 控制将 SCF 收敛轨道存储为 Pyscf 轨道格式。
14. **Output SCF Orbitals in Molden format**: 控制将分子轨道输出为 Molden 格式, 以做后续的波函数分析。

### 6.3 结构优化计算参数界面

上图是启动 BDF 任务提交界面的结构优化计算参数界面, 前面的计算任务选择 **Optimization (结构优化)**、**Opt+Freq (结构优化 + 频率计算)**、**TDDFT-OPT (激发态结构优化)**、**TDDFT-OPT+Freq (激发态结构优化 + 频率计算)**, 则激活该界面模块。

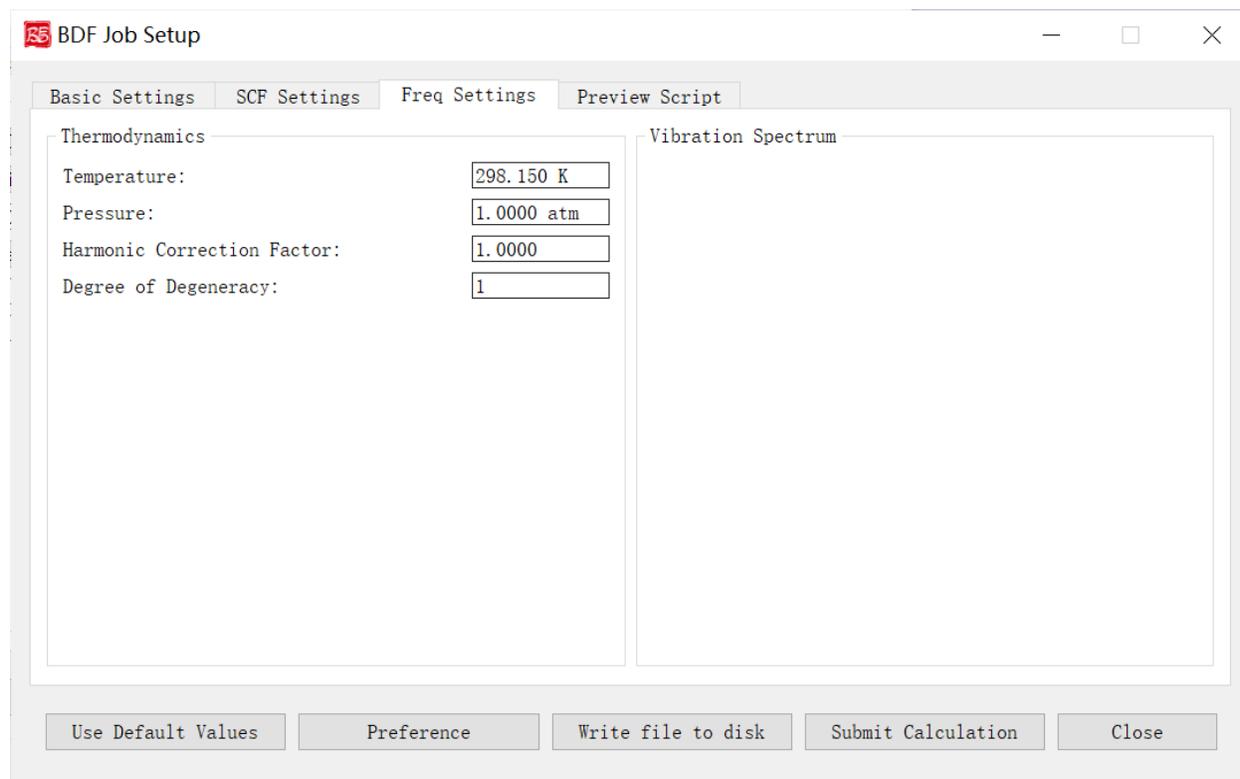
下面我们对上图的图形界面的控件及其功能一一说明:

1. **Geometry Optimizer**: 指定几何结构优化使用的求解器。下拉框支持 **DL-Find** 和 **BDF**, **DL-Find** 优化器支持在直角坐标或内坐标下, 进行能量极小化、过渡态搜索、高阶鞍点搜索、锥形交叉点搜索、最小能量交叉点 (MECP) 搜索等; **BDF** 优化器将使用 BDF 程序自行开发的优化器进行优化。如果在冗余内坐标下 (参见 ICoord 关键词) 进行能量极小化、过渡态搜索, 建议使用 **BDF** 优化器。



2. Optimize to: 优化类型。下拉框支持 **Minimal** (极小值点结构优化), **Transition State** (过渡态结构优化)。
3. Maximum Iterations: 指定最大优化步数。
4. Convergence Threshold: 同时指定均方根梯度和均方根步长的收敛标准。下拉框支持 **Very Tight**, **Tight**, **Default**, **Loose**, **Very Loose** 五种。
5. Trust Radius (Step Size): 指定优化的置信半径, 当设置了优化的初始置信半径  $r$ , 但在随后的结构优化步骤中可能会视优化情况而动态地增加或减少置信半径。而 **Upper Limit** 复选框可以设置优化的初始置信半径为  $-r$ , 且随后的结构优化步骤中保证置信半径不会超过  $|r|$ 。
6. Recalculate Hessian Per: 指定在几何优化中, 每隔多少步计算一次数值 Hessian。
7. Constraints: 指定进行约束性优化, 即在约束一个或多个键长、键角或二面角的情况下, 优化分子其余的自由度。目前该界面选项仅支持前面选择 **BDF** 优化器才生效。用户可自行编辑, 该关键词后面的第一行应是一个整数, 表示约束的数目, 设其为  $N$ ; 第 2 行到第  $N+1$  行, 每一行分别由 2~4 个整数组成。如某一行有 2 个整数, 表示原子编号为这 2 个整数的原子之间的键被冻结; 如某一行有 3 个整数, 表示原子编号为这 3 个整数的原子之间的键角被冻结; 如某一行有 4 个整数, 表示原子编号为这 4 个整数的原子之间的二面角被冻结。

## 6.4 频率计算参数界面

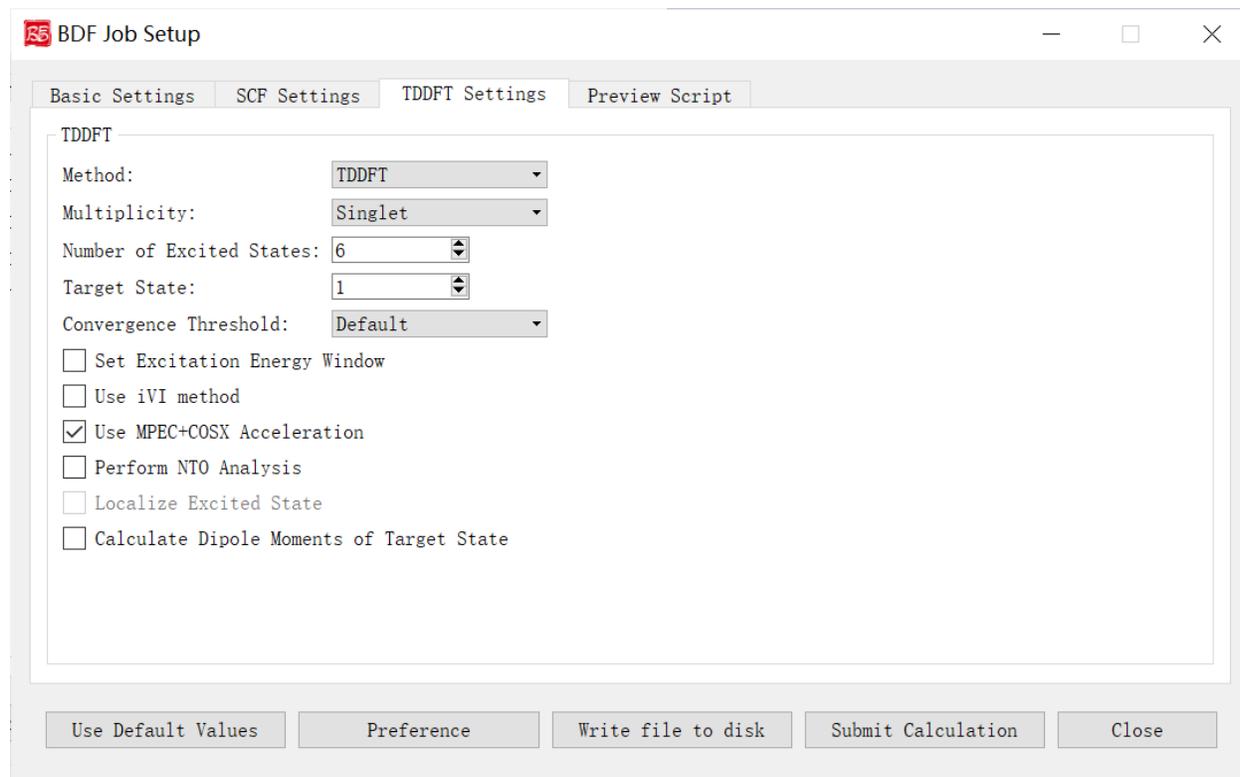


上图是启动 BDF 任务提交界面的频率计算参数界面，前面的计算任务选择 **Frequency**（频率计算）、**Opt+Freq**（结构优化 + 频率计算）、**TDDFT-Freq**（激发态频率计算）、**TDDFT-OPT+Freq**（激发态结构优化 + 频率计算），则激活该界面模块。

下面我们对上图的图形界面的控件及其功能一一说明：

1. **Temperature**: 指定体系进行热化学分析的温度。
2. **Pressure**: 指定体系进行热化学分析的压强。
3. **Harmonic Correction Factor**: 指定频率校正因子。
4. **Degree of Degeneracy**: 指定电子态的简并度，用于计算热化学分析中的吉布斯自由能。电子简并度等于空间简并度乘以自旋简并度，其中空间简并度等于当前电子态所属不可约表示的维数（当分子属于阿贝尔群时，空间简并度等于 1），自旋简并度对于非相对论计算和标量相对论计算等于自旋多重度 ( $2S+1$ )，而对考虑了旋轨耦合的计算等于  $2J+1$ ，其中  $J$  为当前电子态的总角动量量子数。注意即使对于电子简并度不等于 1 的体系，用户必须手动指定正确的 **NDeg** 值，这一点对于开壳层体系的吉布斯自由能计算尤其重要。

## 6.5 激发态计算参数界面



上图是启动 BDF 任务提交界面的激发态计算参数界面，前面的计算任务选择 **TDDFT**（激发态计算）、**TDDFT-SOC**（自旋轨道耦合计算）、**TDDFT-NAC**（非绝热耦合计算）、**TDDFT-OPT**（激发态结构优化）、**TDDFT-Freq**（激发态频率计算）、**TDDFT-OPT+Freq**（激发态结构优化 + 频率计算），则激活该界面模块。

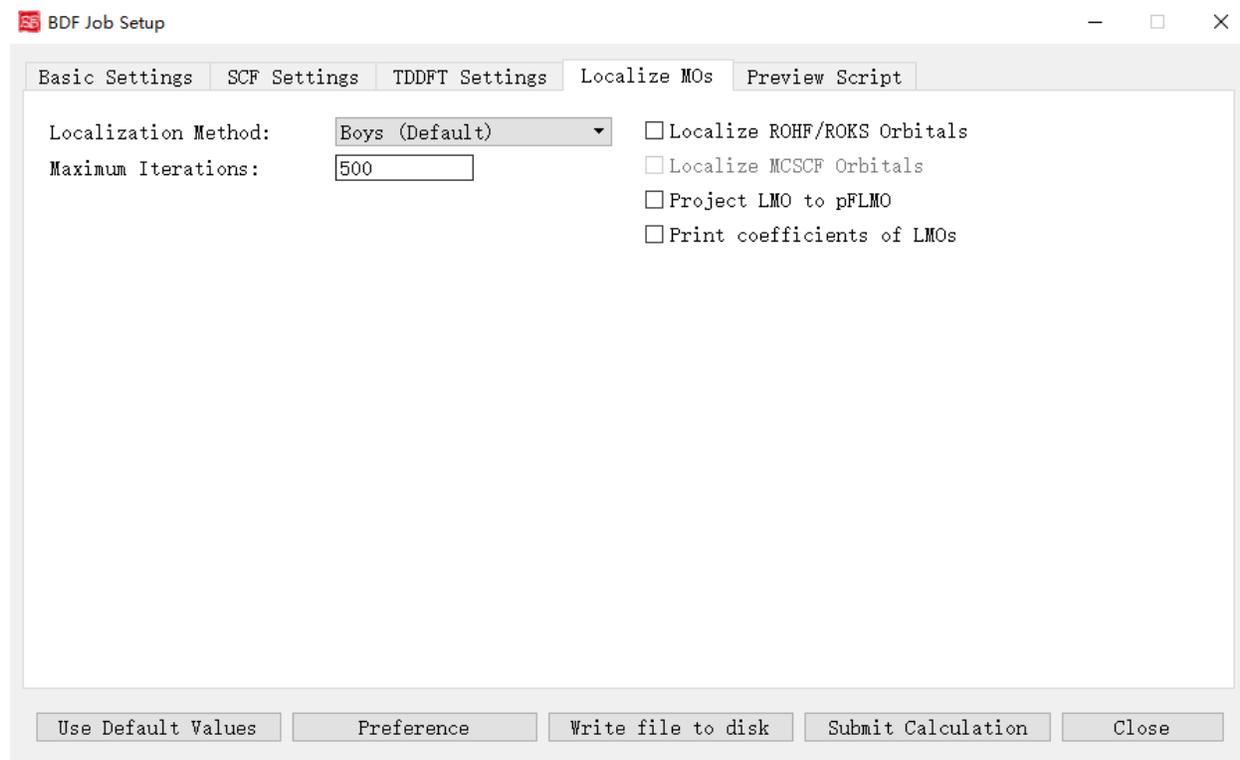
下面我们对上图的图形界面的控件及其功能一一说明：

1. **Method**: 指定计算方法。下拉框支持 **TDDFT** 和 **TDA**。
2. **Multiplicity**: 指定计算激发态的自旋多重度。下拉框支持两组情况：一组为当初始参数界面-Multiplicity=1，则下拉选项为 **Singlet**（计算单重态），**Triplet**（计算三重态），**Singlet & Triplet**（分别计算单重态与三重态）；二组为当初始参数界面-Multiplicity=2，则下拉选项为 **Doublet**（计算二重态），**Quartet**（计算四重态），**Doublet & Quartet**（分别计算二重态和四重态）。
3. **Delta Ms**: 控制是否进行 spin-flip 的 TDDFT 计算。可选值 0, 1, -1；0 为 no spin-flip（或称 spin-conserving，计算磁量子数 Ms 与基态相同的激发态）；1 为 spin flip up（计算 Ms 比基态大 1 的激发态）；-1 为 spin flip down（计算 Ms 比基态小 1 的激发态）。当初始参数界面-Multiplicity>2，被激活该界面选项。
4. **Number of Excited States**: 指定计算的激发态数目。
5. **Target State**: 指定计算第几个激发态偶极矩。仅当勾选 **Calculate Dipole Moments of Target State** 复选框，该界面选项才能生效。
6. **Convergence Threshold**: 指定 TDDFT 计算能量和波函数的收敛阈值。下拉框支持 **Very Tight**, **Tight**, **Default**, **Loose**, **Very Loose** 五种。**Default** 表示 1E-7 1E-5, **Very Tight** 表示 1E-9 1E-7, **Tight** 表示 1E-8

1E-6, **Loose** 表示 1E-6 1E-4, **Very Loose** 表示 1E-5 1E-3。

7. Set Excitation Energy Window: 指定计算哪个能量/波长范围内的激发态, 即直接指定激发能/激发波长的范围。
8. Use iVI method: 指定 TDDFT 的 iVI 对角化方法 (不支持非阿贝尔点群), 对于下述情况之一建议使用该方法: 第一、X 射线吸收/发射光谱等涉及很高的激发态的计算; 第二、计算某个能量或波长范围内的所有激发态, 并且要求既不多算该范围外的激发态, 又不少算该范围内的激发态。
9. Use MPEC+COSX Acceleration: 指定利用多级展开库伦势 (Multipole expansion of Coulomb potential, MPEC) 方法计算 J 矩阵, COSX (Chain-of-sphere exchange) 方法计算 K 矩阵。此外, 该方法适合计算大分子体系的, 对于小于 20 个原子体系, MPEC+COSX 不推荐使用。
10. Perform NTO Analysis: 指定对 TDDFT 计算的所有态做 NTO 分析, 目前仅支持阿贝尔点群的 TDDFT 计算。
11. Localize Excited State: 指定计算定域化激发态。目前该界面上不允许用户选择, 用户可自行在生成的 BDF 输入文件 (.inp) 中进行修改。
12. Calculate Dipole Moments of Target State: 指定计算激发态偶极矩。

## 6.6 分子轨道定域化参数界面



上图是启动 BDF 任务提交界面的分子轨道定域化参数界面, 当初始参数界面-计算类型勾选了 **Localize molecular orbital** 复选框, 则激活该界面模块。

下面我们对上图的图形界面的控件及其功能一一说明：

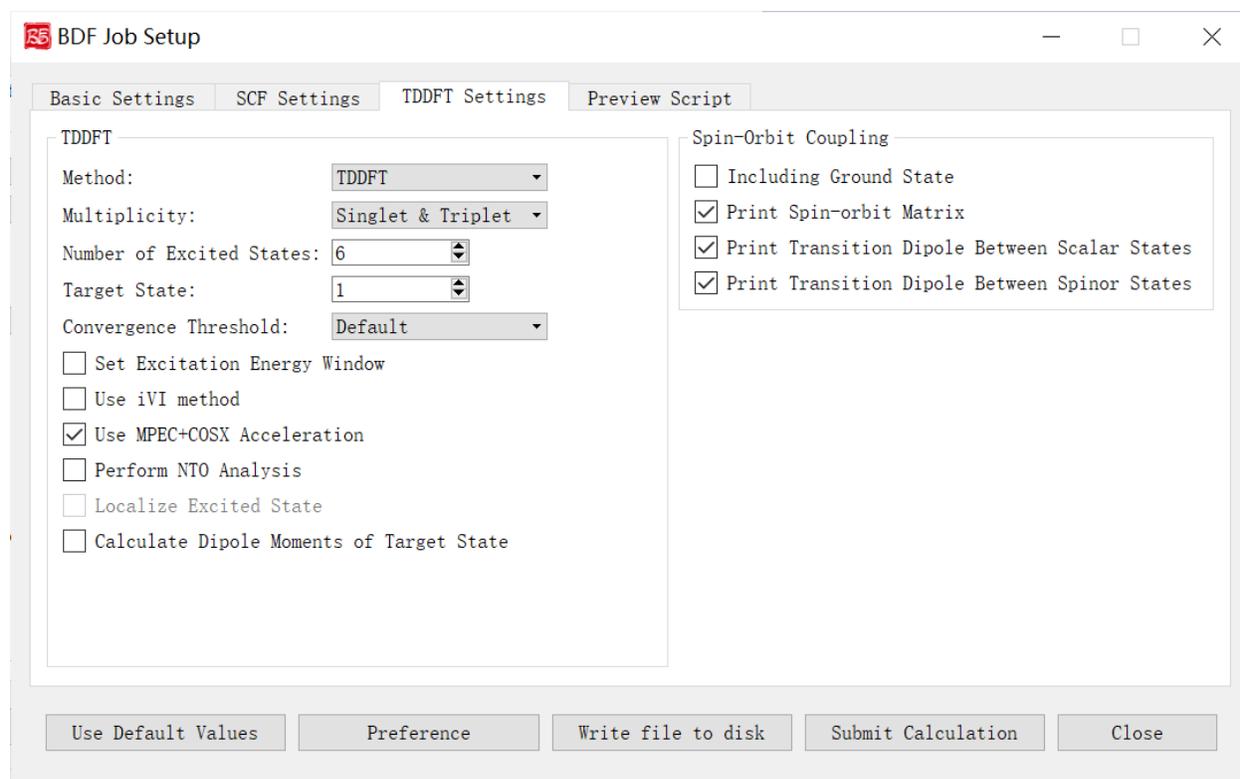
1. Localization Method: 指定产生定域化的分子轨道的方法。下拉框支持 **Boys (Default)** , **Modified Boys** , **Four-center moment** , **Pipek-Mezey** 四种。
2. Exponential Factor: 指定指数因子。前面的 Localization Method 选择 Modified Boys 或 Four-center moment 时激活该界面选项。
3. Atomic Charge: 指定使用的电荷类型。下拉框支持 **Mulliken** 和 **Lowdin** 。前面的 Localization Method 选择 Pipek-Mezey 时激活该界面选项。
4. Pipek-Mezey Method: 具体指定 Pipek-Mezey 方法定域轨道的方法。下拉框支持 **Jacobi Sweep** (指定 Pipek-Mezey 方法利用雅可比旋转定域轨道) 和 **Trust Region** (指定 Pipek-Mezey 方法利用 Trust Region 方法定域轨道)。前面的 Localization Method 选择 Pipek-Mezey 时激活该界面选项。
5. Maximum Iterations: 指定定域化允许的最大循环次数。
6. Localize ROHF/ROKS Orbitals: 指定定域化 ROHF/ROKS 轨道。
7. Localize MCSCF Orbitals: 指定定域化多组态自洽场轨道。目前该功能尚未在当前发布的 BDF 版本中支持, 故界面上不允许用户选择, 在未来 BDF 软件更新相关功能之后才会解锁。
8. Project LMO to pFLMO: 指定投影 LMO 到 pFLMO。
9. Print coefficients of LMOs: 指定打印定域化分子轨道的系数。

## 6.7 自旋轨道耦合计算参数界面

上图是启动 BDF 任务提交界面的自旋轨道耦合计算参数界面, 即 Spin-Orbit Coupling 部分参数, 前面的计算任务选择 **TDDFT-SOC (自旋轨道耦合计算)**, 则会激活该界面模块。

下面我们对上图的图形界面的控件及其功能一一说明：

1. Including Ground State: 指定 TDDFT-SOC 计算是否包含基态。勾选该界面选项则表示为 TDDFT-SOC 计算包含基态, 此时可以得到包含 SOC 校正的光谱, 且可以计算基态的 SOC 校正, 但此时纳入 TDDFT-SOC 处理的标量激发态的数目不宜过多 (一般以 10~100 个左右为宜), 否则会低估基态能量, 从而高估激发能; 不勾选该界面选项则表示为 TDDFT-SOC 计算不包含基态, 此时无法得到基态和考虑了 SOC 的激发态 (即旋量态) 之间的跃迁偶极矩, 因此无法绘制包含 SOC 校正的光谱, 同时也无法计算基态的 SOC 校正, 但仍可得到包含 SOC 校正的激发能。
2. Print Spin-orbit Matrix: 指定需要计算的 SOC 矩阵元。
3. Print Transition Dipole Between Scalar States: 指定在 TDDFT-SOC 计算里, 计算标量态之间的跃迁偶极矩, 勾选该选项则表示可以打印所有标量态间的跃迁偶极矩。
4. Print Transition Dipole Between Spinor States: 指定打印考虑 SOC 之后的旋量态之间的跃迁偶极矩 (以及对应的振子强度, 和根据费米黄金规则计算得到的辐射跃迁速率常数)。

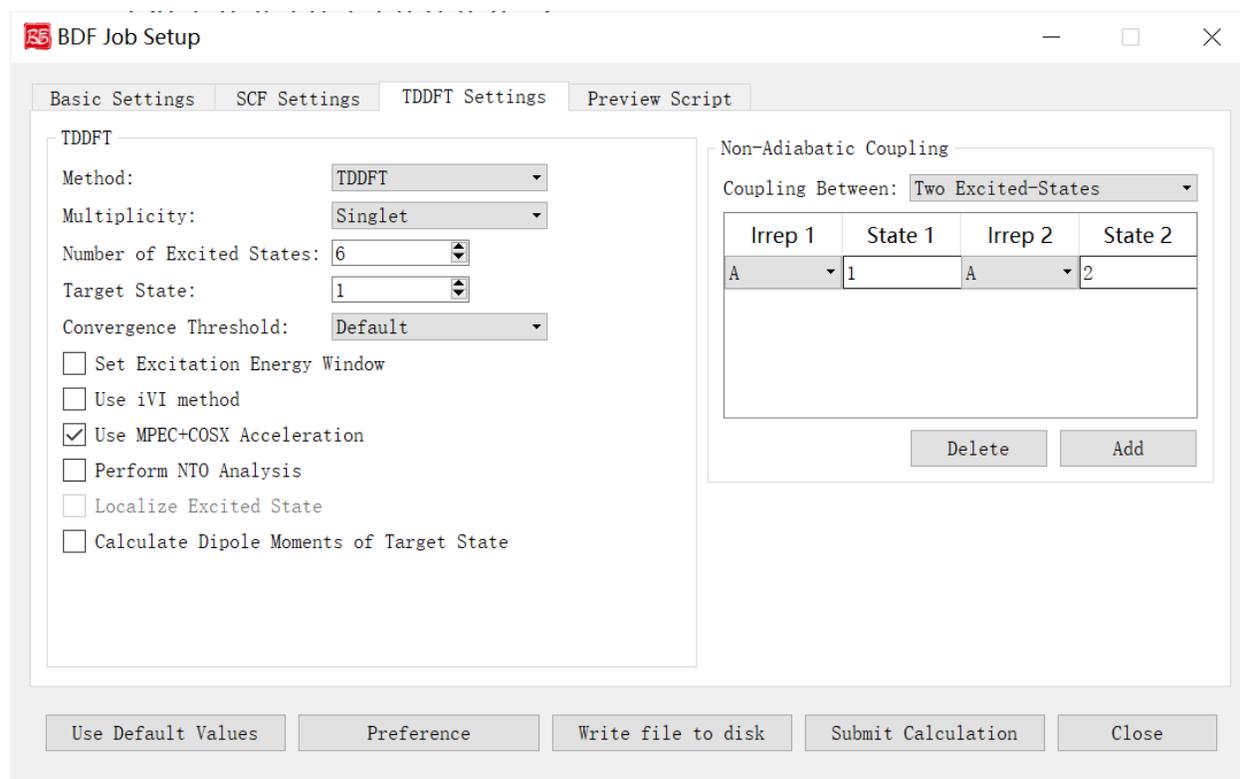


## 6.8 非绝热耦合计算参数界面

上图是启动 BDF 任务提交界面的非绝热耦合计算参数界面，即 Non-Adiabatic Coupling 部分参数，前面的计算任务选择 **TDDFT-NAC (非绝热耦合计算)**，则会激活该界面模块。

下面我们对上图的图形界面的控件及其功能一一说明：

1. **Coupling Between:** 指定计算哪些电子态间的非绝热耦合矩阵元（包括基态-激发态之间的非绝热耦合矩阵元，和激发态-激发态之间的非绝热耦合矩阵元）。下拉框支持 **Ground and Excited-State (基态-激发态)** 和 **Two Excited-States (激发态-激发态)**。Irrep 1 和 State 1 分别指定激发态的第几个不可约表示和该不可约表示的第几个根，用于指定计算基态-激发态非绝热耦合向量。Irrep, State 1 和 Irrep, State 2 分别指定两组激发态的第几个不可约表示和该不可约表示的第几个根，用于指定计算激发态-激发态非绝热耦合向量。





## MODULE FUNCTIONS

See *the flowchart of BDF modules and calculations* for the invocation of each BDF module.

### 7.1 分子自动分片，FLMO 和 iOI 计算 - AUTOFRAG 模块

autofrag 的主要功能是对大分子进行自动分片，并自动产生 FLMO 计算的输入。autofrag 还是 iOI-SCF 的计算引擎，用于调度 SCF 的其他模块完成 iOI 计算。autofrag 模块主要计算流程为：

- 根据分子坐标自动产生原子的键连信息；
- 切断某些键，产生合适的分子片段；
- 对分子片段增加缓冲原子；
- 在断键处添加连接 H 原子或者投影杂化轨道 PHO (Projected Hybrid Orbital)；
- 生成子体系分子片段的输入文件，利用并行化的模式计算分子片段；
- 组织子体系分子片段计算结果，运行整体分子计算。

autofrag 必须放在 compass 之前。autofrag 会处理 compass 模块中的分子结构，进行分子键判断，分片等。autofrag 后的输入还被作为产生分子片段及整体计算输入的模版。利用 autofrag 做完整的例子见 iOI-SCF 计算示例

#### 7.1.1 Method 参数类型：字符串

- 默认值：FLMO
- 可选值：FLMO、iOI

设定计算方法。FLMO - 计算分片定域化分子轨道；iOI - iOI-SCF 计算，用分子片合成分子的方法计算大分子体系。iOI 目前仅支持限制性自洽场计算。

### 7.1.2 *nprocs* 参数类型：整数

- 默认值：1
- 可选值：小于等于分子片数的正整数

设定 iOI 计算最大可用进程数。由于分子片计算是独立的，可并行计算，此处 *nprocs* 指的即是最多允许多少个分子片同时计算。注意当 *nprocs* 不等于 1 时，环境变量 `OMP_NUM_THREADS` 指定的是 iOI 计算的总 OpenMP 线程数，而不是每个进程的可用 OpenMP 线程数。例如当环境变量 `OMP_NUM_THREADS=16`，且 *nprocs* 为 2 时，代表 iOI 计算中同时最多有 2 个子体系在并行计算，每个子体系使用 8 个 OpenMP 线程。

### 7.1.3 *radcent* 参数类型：浮点

- 默认值：3.0
- 可选值：非负浮点数

设定初始分子片（即尚未添加缓冲区的分子片）的大小（单位：埃），增加 *radcent* 会增加每个分子片的尺寸，且减少分子片的数目。

### 7.1.4 *radbuff* 参数类型：浮点

- 默认值：2.0
- 可选值：非负浮点数

设定分子片缓冲区半径（单位：埃）。与 *radcent* 不同，增加 *radbuff* 不改变分子片数目，但会增加每个分子片的尺寸。对于 iOI 计算，*radbuff* 定义的是第 0 次宏迭代时分子片缓冲区的大小，缓冲区将在宏迭代过程中增大。

### 7.1.5 *iOIThresh* 参数类型：浮点

- 默认值：0.1
- 可选值：正浮点数

设定 iOI-SCF 分子片段计算的收敛阈值。减小 *iOIThresh* 会增加 iOI 计算的宏迭代次数，但改进了整体分子计算的初始轨道，所以会加快整体分子的 SCF 收敛。

### 7.1.6 NoPHO 参数类型: Bool

设置不利用 **PHO** (Project Hybrid orbital), 而是利用连接氢 (**H**) 原子来饱和分子分片时的断键处。使用该关键词时, 子体系计算的计算量比默认的 **PHO** 方法稍小, 但子体系轨道的精度低, 可能导致整体分子的 SCF 迭代次数增加, 乃至使总计算时间增加。

### 7.1.7 charge 参数类型: 整数数组

- 默认值: 无

设定给定原子的电荷数, 用以协助指定分子片段的电荷。当程序难以自动确定某分子片段电子数时, 用户可以通过指定电荷来确定分子片段总的电子数。格式如下:

```
charge
10 +2 25 -1 78 -1
```

这里, 指定第 10 个原子的电荷数为 +2, 第 25 个原子的电荷数为-1, 第 78 个原子的电荷数为-1。原子所属的分子片段电荷数将会依照用户给出的原子电荷数来进行确定。

### 7.1.8 spinocc 参数类型: 整数数组

- 默认值: 无

设定给定原子的形式自旋, 用以协助计算到合适的自旋态。输入格式和 charge 关键词相同。

```
spinocc
13 +1 17 -1
```

这里, 指定第 13 个原子有 1 个未成对的 alpha 电子, 第 17 个原子有 1 个未成对的 beta 电子。注意, 所有的开壳层原子都应该被指定。例如一个体系有两个 Cu(II) 中心, 则两个 Cu 的形式自旋可以都不指定 (此时总体系收敛到哪个自旋态是不确定的), 也可以都指定, 但是不能只指定其中一个的形式自旋而不指定另一个的形式自旋; 但是如果两个 Cu 原子之中有一个是 Cu(I), 则 Cu(I) 的形式自旋可以不指定, 因为其为闭壳层原子。如果体系有离域的自旋, 则应该画出让该自旋局域在某个原子上的共振式, 再按该共振式指定形式自旋。例如乙烯自由基阳离子的两个碳原子均带有形式正电荷 +0.5 和形式自旋 +0.5, 但是指定形式自旋时应该将其中任意一个碳的形式自旋指定为 +1, 另一个碳的形式自旋指定为 0 (因为此时该碳原子是闭壳层原子, 相应的形式自旋也可以不用指定), 而不能把两个碳原子的形式自旋都指定为 +0.5。

### 7.1.9 *maxiter* 参数类型：整数

- 默认值：50

指定 iOI-SCF 最大的宏迭代次数。

### 7.1.10 *Dryrun* 参数类型：Bool

- 默认值：False

设定只产生 FLMO 或 iOI-SCF 输入文件，而不执行计算。

## 7.2 对称性及预处理 - COMPASS 模块

COMPASS 模块主要完成计算任务的初始化工作，包括读入用户定义的分子结构、基组等基本信息，判断分子的对称性及其分子点群，产生对称匹配的轨道等，并将其转换为 BDF 内部的数据存储起来。Compass 模块的主要参数有：

### 7.2.1 *Basis* 参数类型：字符串

指定计算所用的基组名称。BDF 基组存储在 \$BDFHOME/basis\_library 中，当前计算任务中所有原子的基组应被放置在本参数指定的文件中。由于基组通过本参数指定的文件读入，用户可以通过自定义基组（见自定义基组说明）文件为不同原子指定不同的基组。

```
$Compass
Basis
  cc-pVDZ
Geometry
H   0.00  0.00  0.707
H   0.00  0.00 -0.707
End Geometry
$End
```

### 7.2.2 *Basis-block* 参数类型：字符串

对不同元素指定不同的基组，第一行是默认基组，之后的行对不同元素或原子指定其它基组，格式为 元素 = 基组名或者 元素 1, 元素 2, ..., 元素 n = 基组名。必须以 End Basis 做为结束。

```
$Compass
Basis-block
  3-21g
```

(continues on next page)

(continued from previous page)

```

C,N = 6-31g
Xe = cc-pvdz-pp
End Basis
Geometry
H    0.0  0.0 -1.1
C    0.0  0.0  0.0
N    0.0  0.0  1.0
Xe   3.0  0.0  0.0
End geometry
$End

```

### 7.2.3 MPEC+COSX 参数类型: Bool 型

指定利用多级展开库伦势 (Multipole expansion of Coulomb potential, MPEC) 方法计算 J 矩阵, COSX (Chain-of-sphere exchange) 方法计算 K 矩阵。

### 7.2.4 RI-J、RI-K、RI-C 参数类型: 字符串

密度拟合近似 (Density-fitting approximation) 加速算法的辅助基组。

- RI-J: 库伦拟合基组
- RI-K: 交换拟合基组
- RI-C: 相关拟合基组

```

$Compass
Basis
  DEF2-SVP
RI-J
  DEF2-SVP
Geometry
H    0.00  0.00  0.707
H    0.00  0.00 -0.707
End Geometry
$End

```

## 7.2.5 Geometry 参数类型：字符串数组

指定计算的分子结构，可以是直角坐标模式，也可以是内坐标模式。分子坐标的定义从 `Geometry` 参数下一行开始，以 `End Geometry` 前一行结束。

直角坐标模式

```
$Compass
Basis
  cc-pVDZ
Geometry
H   0.00  0.00  0.707
H   0.00  0.00 -0.707
End Geometry
$End
```

内坐标模式

```
$Compass
Basis
  cc-pVDZ
Geometry
O
H   1  0.9
H   1  0.9  2 109.0
End Geometry
$End
```

## 7.2.6 Restart 参数类型：Bool 型

使用 `$BDFTASK.optgeom` 文件里的坐标，而非 `Geometry` 关键词下给定的坐标来进行计算，其中 `$BDFTASK` 是输入文件名称去掉后缀 `.inp` 后剩余的字符串。注意虽然此时计算不会使用 `Geometry` 关键词后的坐标数值，但是 `Geometry` 关键词后的坐标并不能省略，而且原子的种类、个数和顺序必须正确，只不过坐标数值可以是任意的而已。例如假设输入文件名为 `1.inp`，而 `1.optgeom` 文件的内容为

```
GEOM
O 0. 0. 0.
H 0. 0. 2.
H 0. 2. 0.
```

则当 `1.inp` 的 `$compass` 模块为以下形式时，程序可以正常运行：

```
$compass
...
```

(continues on next page)

(continued from previous page)

```

geometry
O 0. 0. 0.
H 0. 0. 2.1
H 0.1 2.0 0.
end geometry
restart
...
$end

```

该写法等价于以下输入（即便在以上写法中指定坐标单位为埃也是如此）：

```

$compass
...
geometry
O 0. 0. 0.
H 0. 0. 2.
H 0. 2. 0.
end geometry
unit
  bohr
...
$end

```

但 1.inp 的 \$compass 模块不能按以下的形式写，因为原子数目和.optgeom 文件不符：

```

$compass
...
geometry
O 0. 0. 0.
H 0. 2.1 0.
end geometry
restart
...
$end

```

也不能写成以下的形式，因为原子的顺序和.optgeom 文件不符：

```

$compass
...
geometry
H 0. 2.1 0.
O 0. 0. 0.
H 0. 0. 2.1
end geometry

```

(continues on next page)

(continued from previous page)

```
restart
...
$end
```

`restart` 主要用于结构优化的断点续算。仍以 `1.inp` 为例，假设 `1.inp` 是一个结构优化任务的输入文件，但因优化不收敛、其他程序报错或用户中止计算而非正常结束，则结构优化最后一步的结构保存在 `1.optgeom` 内，此时在 `1.inp` 的 `$compass` 模块内添加 `restart` 关键字，然后重新运行 `1.inp`，即可起到从之前结构优化的最后一帧开始继续进行结构优化的作用，而无需手动将 `1.optgeom` 的内容拷贝至 `1.inp` 内。

### 7.2.7 Group 参数类型：字符串

指定分子的对称点群。BDF 会自动判断分子的对称性，HF/DFT/TDDFT 都支持高阶分子点群。一些电子相关计算，如 MCSCF, MRCI 等只支持 `D2h` 及其子群。因此，本参数可用来强制 BDF 使用阿贝尔群计算。

```
# 苯分子最高对称性是 D6h, 不指定分子点群, BDF 可以判断出苯分子的对称性, 并按 D6h 群计算
$COMPASS
Title
  C6H6 Molecule test run, cc-pVDZ
Basis
  cc-pVDZ
Geometry
C    0.0000000000000000    1.3949910000000000    0.0000000000000000
C   -1.20809764405066    0.6974955000000000    0.0000000000000000
C    0.0000000000000000   -1.3949910000000000    0.0000000000000000
C   -1.20809764405066   -0.6974955000000000    0.0000000000000000
C    1.20809764405066   -0.6974955000000000    0.0000000000000000
C    1.20809764405066    0.6974955000000000    0.0000000000000000
H    0.0000000000000000    2.4946010000000000    0.0000000000000000
H   -2.16038783830606    1.2473005000000000    0.0000000000000000
H    0.0000000000000000   -2.4946010000000000    0.0000000000000000
H   -2.16038783830607   -1.2473005000000000    0.0000000000000000
H    2.16038783830607   -1.2473005000000000    0.0000000000000000
H    2.16038783830606    1.2473005000000000    0.0000000000000000
End geometry
Check
$END

# D6h 的子群有 D3h, C6v, D3d, D2h, C2v, C1 等。下面的例子指定用 D2h 群计算苯分子。
$COMPASS
Title
  C6H6 Molecule test run, cc-pVDZ
Basis
  cc-pVDZ
```

(continues on next page)

(continued from previous page)

```

Geometry
...
End geometry
Check
Group
  D(2h)
$END

```

### 7.2.8 *Nosymm* 参数类型: Bool 型

- 默认值: false

强制 BDF 在计算时忽略分子对称性。

**Attention:** 与指定 C1 群不同, 一旦使用了这个参数, 分子坐标不会旋转。默认情况下, 分子坐标会被旋转到标准取向 (Standard orientation)

### 7.2.9 *Norotate* 参数类型: Bool 型

强制不将分子坐标旋转到标准取向。与 *Nosymm* 不同, *Norotate* 不会忽略分子的对称性, 但是当分子具有对称轴或者镜面时, 对称轴和镜面的位置必须和分子旋转到标准取向时对称轴和镜面的位置一致。例如计算某个处于 *yz* 平面上, 且关于 *xz* 平面对称的水分子, 默认情况下 BDF 会将其旋转到标准取向, 导致该分子处于 *xz* 平面。此时用 *Norotate* 可以强制 BDF 在该水分子处于 *yz* 平面的情况下进行计算, 且仍然可以利用水分子的 C(2v) 对称性, 因为不管水分子是处于 *xz* 平面还是 *yz* 平面, 水分子的对称轴都是位于 *z* 轴, 且两个镜面都是位于 *xz* 面和 *yz* 面。但如果该水分子处于 *xy* 面, 则指定 *Norotate* 将会报错, 因为此时水分子的对称轴并非位于 *z* 轴。

### 7.2.10 *Unit* 参数类型: 字符串

- 默认值: Angstrom
- 可选值: Bohr, Angstrom

Bohr 表示坐标输入的长度单位为原子单位制, Angstrom 表示长度单位为埃。

### 7.2.11 *Skeleton* 参数类型: Bool 型

指定在 BDF 计算中对称性的处理方式。BDF 有两种分子点群对称性处理方式：一种是先构造对称匹配的轨道，原子轨道积分计算过程中会对称化积分，并存储基于对称匹配轨道的积分，该方法只支持非积分直接的波函数理论计算，如 SCF, MCSCF, MRCI, CCSD 等；另一种做法是不产生对称匹配的积分，而是只计算存储对称独立的原子轨道积分，在做 Hartree-Fock、Kohn-Sham DFT 等计算时，直接产生对称匹配的算符如 J、K 矩阵。后一种做法称为 *Skeleton* 方法，即只计算“骨架”原子轨道积分。*Skeleton* 方法对各种非阿贝尔点群提供支持。

BDF 最初默认使用第一种做法，如今默认使用第二种做法，但是无法用于各种后-HF 波函理论方法，此时需要用 *Saorb* 关键词切换到第一种做法。

### 7.2.12 *Saorb* 参数类型: Bool 型

指定在 BDF 计算中对称性的处理方式为传统做法。只有做各种后-HF 波函理论计算时才需要指定该关键词。参见 *Skeleton* 关键词。

### 7.2.13 *Extcharge* 参数类型: Bool 型

无需输入值

指定计算需要外加点电荷，外加点电荷放置于名为 `$BDFTASK.extcharge` 的文件中。该文件内容如下：

第一行是标题行，可以为空行。

第二行输入一个整数 N，定义需要多少个附加电荷。

第三到末尾，总共 N 行，定义附加电荷的坐标与电量，格式如下：

```
Atom Charge x y z
```

### 7.2.14 *Thresh* 参数类型: 字符串

- 默认值: Medium
- 可选值: Coarse, Medium, Strict

指定判断分子对称性的精度。BDF 的一个特色是对分子点群的支持。*compass* 模块可以自动识别分子所属的对称群，并按照对称群将分子严格对称化。由于分子建模精度，分子可能不严格属于某个对称点群，本参数可以控制判断分子对称的精度。

```
$COMPASS
Basis
  cc-pVDZ
Geometry
C      0.0000000000000000    1.39499100000000    0.0000000000000000
```

(continues on next page)

(continued from previous page)

```

C   -1.20809764405066   0.69749550000000   0.00000000000000
C     0.00000000000000  -1.39499100000000   0.00000000000000
...
End geometry
Thresh
  Medium
$END

```

### 7.2.15 *Uncontract* 参数类型: Bool 型

强制使用非收缩的原高斯基函数计算，无论输入基组是否是收缩基组。通常用于测试。

### 7.2.16 *Primitive* 参数类型: Bool 型

指定只输入特定格式的原高斯基函数。通常用于测试。

### 7.2.17 *Check* 参数类型: Bool 型

按照特定格式打印最重要结果。通常用于测试。

## 7.3 单、双电子积分计算 - XUANYUAN 模块

XUANYUAN 模块主要计算单、双电子积分和其他必要的积分并存储到文件中。

### 7.3.1 *Direct* 参数类型: Bool 型

指定使用积分直接的 SCF (Direct) 计算。如今是默认选项，不需要用户设置。

积分直接的 SCF 不存储双电子积分，按照 Schwartz 不等式，结合积分对 Fock 矩阵的贡献，对积分进行预筛选。当基函数数目大于 300 左右，可以有效地利用双电子积分重复计算来避免 IO 操作，且支持 OpenMP 的多核并行计算。BDF 中大多数需要计算 Fock-Like 矩阵 (J 与 K 矩阵) 的模块，如 SCF, TDDFT 等都已经实现了积分直接计算。

---

**Note:** 积分直接的 SCF 计算需要同时在 compass 模块中加上 Skeleton 关键词，如今也是默认的。关闭积分直接的 SCF 计算可以在 compass 模块中加上 Saorb 关键词。

---

### 7.3.2 Maxmem 参数类型: 字符串

指定非积分直接的 SCF 双电子积分计算缓存大小。大的缓存可以减少积分排序的次数。输入格式为数字 +MW 或数字 +GW, 1 Word=2 Byte, 因此 512MW 就等于 1024MB。

```
$xuanyuan
Maxmem
  512MW
$end
```

### 7.3.3 RSOMEGA / RS 参数类型: 浮点型

指定 Range-Separated 泛函如 CAM-B3LYP 等的  $\omega$  (某些文献称  $\mu$ ) 系数。RS 是 RSOMEGA 的同义词。如果 DFT 使用了 Range-Speration 泛函, 必须加入此参数。建议值如下:

Table 7.1: 标准范围分离泛函中的  $\omega$  值

标准范围分离泛函	$\omega$ 值
CAM-B3LYP	0.33
LC-BLYP	0.33
wB97	0.40
wB97X	0.30

```
$xuanyuan
RSOMEGA
  0.33
$end

$scf
  dft
    cam-b3lyp
$end
```

### 7.3.4 Heff 参数类型: 整型

- 默认值: 0
- 可选值: 0、1、2、3/4、5、21、22、23

指定标量相对论哈密顿, 包括:

- 0: 非相对论, 包括使用 ECP 的情况
- 1: sf-ZORA (不建议普通用户使用)

- 2: sf-IORA (不建议普通用户使用)
- 3、4: sf-X2C (二者计算流程不同, 一般用 3 即可)
- 5: sf-X2C + so-DKH3 (无自旋部分, 需结合 H<sub>soc</sub> 使用, 精度有待进一步测试) [12]
- 21: sf-X2C (同 3、4, 但支持解析导数和部分单电子性质) [27]
- 22: sf-X2C-aXR (使用原子 X 矩阵近似的 sf-X2C, 支持解析导数和部分单电子性质) [27]
- 23: sf-X2C-aU (使用原子酉变换近似的 sf-X2C, 支持解析导数和部分单电子性质) [27]

```
$xuanyuan
Hef
  3
$end
```

### 7.3.5 H<sub>soc</sub> 参数类型: 整型

- 可选值: 0、1、2、3、4、5

指定自旋轨道 (SO) 积分的类型, 包括:

- 0: so-1e, 仅计算单电子 SO 积分。
- 1: so-1e + SOMF, 通过有效 Fock 算符计算双电子 SO 积分。对于全电子计算, 这是最准确的方法
- 2: so-1e + SOMF-1c, 使用单中心近似的 SOMF。对于全电子计算, 这是推荐选项, 尤其是计算大分子
- 3: so-1e + SOMF-1c / no soo, 关闭 2 中的自旋-其它轨道 (SOO) 贡献
- 4: so-1e + SOMF-1c / no soo + WSO\_XC, 采用 DFT 计算 SOO 贡献
- 5: so-1e + somf-1c / no soo + WSO\_XC-2x, 根据 Neese 的建议, 把 DFT 部分乘以-2 来模拟 SOO 贡献
- 以上参数加上 10, 将使用 BP 近似下的算符
- 对于 ECP 基组 (包括标量 ECP 基组、SOECP 基组、全电子非相对论基组的混合), 唯一接受的值是 10, 并且是默认选择。也就是使用 BP so-1e, 其中对 SOECP 原子计算 SOECP 积分, 标量 ECP 原子和全电子非相对论原子用有效核电荷, 但是后者支持的元素和基组类型有限 (见 soint\_util/zefflib.F90)。

```
$xuanyuan
Hsoc
  1
$end
```

### 7.3.6 *Nuclear* 参数类型：整数

- 默认值：0
- 可选值：0、1

指定原子核电荷分布模型。0 为点电荷模型；1 为高斯电荷模型。对于 110 号 (Ds) 之前的元素，均方根 (RMS) 核半径取自 Visscher 和 Dyall 汇总的数据 [70]；从 Ds 元素开始，RMS 核半径通过核素质量 A 估算 (单位：费米)：

$$\langle r^2 \rangle \approx 0.57 + 0.836 A^{1/3}$$

其中的核素质量 A 与核电荷数 Z 近似满足以下关系 [71, 72]：

$$A \approx 0.004467 Z^2 + 2.163 Z - 1.168$$

## 7.4 Hartree-Fock 及 Kohn-Sham 自洽场计算 - SCF 模块

SCF 模块是 BDF 的核心计算模块之一，进行 Hartree-Fock 和 DFT 计算。

计算方法关键词

### 7.4.1 *RHF* / *UHF* / *ROHF* 参数类型：Bool 型

如果做 Hartree-Fock 计算，这三个参数必须 3 选 1，用于控制 Hartree-Fock 计算的类型。

- RHF Restricted Hartree-Fock
- UHF Unrestricted Hartree-Fock
- ROHF Restricted Open-shell Hartree-Fock

### 7.4.2 *RKS* / *UKS* / *ROKS* 参数类型：Bool 型

如果做 DFT 计算，这三个参数必须 3 选 1，用于控制 DFT 计算的类型。

- RKS Restricted Kohn-Sham
- UKS Unrestricted Kohn-Sham
- ROKS Restricted Open-shell Kohn-Sham

波函数与占据数关键词

### 7.4.3 Charge 参数类型：整数

- 默认值：0

指定分子体系的净电荷数。

### 7.4.4 Spinmulti 参数类型：整数

- 默认值：对偶数电子体系为 1，对奇数电子体系为 2

指定分子体系的自旋多重度。自旋多重度定义为  $2S+1$ （其中  $S$  是自旋角动量），可以由  $l$  自旋向上的单电子数 - 自旋向下的单电子数  $l+1$  计算得到，因此当体系的所有单电子的自旋都彼此平行时，自旋多重度等于体系内的单电子数加 1。

### 7.4.5 Occupy 参数类型：整数数组

指定每个不可约表示分子轨道中双电子占据的轨道数目，用于 RHF/RKS 计算。

### 7.4.6 Alpha 参数类型：整数数组

见下述 Beta 条目。

### 7.4.7 Beta 参数类型：整数数组

Alpha 和 Beta 两个关键词必须联用，用于 UHF/UKS 计算，分别指定 alpha 或 beta 轨道每种不可约表示占据轨道数目。

### 7.4.8 Guess 参数类型：字符串

- 默认值：atom
- 可选值：atom、Hcore、Huckel、Readmo

指定初猜的类型。一般情况下 atom 较 Hcore、Huckel 好，因此正常情况下无需选择 Hcore 或 Huckel。若选择 Readmo，则程序会依次检查下述文件是否存在：

1. \$BDF\_TMPDIR/\$BDFTASK.inporb
2. \$BDF\_TMPDIR/inporb
3. \$BDF\_WORKDIR/\$BDFTASK.scforb

其中 \$BDF\_TMPDIR 为当前 BDF 计算的临时目录，\$BDF\_WORKDIR 为当前 BDF 计算的工作目录，\$BDFTASK 为当前 BDF 任务的输入文件名去掉后缀.inp 后剩余的字符串。程序会读取以上列表中第一个

存在的文件里的轨道信息，如读取失败，或读取到的轨道信息与当前计算不兼容（例如基函数数目不同），则程序会自动改为 atom 猜测。读取到的轨道会先进行 Lowdin 正交归一化，然后才用于 SCF 迭代。

**Hint:** 读取的轨道文件必须和当前计算在以下方面相符：

1. 原子的数目和种类必须相同；
2. 原子的排列顺序必须相同；
3. 使用的点群必须相同；
4. 使用的基组必须相同；
5. 要么两个计算均为 RHF、RKS、ROHF 或 ROKS，要么两个计算均为 UHF 或 UKS。

除此之外的大多数方面不要求相同，例如原子坐标、电荷、自旋多重度、泛函等等均可以不同。其中如果 (1)、(2)、(3)、(5) 均满足，只有 (4) 不满足，可以用 `expandmo` 模块将轨道文件所用的基组投影到当前计算所用基组上，再读取轨道作为初猜（参见 `expandmo`）。

例如，假如某输入文件 `mol-B3LYP-Energy.inp` 在 B3LYP/def2-TZVP 水平下计算了某分子在某个结构下的单点能，现改用 M06-2X/def2-TZVP 计算同一个分子在另一个结构下的单点能（输入文件名为 `mol-M062X-Energy.inp`），则为节约计算时间，可以利用此前 B3LYP/def2-TZVP 水平下的收敛的 SCF 波函数：

```
cp mol-B3LYP-Energy.scforb mol-M062X-Energy.scforb
```

并在 `mol-M062X-Energy.inp` 的 `$scf` 块里添加

```
guess
readmo
```

此时运行 `mol-M062X-Energy.inp`，即可读取 B3LYP 单点计算的波函数作为初猜（尽管 B3LYP 单点计算所用的分子结构和当前计算不同，泛函也不相同）。

#### 7.4.9 *Mixorb* 参数类型：整数/浮点数组

将初猜轨道按一定比例进行混合。`Mixorb` 后的第一行是一个整数（以下记为  $N$ ），表示需要混合的轨道对的数目；第 2 行到第  $N+1$  行每行为 5 个数，给出需要混合的轨道对的信息。其中每一行的第一个数表示混合的是 alpha 还是 beta 轨道（1 为 alpha，2 为 beta；对于 RHF/RKS/ROHF/ROKS 计算，该数必须为 1）；第二个数表示待混合轨道的不可约表示编号（对于不考虑点群对称性的计算，该数必须为 1）；第三、第四个数表示待混合轨道在所指定不可约表示下的序号；第五个数（以下记为  $\theta$ ，单位：度）表示将这两个轨道按以下公式进行混合：

- 新的第一个轨道 =  $\cos \theta \times$  原来的第一个轨道 +  $\sin \theta \times$  原来的第二个轨道
- 新的第二个轨道 =  $\sin \theta \times$  原来的第一个轨道 -  $\cos \theta \times$  原来的第二个轨道

一般使用较多的是  $\theta = 45$  和  $\theta = 90$  的情况，由以上公式可以看出， $\theta = 45$  相当于把两个轨道按等比例混合，得到一个同相位组合轨道和一个反相位组合轨道； $\theta = 90$  相当于把两个轨道交换。以下算例将第 3 个不可约表示的第 10 个 beta 轨道和第 11 个 beta 轨道进行等比例混合（例如为了进行自旋对称性破缺的计算）：

```
$scf
UHF
guess
  readmo
mixorb
  1
  2,3,10,11,45
$end
```

以下算例将第 5 个不可约表示的第 7 个轨道和第 8 个轨道交换，同时还将第 6 个不可约表示的第 3 个轨道和第 4 个轨道交换：

```
$scf
ROHF
guess
  readmo
mixorb
  2
  1,5,7,8,90
  1,6,3,4,90
$end
```

注意一般只有在 `Guess` 设定为 `Readmo` 时，才能使用 `Mixorb`，否则用户撰写输入文件时不清楚初猜轨道的成分，因此无法知道应当混合哪些轨道。

#### DFT 交换相关泛函关键词

### 7.4.10 DFT 参数类型：字符串

指定 DFT 计算的交换相关泛函。参见 BDF 支持的交换相关泛函列表。

### 7.4.11 D3 参数类型：Bool 型

指定对 DFT 计算加入 Grimme 的 D3 色散矫正。

### 7.4.12 *FACEX* 参数类型：浮点型

指定泛函的 HF 交换项比例。注意目前只有 SVWN、SVWN5、PBE、PBE0、PW91、BP86、BLYP、B3LYP、GB3LYP、B3PW91、BHLYP、SF5050、B2PLYP 泛函允许用户自定义 FACEX。例如以下输入将 PBE 的 HF 交换项比例由默认的 0% 改为 37.5%，得到 PBE38 泛函：

```
$scf
...
DFT
  PBE
facex
  0.375
$end
```

### 7.4.13 *FACCO* 参数类型：浮点型

指定泛函的 MP2 相关项比例。注意目前只有 B2PLYP 泛函允许用户自定义 FACCO。例如以下输入通过改变 B2PLYP 的 FACEX 和 FACCO，同时自定义 MP2 模块里的 spin component scaling 参数 FSS 和 FOS（参见 mp2），自定义了 DSD-BLYP 泛函：

```
$scf
...
dft
  B2PLYP
facex
  0.75
facco
  0.47
$end

$mp2
fss
  0.60
fos
  0.46
$end
```

#### 7.4.14 *RSOMEGA* / *RS* 参数类型：浮点型

指定 Range-Separated 泛函如 CAM-B3LYP 等的  $\omega$  (某些文献称  $\mu$ ) 系数。RS 是 RSOMEGA 的同义词。本关键词在 `scf` 模块中仅用于调试，建议在 `xuanyuan` 模块中设置。

DFT 数值积分格点控制参数关键词

#### 7.4.15 *NPTRAD* 参数类型：整型

指定数值积分的径向格点数。本参数一般用于调试程序，正常计算不需要指定该参数。

#### 7.4.16 *NPTANG* 参数类型：整型

指定数值积分的角向格点数。本参数一般用于调试程序，正常计算不需要指定该参数。

#### 7.4.17 *Grid* 参数类型：字符串

- 默认值：Medium
- 可选值：Ultra Coarse、Coarse、Medium、Fine、Ultra Fine、SG1

指定 DFT 计算的格点类型。

#### 7.4.18 *Gridtype* 参数类型：整型

- 默认值：0
- 可选值：0、1、2、3

指定 DFT 计算的径向与角向布点方法。

#### 7.4.19 *Partitiontype* 参数类型：整型

- 默认值：1
- 可选值：0、1

指定 DFT 格点分割类型。0 为 Becke 分割；1 为 Stratmann-Scuseria-Frisch 分割。一般用户无需改变该参数。

#### 7.4.20 *Numinttype* 参数类型: 整型

- 默认值: 0

指定数值积分计算方法。本参数一般用于调试程序, 正常计算不需要指定该参数。

#### 7.4.21 *NosymGrid* 参数类型: Bool 型

指定数值积分不使用分子对称性, 仅用于程序调试。

#### 7.4.22 *DirectGrid / NoDirectGrid* 参数类型: Bool 型

指定数值积分采用直接积分的模式, 不保存基组值等信息。对于 *DirectSCF* 必须使用 *DirectGrid*。只有非 *DirectSCF* 情况下才有必要使用 *NoDirectGrid*。本参数一般用于调试程序, 正常计算不需要指定该参数。

#### 7.4.23 *NoGridSwitch* 参数类型: Bool 型

指定数值积分过程不变换格点。为了降低计算量, BDF 默认使用 *ultra coarse* 类型格点迭代几次 DFT, 到了一定的阈值, 再使用用户设置的积分格点。*NoGridSwitch* 参数强制不变换积分格点。

#### 7.4.24 *ThreshRho & ThreshBSS* 参数类型: 浮点型

控制积分格点的预筛选精度, 仅用于程序调试。

SCF 加速算法

#### 7.4.25 *MPEC+COSX* 参数类型: Bool 型

指定利用多级展开库伦势 (Multipole expansion of Coulomb potential, MPEC) 方法计算 J 矩阵, *COSX* (Chain-of-sphere exchange) 方法计算 K 矩阵。在 *Scf* 模块中保留该关键词只是为了向下兼容, 建议在 *Compass* 模块中设定该关键词。

#### 7.4.26 *Coulpot* 参数类型: 整型

- 默认值: 0
- 可选值: 0、1、2

控制 MPEC 计算产生库伦势  $V_c$  与原子核吸引势  $V_n$  矩阵的方法。0 为利用解析积分计算  $V_c$  与  $V_n$ ; 1 为利用多级展开计算  $V_c$ , 利用解析积分计算  $V_n$ ; 2 为利用多级展开计算  $V_c$ , 数值积分计算  $V_n$ 。

#### 7.4.27 *Coulpotlmax* 参数类型: 整型

- 默认值: 8

定义 MPEC 方法多级展开最高的角动量 L 值。

#### 7.4.28 *Coulpottol* 参数类型: 整型

- 默认值: 8, 含义为 1.0E-8

定义多级展开的精度阈值, 越大越精确。

#### 7.4.29 *MPEC* 参数类型: Bool 型

指定用 MPEC 方法计算 J 矩阵。

#### 7.4.30 *COSX* 参数类型: Bool 型

指定用 COSX 方法计算 K 矩阵。

SCF 收敛控制关键词

#### 7.4.31 *Maxiter* 参数类型: 整型

- 默认值: 100

定义 SCF 计算的最大迭代次数。

#### 7.4.32 *Vshift* 参数类型: 浮点型

- 默认值: 0
- 可选值: 非负实数
- 建议范围 (当取值不为 0 时): 0.2~1.0

指定分子轨道能级移动值。人为地将虚轨道能量加上用户指定数值, 以加大 HOMO-LUMO 能隙, 加速收敛。Vshift 值越大, 收敛过程越不容易出现振荡, 但 Vshift 值太大会导致收敛变慢。一般只有在分子的 HOMO-LUMO 能隙较小 (如小于 2 eV), 且 SCF 迭代时能量非单调降低时, 才需要设置 Vshift。

### 7.4.33 *Damp* 参数类型：浮点型

- 默认值：0
- 可选值：大于等于 0、小于 1 的实数
- 建议范围（当取值不为 0 时）：0.5~0.99

指定本次 SCF 迭代与上次迭代的密度矩阵以一定比例混合 ( $P(i):=(1-C)*P(i)+C*P(i-1)$ )，从而加速 SCF 收敛。Damp 值越大，收敛过程越不容易出现振荡，但 Damp 值太大会导致收敛变慢。一般只有在 SCF 迭代能量非单调降低的时候，才需要设置 Damp。

### 7.4.34 *ThrEne* 参数类型：浮点型

- 默认值：1.d-8

指定 SCF 收敛的能量阈值（单位：Hartree）。

### 7.4.35 *ThrDen* 参数类型：浮点型

- 默认值：5.d-6

指定 SCF 收敛的均方根密度矩阵元阈值。

### 7.4.36 *ThreshConv* 参数类型：浮点型

同时指定 SCF 收敛的能量和密度矩阵阈值。例：

```
$scf
...
ThreshConv
  1.d-6 1.d-4
$end
```

等价于

```
$scf
...
ThrEne
  1.d-6
ThrDen
  1.d-4
$end
```

### 7.4.37 *NoDiis* 参数类型: Bool 型

指定不使用 DIIS 加速 SCF 收敛。一般只有在 SCF 能量以较大幅度 ( $> 1.d-5$ ) 振荡不收敛, 且 Damp 和 Vshift 效果不明显时, 才需要指定 NoDiis。

### 7.4.38 *MaxDiis* 参数类型: 整型

- 默认值: 8

指定 DIIS 方法的子空间维数。

### 7.4.39 *SMH* 参数类型: Bool 型

指定使用 Semiempirical Model Hamiltonian (SMH) 方法加速 SCF 收敛。该方法对于一般的有机体系, 可节省约 10~15 % 的 SCF 迭代步数, 对于具有显著电荷转移、自旋极化的体系, 加速收敛效果更为显著。此外该方法还可增加收敛到稳定波函数的概率。对于满足以下情况之一的计算, 不支持使用 SMH: (1) ROHF/ROKS 计算; (2) 用户指定了 Smartemp 时; (3) 基组存在线性相关时。除这些情况外, SMH 一律默认开启。

### 7.4.40 *NoSMH* 参数类型: Bool 型

指定不使用 SMH 方法加速 SCF 收敛。

### 7.4.41 *Smartemp* 参数类型: 浮点型

- 默认值: 0
- 可选值: 非负实数 (单位: Kelvin)

指定体系的电子温度, 也即通过费米展宽 (Fermi Smearing) 方法改变前线轨道的占据数。注意 BDF 如果使用 Fermi Smearing 方法, 最终的能量包含了电子熵能 (the electronic entropy) 的贡献, 名为 -TS\_ele, 从 E\_tot 中间减掉这一项 (注意这一项是负的, 也就是说需要加上这一项的绝对值) 可以得到电子能量。Smartemp 不可与 Vshift 或 SMH 同时使用, 也不可在 FLMO 或 iOI 计算中使用。

该关键词主要有以下几类应用场景:

- 用于研究温度对电子结构的影响, 以及由此导致的对能量、各种性质的影响。例如将 Smartemp 设为 1000 进行结构优化, 可以得到 1000 K 下分子的平衡结构, 理论上会和 0 K 下的平衡结构有少许区别。注意大部分实验 (如 X 射线单晶衍射、微波光谱等) 测得的结构是热平均结构而不是平衡结构, 而热平均结构对温度的敏感性远较平衡结构更高, 所以用户不应盲目利用 Smartemp 关键词试图重现实验上观察到的分子结构随温度的变化情况, 除非已知所用实验手段测得的是平衡结构。
- 对于 HOMO-LUMO 能隙非常小或者前线轨道能级简并的体系, 该方法能改善 DFT 的收敛性, 但会轻微改变收敛的结果, 为了得到和 0 K 下相同的 SCF 结果, 需要在 Fermi Smearing 计算收敛或几乎收敛后, 读取波函数作为初猜, 继续做一个不使用 Fermi Smearing 的计算。一般为了达到明显改进收敛的效

果，需要设定较高的电子温度，如对纯泛函设定为 5000 K 左右，对杂化泛函设定为 10000 K 左右，对 HF 设定为 20000 K 左右。

- 对于 HF 或 DFT 破坏分子空间对称性的情况，Smeartemp 有助于得到符合空间对称性的轨道。例如环丁二烯的 Kohn-Sham 波函数仅有  $D_{2h}$  对称性，但在适当的电子温度下计算，可以得到符合  $D_{4h}$  对称性的轨道。

**Fock 矩阵对角化控制关键词**

#### 7.4.42 Sylv 参数类型：Bool 型

控制在 SCF 迭代中利用求解 Sylvester 方程的方法进行块对角化，代替全对角化，以节省计算时间。例如：

```
$scf
...
sylv
$end
```

对于特别大的体系（例如原子数大于 1000、基函数数目大于 10000）的计算，Fock 矩阵对角化占总计算的时间常常不可忽略，此时以上写法通常可以降低计算量，因为用较快的块对角化代替了全对角化，并且可以充分利用 Fock 矩阵的稀疏性加速计算。但需要注意的是，此时 SCF 收敛得到的轨道不是正则轨道（特别地，当初猜为 FLMO、iOI 等计算得到的局域轨道时，收敛的轨道也是局域轨道），不过收敛的占据轨道张成的空间和正则占据轨道张成的空间相同，能量、密度矩阵等也和传统全对角化计算的结果一致。如需要得到正则轨道，应当另写一个不带 sylv 关键词的 BDF 输入文件，读取当前计算的收敛轨道作为初猜，进行全对角化计算。

#### 7.4.43 Iviop 参数类型：整型

- 默认值：无
- 可选值：1~3
- 建议值：1

控制在 SCF 迭代中使用 iVI 方法，需要与 Blkiop=7 联用。

#### 7.4.44 Blkiop 参数类型：整型

- 默认值：当指定 Sylv 时，默认值为 3，否则无默认值
- 可选值：1~8，分别代表 SAI、DDS、DNR、DGN、FNR、FGN、iVI、CHC
- 建议值：3

指定块对角化的方法，通常用于 iVI 或 FLMO 计算。如不指定该关键词，默认进行全对角化。

**打印与分子轨道输出控制参数**

#### 7.4.45 *Print* 参数类型：整型

- 默认值：0
- 可选值：0、1

仅用于程序调试，控制 SCF 的打印级别。

#### 7.4.46 *IpvtMo* 参数类型：整型

- 默认值：0
- 可选值：0、1、2

控制是否打印分子轨道系数。若设为 0，不打印分子轨道；若设为 1（默认），打印前线轨道（每个不可约表示的 HOMO-5 到 LUMO+5）的占据数、能量、系数；若设为 2，打印所有轨道的占据数、能量、系数。

#### 7.4.47 *Noscforb* 参数类型：Bool 型

强制不将分子轨道存入.scforb 文件。

#### 7.4.48 *Pyscforb* 参数类型：Bool 型

控制将 SCF 收敛轨道存储为 Pyscf 轨道格式。

#### 7.4.49 *Molden* 参数类型：Bool 型

控制将分子轨道输出为 Molden 格式，以做后续的波函数分析。

##### 相对论单电子性质计算

相对论单电子性质计算支持 sf-X2C 哈密顿及其局域变体（*xuanyuan* 模块中 *Heff* 设置为 21, 22, 或 23）。

#### 7.4.50 *Reled* 参数类型：整型

对于原子序数大于等于此值的元素计算 **有效接触密度**。无默认值。必须结合 *xuanyuan* 模块中的有限核模型 *nuclear = 1* 一起使用。

基因组线性相关检查关键词

### 7.4.51 *Checklin* 参数类型: Bool 型

强制 SCF 进行基组线性相关检查。BDF 默认对 DirectSCF 进行基组线性相关检查，以提高使用弥散基函数时的 SCF 收敛性。

### 7.4.52 *Tollin* 参数类型: 浮点型

- 默认值: 1.D-7

控制基组线性相关检查的阈值。

#### **mom** 方法控制关键词

mom 是一种  $\Delta$ SCF 方法，可以通过强制 SCF 每次迭代的占据轨道与初始占据轨道最大重叠来使 SCF 收敛到激发态。mom 方法通常比基态收敛困难。

### 7.4.53 *laufbau* 参数类型: 整型

- 默认值: 当用户设定了 Occupy、Alpha 或 Beta 时为 0，否则为 1
- 可选值: 0、1、2、3

定义用什么方法指定轨道占据数。0 表示轨道占据数始终与初猜一致；1 表示按照 Aufbau 规则指定轨道占据数；2 表示按照 mom 方法指定轨道占据数，即令占据数尽可能和初始猜测轨道保持一致，结合前述 Mixorb 关键词可以实现用 DeltaSCF 方法计算激发态；3 用于程序调试，正式计算一般无需使用。

### 7.4.54 *IfPair* & *hpalpha* & *hpbeta* 参数类型: 整型

Ifpair 参数指定电子如何激发，确定 mom 方法的电子占初态，必须与 hpalpha 和 hpbeta 参数联用。电子激发通过相对于基态通过指定从占据轨道到虚轨道的激发确定。

```
# 一个分子，其分子轨道分属 4 个不可约表示，我们想激发不可约表示 1 的 alpha 分子轨道 5、6 上的电子
# 到 alpha 轨道 7、8，不可约表示 3 的 alpha 轨道 3、4 的电子到不可约表示 1 的轨道 7、8
$scf
Ifpair
Hpalpha
2
5 0 3 0
8 0 4 0
6 0 0 0
9 0 0 0
Hbeta
1
7 0 0 0
8 0 0 0
```

(continues on next page)

(continued from previous page)

```
...
$end
```

### 7.4.55 *Pinalpha & Pinbeta* 参数类型: 整型

指定固定的分子轨道。

## 7.5 含时密度泛函 - TDDFT 模块

TDDFT 模块基于线性响应理论, 通过求解 Casida 方程计算分子激发态。TDDFT 模块支持 TDDFT (包括 TDHF)、TDA (包括 CIS) 等方法, 可以处理基态是闭壳层或者是开壳层的分子体系。其中基态是开壳层的体系不仅可以用传统的 U-TDDFT 方法处理, 还可以用自旋匹配的 SA-TDDFT (又称 X-TDDFT) 来处理, 该方法是 BDF 的特色。除此之外, BDF 还支持自旋翻转 (SF-) TDDFT 方法, 包括自旋向上翻转 TDDFT 和自旋向下翻转 TDDFT, 可以用来计算和基态自旋多重度不同的激发态。

常用关键词

### 7.5.1 *Imethod* 参数类型: 整型

- 默认值: 当参考态为 RHF/RKS 时为 1, 否则为 2
- 可选值: 1、2

指定基于哪种基态计算方法进行 TDDFT 计算。1 为 R-TDDFT, 基态为 RHF/RKS 参考态; 2 为 U-TDDFT, 基态为 UHF/UKS 参考态。其中自旋匹配 X-TDDFT 需要从 ROKS/ROHF 出发, 采用 U-TDDFT, 利用 `imethod=2, itest=1, icorrect=1` 计算 (见下)。该参数用户一般无需手动指定, 程序会自动选取合理的默认值。注意对于 U-TDDFT 和 X-TDDFT, 程序只支持在阿贝尔点群下进行计算。

### 7.5.2 *Isf* 参数类型: 整型

- 默认值: 0
- 可选值: 0、1、-1

控制是否进行 spin-flip 的 TDDFT 计算。0 为 no spin-flip (或称 spin-conserving, 计算磁量子数  $M_s$  与基态相同的激发态); 1 为 spin flip up (计算  $M_s$  比基态大 1 的激发态); -1 为 spin flip down (计算  $M_s$  比基态小 1 的激发态)。一个特例是当 `imethod=1` 且 `isf=1` 时, 程序计算的是三重态的  $M_s=0$  分量而非  $M_s=1$  分量, 因此此时不能称该计算为 spin-flip TDDFT 计算, 而仍应称之为 spin-conserving R-TDDFT 计算。注意当 `isf` 不等于 0 且 `imethod=2` 时, `itda` 必须设为 1。

### 7.5.3 *Itda* 参数类型：整型

- 默认值：0
- 可选值：0、1

控制是否使用 Tamm-Dancoff approximation (TDA)。0 为不使用 TDA 的 TDDFT 计算；1 为 TDA 计算。

### 7.5.4 *Ialda* 参数类型：整型

- 默认值：0
- 可选值：0、1、2、3、4

指定 TDDFT 交换相关核。0 为 full non-collinear kernel；1 为 non-collinear ALDA kernel；2 为 no-collinear ALDA0 kernel；3 为 full non-collinear kernel，产生自旋平均密度；4 为 full collinear kernel。

对于  $isf=0$  的计算，*ialda* 关键词不起作用。对于  $isf$  不等于 0 且参考态不是 RHF/RKS 的单点计算，*ialda* 建议设为 2，因为它的数值稳定性较默认的 0 更好。对于  $isf$  不等于 0 的 TDDFT 结构优化、TDDFT 数值频率和 NAC-TDDFT 计算，*ialda* 必须设为 4（注意：这会引入近似，导致计算结果和 *ialda* 不等于 4 时的结果不可比，且精度较 *ialda* 不等于 4 时低。也就是说， $isf$  不等于 0 时的 TDDFT 结构优化、TDDFT 数值频率的计算结果是能与 TDDFT 单点能结果互相比较的）。

### 7.5.5 *Itest & icorrect* 参数类型：整型

- 默认值：0
- 可选值：0、1

当 *Itest*、*icorrect* 参数均设置为 1，*imethod* 为 2，且参考态为 ROKS/ROHF 时，程序做 X-TDDFT 计算。

### 7.5.6 *Iact & elw & eup* 参数类型：整型，浮点型，浮点型

*Iact*=1，通过定义激发能量下限与上限，指定 TDDFT 求解某个能量窗口的激发态。*Elw* 为浮点数，能量下限，单位 eV；*Eup* 为浮点数，能量上限，单位 eV。

### 7.5.7 *Idiag* 参数类型：整型

- 默认值：1
- 可选值：1、2、3

指定 TDDFT 的对角化方法。1 为基于 Davidson 方法的迭代对角化；2 为完全对角化；3 为 iVI 对角化（不支持非阿贝尔点群）。

对于下述情况之一，建议用 *idiag*=3：

- X 射线吸收/发射光谱等涉及很高的激发态的计算（详见 `iwindow` 关键词的相关说明）；
- 计算某个能量或波长范围内的所有激发态，并且要求既不多算该范围外的激发态，又不少算该范围内的激发态（详见 `iwindow` 关键词的相关说明）。

对于下述情况，建议用 `idiag=2`：

- 分子很小，且需要的激发态数目非常多，接近或等于分子占据轨道数和虚轨道数的乘积。

对于其余情况，建议用默认的 `idiag=1`。

### 7.5.8 `Aokxc` 参数类型：Bool 型

指定基于 AO 计算交换相关 Kernel 对 TDDFT 的 Casida 矩阵的贡献。对于 AO-TDDFT 计算，默认开启 `aokxc`，因此此时无需指定 `aokxc`。

### 7.5.9 `Iguess` 参数类型：整型

- 可选值： $10*x+y$ ，其中  $x \in \{0,1,2\}$ ， $y \in \{0,1\}$
- 默认值：对于使用阿贝尔点群的 AO-TDDFT 计算为 20，其余情况下为 0

控制 TDDFT 初始猜测波函数。X=0: 对角元猜测；X=1: 从文件读入初始波函数；X=2: 紧束缚近似猜测；Y=0: 不存储 Davidson/iVI 迭代中间过程向量；Y=1: 存储 Davidson/iVI 迭代中间过程向量。

### 7.5.10 `Itrans` 参数类型：整型

- 可选值：0、1
- 默认值：0

控制是否将自旋轨道基的激发态矢量转到自旋张量基。仅当参考态为 ROKS，且后续不需要用 `$resp` 模块进行 TDDFT 梯度、激发态偶极矩等计算，也不需要计算 NTO 时，`itrans` 才可设为 1；其中，当参考态为 ROKS 且后续需要进行 TDDFT-SOC 计算时，`itrans` 必须设为 1。

收敛控制关键词

### 7.5.11 `Crit_e` 参数类型：浮点型

- 默认值： $1e-7$

指定 TDDFT 计算能量的收敛阈值（单位：Hartree）。

### 7.5.12 *Crit\_vec* 参数类型：浮点型

- 默认值：1e-5

指定 TDDFT 计算波函数的收敛阈值。

激发态数目控制关键词

### 7.5.13 *Iroot* 参数类型：整型

- 默认值：10
- 可选值：非零整数

当 *iroot*>0 时，表示每个不可约表示下计算 *iroot* 个根。当 *iroot*<0 时，表示所有不可约表示下总共计算 *liroot* 个根，由程序自动判断每个不可约表示下应该计算多少个根。注意对于简并的不可约表示，同一个态的不同简并分量按一个态处理，例如当分子存在二维表示，且 *iroot*=3 时，该不可约表示下会计算得到 3 个能量彼此不同的态。同义词：*ixit*。

### 7.5.14 *Nroot* 参数类型：整型数组

对每个不可约表示指定不同数目的根。如 *Nroot* 为 5 1 3，表示计算 5 个属于第 1 个不可约表示的激发态，1 个属于第 2 个不可约表示的激发态，和 3 个属于第 3 个不可约表示的激发态。如果同时指定 *iroot* 和 *nroot*，*nroot* 会被忽略。

### 7.5.15 *Iwindow* 参数类型：浮点数组

指定计算哪个能量/波长范围内的激发态。众所周知，当用户计算光谱时，一般关心的是计算某个能量/波长范围内的光谱，而不是计算前 N 个激发态。然而很多量化程序仅支持指定激发态的数目，因此用户不得不反复试错，逐渐加大激发态数目，直至激发态涵盖用户感兴趣的范围，这显然是极其费时费力的。而 BDF 则支持直接指定激发能/激发波长的范围，使用户无需浪费机时和精力调整激发态的数目反复重算。

*Iwindow* 的下一行应当包含两个浮点数，表示能量/波长范围，此外后面还可以加一个单位 (au/eV/nm/cm-1)，当没有给定单位时，默认单位为 eV。*Iwindow* 一般建议结合 *iVI* 方法使用 (*idiag*=3)，此时程序可以确保计算出该能量/波长范围内的所有激发态，没有任何遗漏，同时又尽量不浪费时间在计算该范围以外的激发态上面，也即如果一个激发态尚未完全收敛，程序即已确定该激发态不属于用户指定的能量/波长范围，则程序不再继续收敛该态。例如以下输入表示计算激发能在 1~5 eV 之间的所有激发态：

```
$tddft
...
idiag
3
iwindow
```

(continues on next page)

(continued from previous page)

```
1 5 eV
$end
```

当使用 Davidson 方法 (`idiag=1`) 时, `iwindow` 关键词仍然可以使用: (注意以下算例没有写 `idiag` 关键字, 这是因为 Davidson 方法是 TDDFT 模块默认的对角化方法)

```
$tddft
...
iwindow
1 5 eV
$end
```

此时输入的 1 (eV) 将被忽略, 也即程序计算 5 eV 以下的所有激发态, 而不管这些激发态是否高于 1 eV。不仅如此, 程序既不能严格保证计算出来的所有激发态都在 1~5 eV 内, 也不能严格保证所有在 1~5 eV 内的激发态都会被计算出来, 但这也意味着程序不需要花费额外的计算资源来保证没有遗漏任何 0~5 eV 以内的激发态, 因此此时计算速度往往比同样 `iwindow` 的 `iVI` 计算要快。然而当能量区间的下限非常高时 (比如在计算 X 射线吸收谱时), 例如以下输入:

```
$tddft
...
iwindow
300 305 eV
$end
```

则 Davidson 方法在计算 0~300 eV 的激发态上浪费的计算资源, 将远大于其节省的计算资源, 乃至导致 Davidson 方法对于该类情况完全无法使用。此时用户必须选择 `iVI` 方法。

---

**Hint:** `Iwindow` 不支持和 `idiag=2` 同时使用。

---

当指定 `iwindow` 时, `iroot`、`nroot` 对程序计算的激发态数目没有影响。但对于既指定了 `iwindow` 又使用了 `iVI` 的计算, `iroot`、`nroot` 对程序的内存分配仍然有一定影响; 该情况下虽然程序一般会设定合理的 `iroot`、`nroot` 值, 但极少数情况下程序设定值可能会不足, 导致程序报错 “too small `iroot/nroot`, require xxx, but only yyy provided”。此时代表程序在计算前低估了最终计算出的激发态数目, 因此在计算前预先分配的内存不足。这种情况下用户应当用 `iroot` 或 `nroot` 将当前不可约表示下的激发态数目设为大于等于 xxx 的正整数, 重新进行计算。

### 7.5.16 *Maxld* 参数类型：整型

*iVI* 的展开空间的最大维度。一般情况下程序会自动选定合理的默认值，一般足够计算使用，但有极小概率会不足。若遇到程序报错“too small ld xxx, require yyy”，应将 *maxld* 设为大于等于 *yyy* 的正整数，重新进行计算。

波函数存储关键词

### 7.5.17 *Istore* 参数类型：整型

指定将波函数存储于编号为 *istore* 的文件中，以备其他计算使用。

激发组态打印输出控制

### 7.5.18 *Nprt* 参数类型：整型

指定在计算结束后只打印前 *nprt* 个激发态的信息。当用户不指定 *nprt* 或 *nprt* 大于等于用户计算的激发态总数时，程序打印所有激发态的信息。

### 7.5.19 *Cdthrd* 参数类型：浮点型

指定打印绝对值大于 *cdthrd* 的轨道激发信息。

TD-DFT/SOC 和性质计算控制参数

### 7.5.20 *Nfiles* 参数类型：整型

读入 *nfiles* 个 TDDFT 先前计算的波函数，以进行 SOC 计算。

### 7.5.21 *Isoc* 参数类型：整型

- 默认值：1
- 可选值：1、2、3

指定 TDDFT-SOC 计算方法。1 为仅闭壳层体系计算；2 为一般的 SOC 计算；3 为仅打印各个标量态之间的 SOC 耦合矩阵元，不对角化 SOC Hamiltonian。

### 7.5.22 *lfgs* 参数类型：整型

- 默认值：0
- 可选值：0、1

指定 TDDFT-SOC 计算是否包含基态。0 为 TDDFT-SOC 计算不包含基态，此时无法得到基态和考虑了 SOC 的激发态（即旋量态）之间的跃迁偶极矩，因此无法绘制包含 SOC 校正的光谱，同时也无法计算基态的 SOC 校正，但仍可得到包含 SOC 校正的激发能；1 为 TDDFT-SOC 计算包含基态，此时可以得到包含 SOC 校正的光谱，且可以计算基态的 SOC 校正，但此时纳入 TDDFT-SOC 处理的标量激发态的数目不宜过多（一般以 10~100 个左右为宜），否则会低估基态能量，从而高估激发能。

### 7.5.23 *lmat soc* 参数类型：整型数组

指定需要计算的 SOC 矩阵元。

```

...
#SCF calculation for the singlet ground state S0.
$scf
spin
0
...
$end

#First TDDFT, singlets S1-S10.
$tddft
imethod
1
isf
0
iroot
10
....
$end

#Second TDDFT, triplets T1-T10
$tddft
imethod
1
isf
1
iroot
10
$end

```

(continues on next page)

(continued from previous page)

```

$tdfft
....
# 如果 imatsoc<0, 所有的 SOC 矩阵元都会打印;
# 如果 imatsoc=0, 不打印任何 SOC 矩阵元;
# 如果 imatsoc>0, 打印 imatsoc 个矩阵元
imatsoc
  7          # 表示计算 7 个 SOC 矩阵元, 后面的 7 行指定要计算哪 7 个 SOC 矩阵元
0 0 0 2 1 1  # 字符串 "0 0 0" 代表基态
0 0 0 2 1 2  # 3 个数字 "i m n" 代表第 "i" 次 TDDFT 计算, 第 "m" 个不可约表示的第 "n" 个态
1 1 1 2 1 1  # 计算矩阵<S1|HSOC|T1>
1 1 1 2 1 2
1 1 2 2 1 1
1 1 2 2 1 2
2 1 1 2 1 1
2 1 1 2 1 2
$end

```

### 7.5.24 *Imatrsf* 参数类型: 整型

- 默认值: 0
- 可选值: 0、-1

指定在 TDDFT-SOC 计算里, 计算标量态之间的跃迁偶极矩, *imatrsf*=-1 可以打印所有标量态间的跃迁偶极矩。

### 7.5.25 *Imatrso* 参数类型: 整型数组

指定打印考虑 SOC 之后的旋量态之间的跃迁偶极矩 (以及对应的振子强度, 和根据费米黄金规则计算得到的辐射跃迁速率常数)。

```

$TDDFT
...
Imatrso
# 指定需要打印 5 组旋量态之间的跃迁偶极矩, 后面 5 行指定打印哪些旋量态之间的跃迁偶极矩
# 如果这里指定-1, 则后面无需写任何信息, 程序逐对打印旋量态之间的跃迁偶极矩
# 如果这里指定-2, 则后面无需写任何信息, 程序逐对打印所有基态旋量态和所有激发态旋量态之间的跃迁偶极矩,
# 但不打印基态旋量态和基态旋量态之间, 以及激发态旋量态和激发态旋量态之间的跃迁偶极矩。
5
1 1
1 2
1 3
2 3

```

(continues on next page)

(continued from previous page)

```
2 4
$END
```

## 自然跃迁轨道 (Natural Transition Orbital, NTO) 分析

### 7.5.26 *Ntoanalyze* 参数类型: 整型数组

指定对 TDDFT 计算的某些态做 NTO 分析。该功能仅支持阿贝尔点群。

```
$TDDFT
istore
1          # 第一个 TDDFT 完成激发态计算, 并存储 TDDFT 波函数, 以备后续使用
$End

$TDDFT
Ntoanalyze
2          # 指定对两个态做 NTO 分析
1 3       # 指定对第 1 和第 3 个激发态做 NTO 分析
$End
```

## 内存控制参数

### 7.5.27 *Memjkop* 参数类型: 整型

控制积分直接的 TDDFT 计算 J, K 算符时的内存大小, 如果分配的内存不存储所有的 J, K 算符, TDDFT 将按照指定内存计算一次能存储的 J, K 算符数目, 通过多次积分计算完成每次迭代对角化的所有 J, K 算符计算。多次积分计算将降低计算效率。

### 7.5.28 *Imemshrink* 参数类型: 整型

- 默认值: 0
- 可选值: 0、1

控制积分直接 TDDFT 计算 J、K 算符时, OpenMP 并行对内存的使用方式。0 为不降低内存使用量; 1 为降低 OpenMP 并行内存使用量, 效率稍低。如果计算的分子体系特别大, 要求的计算根数目特别多, memjkop 参数无法在增大内存, 使用这个参数比积分多次计算效率高。

## 7.6 分子结构优化 - BDFOPT 模块

BDFOPT 模块是 BDF 程序的分子几何结构优化模块，用来寻找能量极小点、过渡态、锥形交叉点等。与其他模块不同，包含 `bdfopt` 模块的输入文件，并不是按照模块的先后顺序线性执行的，详见“快速入门”部分结构优化相关章节。

### 7.6.1 Solver 参数类型：整型

- 默认值：0
- 可选值：0、1

指定几何结构优化使用的求解器。

Solver=0，BDF 将使用外带的 DL-Find 优化器进行优化，该优化器支持在直角坐标或内坐标下，进行能量极小化、过渡态搜索、高阶鞍点搜索、锥形交叉点搜索、最小能量交叉点 (MECP) 搜索等。

Solver=1，BDF 将使用自带的优化器进行优化。

如果在冗余内坐标下（参见 ICoord 关键词）进行能量极小化、过渡态搜索，建议使用 Solver=1。

### 7.6.2 Maxcycle 参数类型：整型

指定最大优化步数。对于 DL-Find 优化器，默认值为 50；对于 BDF 优化器，默认值为  $\max(100, 6 * \text{原子数})$ 。

### 7.6.3 TolGrad 参数类型：浮点型

指定均方根梯度 (RMS Gradient) 的收敛标准，单位 Hartree/Bohr。对于 DL-Find 优化器，默认值为 2.D-4；对于 BDF 优化器，默认值为 3.D-4。该参数同时还将梯度的最大分量 (Max Gradient) 的收敛标准设为 TolGrad 的 1.5 倍。

### 7.6.4 TolEne 参数类型：浮点型

- 默认值：1.D-6

指定结构优化相邻两步能量变化的收敛标准，单位 Hartree。该参数仅对 DL-Find 优化器有效。

### 7.6.5 TolStep 参数类型：浮点型

- 默认值：1.2D-3

均方根步长 (RMS Step) 的收敛标准，单位 Bohr。该参数仅对 BDF 优化器有效。该参数同时还将步长的最大分量 (Max Step) 的收敛标准设为 TolStep 的 1.5 倍。

### 7.6.6 IOpt 参数类型：整型

- 默认值：3
- 可选值：3、10 (当 Solver=1 时)；0、1、2、3、9、10、11、12、13、20、30、51、52 (当 Solver=0 时)

指定优化目标。对于 DL-Find 优化器，该参数的意义与 DL-Find 的 IOpt 参数意义相同，用户可参见 DL-Find 手册；对于 BDF 优化器，仅支持其中的 2 个 IOpt 值，IOpt=3 (优化极小值点) 和 IOpt=10 (优化过渡态)。

### 7.6.7 Trust 参数类型：浮点型

- 默认值：0.3
- 可选值：非零实数

建议范围：0.005 ~ 0.5 或 -0.5 ~ -0.005

指定优化的置信半径 (trust radius)。当置信半径 r 设定为正数时，程序的初始置信半径将设为 r，但在随后的结构优化步骤中可能会视优化情况而动态地增加或减少置信半径。当置信半径 r 设定为负数时，程序的初始置信半径将设为 |r|，且随后的结构优化步骤中保证置信半径不会超过 |r|。

### 7.6.8 Update 参数类型：整型

- 默认值：对于极小值点优化为 3，对于过渡态优化为 2
- 可选值：0、1、2、3、9

指定几何优化过程中 Hessian 的更新方式。0 为每步均重新计算数值 Hessian；1 为 Powell 更新法 (仅 DL-Find 支持)；2 为针对过渡态的 Bofill 更新法；3 为指定 L-BFGS 更新法 (优化器为 DL-Find)，否则指定 BFGS 更新法；9 为针对极小值点的 Bofill 更新法。如选择 0 以外的值，则程序将在几何优化的第一步构造基于分子力场的初始 Hessian。

### 7.6.9 ICoord 参数类型：整型

- 可选值：0、1

本参数指定几何优化使用的坐标类型。如 ICoord=0，采用直角坐标；如 ICoord=1，采用冗余内坐标。对于 DL-Find 优化器，默认值为 0；对于 BDF 优化器，默认值为 1，且不支持 1 以外的值。

### 7.6.10 ILine 参数类型：整型

- 可选值：0、1

本参数指定是否在几何优化过程中进行线性搜索；如不进行线性搜索，则只进行二次搜索。ILine=0 表示不进行线性搜索，否则表示进行线性搜索。对于 DL-Find 优化器，默认值为 0；对于 BDF 优化器，默认值为 1。

### 7.6.11 Constrain 参数类型：整数数列

本参数指定进行约束性优化 (constrained optimization)，即在约束一个或多个键长、键角或二面角的情况下，优化分子其余的自由度。目前本参数仅支持 BDF 优化器。该关键词后面的第一行应是一个整数，表示约束的数目，设其为 N；第 2 行到第 N+1 行，每一行分别由 2~4 个整数组成。如某一行有 2 个整数，表示原子编号为这 2 个整数的原子之间的键被冻结；如某一行有 3 个整数，表示原子编号为这 3 个整数的原子之间的键角被冻结；如某一行有 4 个整数，表示原子编号为这 4 个整数的原子之间的二面角被冻结。

```
$bdfopt
Constrain
2
1 5      #1 号原子-5 号原子之间的化学键被冻结
1 4 8    #1 号原子-4 号原子-8 号原子的键角被冻结
$end
```

### 7.6.12 Hess 参数类型：字符串

- 可选值：only、init、final、init+final

指定计算数值 Hessian。如为 only，则仅计算数值 Hessian 而不做几何结构优化。如数值 Hessian 计算正常结束，程序将把 Hessian 对角化并进行热化学分析，给出振动频率、振动简正模、零点能、内能、焓、熵、Gibbs 自由能等数据。如为 init，则首先计算数值 Hessian，然后以其为初始 Hessian 进行几何结构优化。该方法主要应用于过渡态搜索中（因为默认的基于分子力场的初始 Hessian 缺乏虚频）。程序不对该 Hessian 进行热化学分析。如为 final，则首先进行结构优化，如结构优化收敛，则在收敛的几何结构上计算数值 Hessian，并进行频率分析和热化学分析。在其他量化程序中，这种计算模式常被称为 opt+freq。如为 init+final，则首先计算初始数值 Hessian，然后进行几何结构优化，优化收敛后再计算数值 Hessian。程序仅对后一个数值 Hessian 进行频率分析和热化学分析，而不对前一个数值 Hessian 进行这些分析。

### 7.6.13 *ReCalcHess* 参数类型：整型

- 可选值：非负整数

指定在几何优化中，每隔多少步计算一次数值 Hessian。如不提供该关键词，默认在几何优化过程中不计算数值 Hessian（除非指定了 Update=0）。

### 7.6.14 *NumHessStep* 参数类型：浮点型

- 默认值：0.001
- 可选值：正实数

建议范围：0.001 ~ 0.01

指定数值 Hessian 计算时，扰动分子的步长（单位：Bohr）。

### 7.6.15 *ReadHess* 参数类型：Bool 型

指定读取 \$BDFTASK.hess 作为结构优化的初始 Hessian（其中 \$BDFTASK 为当前输入文件的名字去掉后缀.inp 得到的字符串）。\$BDFTASK.hess 可以由其他的频率计算任务产生，而不一定需要和当前结构优化计算的理论级别一致。

### 7.6.16 *RestartHess* 参数类型：Bool 型

指定对频率任务进行断点续算。

### 7.6.17 *NDeg* 参数类型：整型

- 默认值：1
- 可选值：正整数

指定当前电子态的电子简并度，用于计算热化学分析中的吉布斯自由能。电子简并度等于空间简并度乘以自旋简并度，其中空间简并度等于当前电子态所属不可约表示的维数（当分子属于阿贝尔群时，空间简并度等于 1），自旋简并度对于非相对论计算和标量相对论计算等于自旋多重度 ( $2S+1$ )，而对考虑了旋轨耦合的计算等于  $2J+1$ ，其中  $J$  为当前电子态的总角动量量子数。注意即使对于电子简并度不等于 1 的体系，NDeg 的默认值仍然是 1，用户必须手动指定正确的 NDeg 值，这一点对于开壳层体系的吉布斯自由能计算尤其重要。

### 7.6.18 *Temp* 参数类型：浮点型

- 默认值：298.15
- 可选值：正实数

指定在什么温度下进行热化学分析（单位：K）。

### 7.6.19 *Press* 参数类型：浮点型

- 默认值：1.0
- 可选值：正实数

指定在什么压强下进行热化学分析（单位：atm）。

### 7.6.20 *Scale* 参数类型：浮点型

- 默认值：1.0
- 可选值：正实数

指定频率分析的校正因子。

## 7.7 Hartree-Fock Gradient - GRAD Module

The GRAD module is used to calculate the resolved gradients of HF/MCSCF.

### Basic Keywords

#### 7.7.1 *Nrootgrad* parameter type: integer

Specifies the gradient of that root for calculating MCSCF.

#### 7.7.2 *Maxiter* Parameter type: integer

Specifies the maximum number of iterations of CPMCHF.

### 7.7.3 *IntCre* Parameter type: integer

The memory size used to increase the storage of AO points is:  $\text{intcre} * 256 * 1024 * 1024 \text{Bytes}$ .

### 7.7.4 *Ishell* Parameter type: integer

### 7.7.5 *Cutcpm* parameter type: floating point

- Default value: 1.D-6

Specifies the convergence threshold for solving the CPMCHF equation.

### 7.7.6 *Printgrad* parameter type: floating point

- Selectable values: 0, 3, >99

Controls gradient printing. 0 is the minimum output; 3 is the contribution of the output single-electron term to the gradient; >99 is for debug mode only.

## 7.8 DFT/TDDFT Gradient and Response Properties - RESP Module

The resp module is used to calculate the gradient of the DFT/TDDFT, the non-adiabatic coupling matrix elements at the TDDFT theoretical level (including the non-adiabatic coupling matrix elements between the ground-state-excited state, and the non-adiabatic coupling matrix elements between the excited state-excited state), and the response properties such as the excited state dipole moment.

### Basic Keywords

### 7.8.1 *lprt* parameter type: integer

Controls the printout level, mainly used for program debugging.

### 7.8.2 *NOrder* parameter type: integer

- Default value: 1
- Optional values: 0, 1, 2

The order of the geometric derivative is currently supported only for (response properties that do not involve analytic gradients, such as excited state dipole moments) and 1 (analytic gradients), but not yet for 2 (analytic Hessian). This parameter requires that *Geom* be specified first.

### 7.8.3 *Geom* parameter type: Bool type

This keyword does not require parameters and needs to be used in conjunction with the *Norder* keyword to specify the first or second order derivative of the calculated geometric coordinates.

ptional values: 1. Gradient or fo-NACMEs; 2. Hessian (under development)

### 7.8.4 *NFiles* parameter type: integer

For TD-DFT response property calculations, specify which *\$tdft* block to read; note that when this parameter is equal to *x*, it does not simply mean that the *x*th *\$tdft* block is read, but that the *\$tdft* block with an *istore* value of *x* is read. For example, the following input for a closed-shell molecule (*\$compass*, *\$xuanyuan*, *\$scf* omitted)

```
$tdft
imethod
1
Nroot
1
istore
1
$end

$tdft
imethod
1
isf
1
Nroot
1
istore
2
$end

$resp
geom
imethod
2
nfiles
2          #Calculate the gradient of the lowest triple excited state, not the
->gradient of the lowest single excited state
          #Because nfiles=2, only the second $tdft block (lowest triple
->excitation state) istore=2
$end
```

### 7.8.5 *Imethod* parameter type: integer

- Default value: 1
- Optional values: 1, 2

Specifies whether to perform DFT ground state or TD-DFT excited state calculations. 1 is the ground state, if specified 2, then it is the excited state calculation. In older versions of BDF the keyword is written Method, currently the program supports both Imethod and Method, but in the future it may only support the former.

```
#Calculate the TD-DFT gradient of the first TD-DFT excited state
$tdfft
Nroot
1
istore
1
$end

$resp
geom
imethod
2
nfiles
1
$end
```

```
#Calculate the ground-state gradient
$resp
geom
$end
```

### 7.8.6 *Ignore* parameter type: integer

- Default value: 0
- Optional values: -1, 0, 1

Data consistency check for TDDFT gradient calculation, mainly for debugging programs.

-1: Recalculates the TDDFT excitation energy, used to check that the Resp and TDDFT modules agree on the energy calculation. For debugger use only.

0: Check if the Wmo matrix is a symmetric matrix. Theoretically, the Wmo matrix should be symmetric, but if the TDDFT or Z-Vector iterations do not converge completely, the Wmo matrix will show significant asymmetry, and the program will exit with an error and tell the user whether the Wmo matrix is asymmetric because the TDDFT did not

converge completely or the Z-Vector equation did not converge completely. Note that sometimes the asymmetry of the  $W_{mo}$  matrix can also be caused by some keyword input errors by the user.

1: Ignore the  $W_{mo}$  matrix symmetry check. The ignore setting should only be 1 set if the user has confirmed that the TDDFT and Z-vector convergence thresholds are tight enough to not affect the accuracy of the computation unacceptably, and that the keywords in the input file have been entered correctly, but the program still reports an error due to a failed symmetry check, the ignore should be set to 1.

### 7.8.7 *IRep* & *IRoot* parameter type: integer

These two keywords specify which/which state(s) of TD-DFT gradient or excited state dipole moment is/are to be calculated. There are 4 cases:

- a. Specify both *IRep* and *IRoot*: e.g. the following input

```
#Calculates the gradient or dipole moment of the 3rd root under the 2nd irreducible_
↔representation (irrep).
irep
2
iroot
3
```

- b. Specify *IRep* only: Compute the gradient or dipole moment of all roots under this integrable representation.

- c. Specify *IRoot* only: for example

```
#All the roots under irreducible representation are sorted by energy from low to high,
↔ and then the gradient or dipole moment of the 3rd root is calculated
iroot
3
```

- d. Neither is specified: calculate the gradient or dipole moment of all states obtained by tddft.

### 7.8.8 *JahnTeller* Parameter type: String

For molecules with certain symmetries, TDDFT structure optimization may lead to Jahn-Teller distortion of the molecule if the point group to which the molecule belongs is a higher-order point group, but the distortion may have multiple directions. For example, assuming that a molecule with  $I_h$  symmetry has a triple-simplex excited state  $T_{2g}$ , the symmetry of the geometry of this state may be reduced to  $D_{2h}$ ,  $D_{3d}$ ,  $D_{5d}$  or subgroups of these groups after the Jahn-Teller distortion. Therefore, in the TDDFT structure optimization, the symmetry of the molecular structure may decrease from the second optimization step. When the point group obtained by Jahn-Teller distortion is not unique, the specific Jahn-Teller distortion can be specified by the *JahnTeller* keyword. For example,

```
$resp  
...  
JahnTeller  
  D(2h)  
$End
```

The above example specifies that when there is a Jahn-Teller aberration and the aberration mode is not unique, preference is given to the aberration mode in which the aberrated structure belongs to the D2h group. If it can be deduced from group theory that the molecule will not undergo a Jahn-Teller aberration in the current electronic state, or that a Jahn-Teller aberration will occur but will not result in a structure belonging to the D2h group, the program prints a warning message and ignores the user input. If the current molecule will undergo a Jahn-Teller distortion but the user does not specify a JahnTeller keyword, the program tries to maintain the higher order symmetry axis of the molecule during the Jahn-Teller distortion. Still using the T2g state of the Ih group above as an example, if the JahnTeller keyword is not specified, the molecule will distort to a D5d structure because this is the only way to maintain the fivefold symmetry axis of the Ih group.

### 7.8.9 *Line* parameter type: Bool type

Perform resp for linear response calculation.

### 7.8.10 *Quad* parameter type: Bool type

Specify resp for secondary response calculation.

### 7.8.11 *Fnac* parameter type: Bool type

Specify resp to calculate the first-order nonadiabatic couplings vectors, which need to be used in conjunction with the Single or Double parameters to specify the calculation of the ground-state-excited state and excited-state-excited state nonadiabatic couplings vectors, respectively.

### 7.8.12 *Single* parameter type: Bool type

Specify the calculation of the ground-state-excited state non-adiabatic coupling vector.

### 7.8.13 *States* parameter type: integer 数组

Specifies which states are calculated for the non-adiabatic coupling vector to the ground state. This parameter is a multi-line parameter.

First line: Enter the integer  $n$ , specifying the non-adiabatic coupling vector between the ground state and the  $n$  excited states to be calculated.

The second line to line  $n+1$  specifies the electronic state in the format of three integers  $m\ i\ l$ .  $m$  is the file number of the previous TDDFT calculation istore specified storage,  $i$  is the  $i$ -th integrable representation, and  $l$  is the  $l$ -th root of that integrable representation.

### 7.8.14 *Double* parameter type: Bool type

Specify the excited-state excited-state non-adiabatic coupling vector for calculation.

### 7.8.15 *Pairs* parameter type: integer 数组

Specifies which set of two excited states to calculate the non-adiabatic coupling vector between. This parameter is a multi-line parameter :

First line: Enter an integer  $n$ , specifying that the non-adiabatic coupling vector between  $n$  pairs of excited states is to be calculated.

The second to  $n+1$  lines specify the electronic states in the format  $m1\ i1\ l1\ m2\ i2\ l2$  six integers, with each three integers specifying an excited state.  $m1$  is the file number of the storage specified by the previous TDDFT calculation istore,  $i1$  is the  $i1$ st integrable representation, and  $l1$  is the  $l1$ st root of that integrable representation. The other three integers are the same.

### 7.8.16 *Noresp* parameter type: Bool type

Specifies that the response term of the leap density matrix is ignored in Double and FNAC calculations. This keyword is recommended.

## 7.9 Energy and Charge Transfer - ELECOUP Module

Energy and Charge Transfer - ELECOUP Module :

1. Calculating the coupling integral between two electronic states of the same molecule based on HF;
2. Calculate the charge migration integral between two molecular sheets;
3. Calculate the energy migration integral between the excited states of two molecular sheets.

---

**Note:** The HF-based calculation of the coupling integral between two excited states of the same molecule requires the use of the  $\Delta$ SCF method, usually using the mom function in the SCF module.

---

### 7.9.1 *Ipvt* parameter type: integer

Print control parameters for debugging programs only.

### 7.9.2 *UHF* parameter type: Bool type

The coupling integral between the two electronic states is based on the UHF wave function.

### 7.9.3 *Nexcit* parameter type: integer

Specify the number of excited states per molecule.

### 7.9.4 *GSApr* parameter type: Bool type

Specifies whether to approximate the base state processing.

**Calculate charge migration integral keywords**

### 7.9.5 *Electrans* parameter type: integer array

This parameter is a multi-line parameter that specifies a number of pairs of Donor and Acceptor molecular orbitals and calculates the charge migration integral between them. Format as follows.

First line: enter the integer  $n$ , specifying that the migration integral between  $n$  pairs of orbits is to be calculated

Line 2 to line  $n+1$ : input three integers  $i j k$ ,  $i$  is the  $i$ -th track of Donor,  $j$  is the  $j$ -th track of acceptor, and parameter  $k$  is 1 or 2, specifying alpha or beta tracks, respectively.

### 7.9.6 *Dft* parameter type: String

Specifies what exchange-dependent general function is used to calculate the charge migration integral. If you do not enter this parameter, the same functional is used by default as when Kohn-Sham is calculated.

**Localized excited states**

### 7.9.7 *locales* parameter type: integer

Specify the method to obtain the excited state of the localization.

- Default value: 0
- Optional values: 0, 1; 0 Boys definite domain method, 1 Ruedenberg localization methods

## 7.10 Molecular orbital localization - LOCALMO module

The LOCALMO module is used to generate delocalized molecular orbitals and contains methods such as Boys, Pipek-Mezey, and a modified Boys delocalization. LOCALMO is also used to generate the initial molecular slice delocalized orbitals for the FLMO method.

### Basic control parameters

#### 7.10.1 *Boys* parameter type: Bool type

Specifies to use the Boys delocalization method to delocalize the track. *boys* is the default method for the LOCALMO module.

#### 7.10.2 *Mboys* parameter type: Integer type

Specifies the use of the improved Boys delocalization method, the next row is an integer, and the exponential factor for the improved Boys method is specified.

#### 7.10.3 *Pipek* parameter type: Bool type

Specifies to use the Pipek-Mezey delocalization method. The default is Mulliken charge, if the Lowdin parameter is set, then the Pipek-Mezey method uses Lowdin charge instead of the default Mulliken charge. This method defaults to the Jacobi rotation domainization track, if you need to specify the Trust-Region method, you need to use the keyword Trust.

#### 7.10.4 *Mulliken* parameter type: Bool type

Specifies that the Pipek-Mezey method uses Mulliken charges. Default option.

### 7.10.5 *Lowdin* parameter type: Bool type

The parameter specifies that the Pipek-Mezey method uses the Lowdin charge.

### 7.10.6 *Jacobi* parameter type: Bool type

Specifies that the Pipek-Mezey method utilizes Jacobi rotational fixed-domain orbits.

### 7.10.7 *Trust* parameter type: Bool type

Specifies that the Pipek-Mezey method utilizes the Trust Region method for domaining tracks.

### 7.10.8 *Hybridboys* parameter type: Integer type

Selectable values: -100, 100

Specifies that the Pipek-Mezey or Boys localization method mixes Jacobi rotation with the Trust Region method to localize the track. By default, the hybrid method is not used, if this parameter is added, the next input line must be an integer. -100: Converts the virtual orbit to the Trust Region method to continue domaining only after the virtual orbit has first been domained 100 times with Jacobi rotation or after the domaining has reached the convergence threshold Hybridthre. 100: Converts the occupied and virtual orbit to the Trust Region method to continue domaining after the occupied orbit has first been domained 100 times with Jacobi rotation or after the domaining has reached the convergence threshold Hybridthre.

### 7.10.9 *Hybridthre* parameter type: floating point

Specifies the conversion threshold for the hybrid localization method.

### 7.10.10 *Thresh* parameter type: floating point

Specify the threshold for the convergence of the fixed-domain method, the input is two floating-point numbers.

### 7.10.11 *Tailcut* parameter type: floating point

- Default value: 1.D-2

Specifies the threshold value for ignoring FLMO tails.

### 7.10.12 *Threshpop* parameter type: floating point

- Default value: 1.D-1

Specifies the threshold value for Lowdin placement.

### 7.10.13 *Maxcycle* parameter type: Integer type

Specifies the maximum number of cycles allowed for Boys domainization.

### 7.10.14 *Rohfloc* parameter type: Bool type

Specify the localized ROHF/ROKS orbit.

### 7.10.15 *orbital* parameter type: string

Specifies that the file is read into the molecular track.

```
$LocalMO
Orbital
hforb      # Specifies the hforb read-in track from scaf computing storage
$End
```

### 7.10.16 *Orbread* parameter type: Bool type

Specifies that the molecular tracks are read from the text file inorb in **BDF\_TMPDIR**.

### 7.10.17 *Flmo* parameter type: Bool type

Specifies the projection LMO to pFLMO.

### 7.10.18 *Frozocc* parameter type: Integer type

Specify the number of double-occupied orbitals for indefinite domainization.

### 7.10.19 *Frozvir* parameter type: Integer type

Specify the number of virtual orbitals for indefinite domainization.

### 7.10.20 *Analyze* parameter type: Bool type

Specifies to analyze a user-given fixed-domain orbit, calculating the number of occupied-empty orbital pairs and MOS (Molecular Orbital Spread). To analyze the fixed domain orbits, a file named `bdftask.testorb` is read from `BDF_TMPDIR` and the orbits are analyzed. This orbital file is in the same format as SCF's `bdftask.scforb`, both are text files.

### 7.10.21 *lapair* parameter type: floating point

Specify the threshold value for counting the overlap of occupied-empty orbital pairs. By default, only occupied-empty orbital pairs with an absolute overlap value greater than  $1.0 \times 10^{-4}$  are counted.

### 7.10.22 *Directgrid* parameter type: Bool type

Specifies the calculation of the absolute overlap of the occupied-null orbital pair using the direct numerical integration method.

### 7.10.23 *Nolmocls* parameter type: Integer type

Specify the occupancy track of the indefinite SCF.

### 7.10.24 *Nolmovir* parameter type: Integer type

Specifies the empty orbit of the indefinite SCF.

### 7.10.25 *Moprt* parameter type: Integer type

Specify the coefficients for printing the domain-definite molecular orbitals.

## 7.11 Different Basis Set Expansion Tracks - EXPANDMO Module

The EXPANDMO module is used to extend the MO of a small basis group calculation to a large basis set MO. The extended MO can be used for the initial guesses of the SCF and also for some Dual Basis calculations. In addition, EXPANDMO can automatically construct the active space and initial guess orbitals for MCSCF calculations using the atomic valence active space.

### 7.11.1 *Overlap* parameter type: Bool type

Specify the expansion of molecular orbitals using the overlapping integrals of the small and large basis groups.

The Expandmo module dependency file is as follows :

filename	description	file format	
\$BDFTASK.chkfil1	The check file for the small basis set calculation	Binary	input file
\$BDFTASK.chkfil2	The check file for the big basis set calculation	Binary	input file
inporb	The MO file produced by the small basis set calculation	text file	input file
\$BDFTASK.exporb	The extended MO coefficient file is stored in BDF_WORKDDIR	text file	input file

```
#Calculate CH2 molecules with cc-pVDZ basis sets and extend molecular orbital
->coefficients to aug-cc-pVDZ group for initial guessing of SCF calculations
# First we perform a small basis set calculation by using CC-PVDZ.
$COMPASS
Title
  CH2 Molecule test run, cc-pvdz
Basis
  cc-pvdz
Geometry
C      0.000000      0.00000      0.31399
H      0.000000     -1.65723     -0.94197
H      0.000000      1.65723     -0.94197
End geometry
UNIT
  Bohr
Check
$END

$XUANYUAN
$END

$SCF
RHF
```

(continues on next page)

(continued from previous page)

```
Occupied
3 0 1 0
$END

#Change the name of check file.
%mv $BDF_WORKDIR/ch2.chkfil $BDF_WORKDIR/ch2.chkfil1
#Copy converged SCF orbital to work directory inporb.
%mv $BDF_WORKDIR/ch2.scforb $BDF_WORKDIR/ch2.inporb

# Then we init a large basis set calculation by using aug-CC-PVDZ
$COMPASS
Title
  CH2 Molecule test run, aug-cc-pvdz
Basis
  aug-cc-pvdz
Geometry
C      0.000000      0.00000      0.31399
H      0.000000     -1.65723     -0.94197
H      0.000000      1.65723     -0.94197
End geometry
UNIT
  Bohr
Check
$END
```

## 7.12 Møller–Plesset Second Order Perturbation - MP2 Module

Møller–Plesset second-order perturbation theory calculation module, mainly for implementing two-hybrid DFT calculations. MP2 supports both RHF and UHF reference wavefunctions for the integral undirected symmetric matched-orbit SCF-based algorithm (see Saorb keyword) and only RHF reference wavefunctions for the integral direct SCF algorithm (by default, see Skeleton keyword).

### 7.12.1 *Nature* parameter type: Bool type

Calculate the approximate density matrix and output the natural orbit.

### 7.12.2 *Molden* parameter type: Bool type

Output natural tracks as molden format files.

### 7.12.3 *lprtmo* parameter type: Integer

Controls the track output print mode.

### 7.12.4 *Fss, Fos* parameter type: floating point

- Default value: 1.0

SCS-MP2 and the spin component scaling parameter used in some two-hybrid generalized functions. After calculating the MP2 energy, the program multiplies the same-spin component by *Fss* and the opposite-spin component by *Fos*.

## 7.13 Nuclear magnetic shielding constant calculation - NMR module

NMR is used to calculate the nuclear magnetic shielding constants of molecules.

### 7.13.1 *igiao* parameter type: Integer

Specify whether to calculate the NMR shielding constant for GIAO, either 0(do not do GIAO calculations) or 1 (do GIAO calculations), the default is 0 and the input form is as follows.

```
igiao
  1
```

### 7.13.2 *icg* parameter type: Integer

Specify whether to calculate the NMR shielding constant for COMMON GAUGE (CG), either 0(do not do CG calculations) or 1 (do CG calculations), with the default being 0, entered in the following form.

```
icg
  1
```

### 7.13.3 *igatom* parameter type: Integer

Specifies the position of the canonical origin of the COMMON GAUGE calculation. The accepted input is either 0, or a value from 1 to the number of atoms, with the default being 0. When *igatom* is 0, the canonical origin of COMMON GAUGE is set at the origin of the spatial coordinates, while when *igatom* is a value from 1 to the number of atoms in the molecule, the canonical origin is set at the atomic center of the first *igatom*. The input form is as follows. .. code-block:: bdf

```
igatom 3 # Sets the specification origin at the center of the 3rd atom
```

### 7.13.4 *cgcoord* parameter type: real 3 numbers

Specifies the position of the canonical origin of the COMMON GAUGE calculation to a coordinate point in space. The default units for the input coordinates are atomic units (i.e. bohr, AU), whose units can be controlled by the parameter *cgunit*.

```
cgcoord
  1.0 0.0 0.0 # Enter 3 real numbers and place the canonical origin on a point with
↳ spatial coordinates of (1.0, 0.0, 0.0).
```

### 7.13.5 *cgunit* parameter type: string

Given the units of the *cgcoord* parameter, the default is atomic units (i.e. bohr, AU), which can be changed to angstrom by entering angstrom. For other inputs (e.g. bohr, AU, etc.), the coordinates are in atomic units.

```
cgunit
  angstrom # The units of the cgcoord coordinates, which default to atomic units,
↳ and when the input is angstrom, the canonical origin coordinate units are angstroms
# Other inputs (e.g. bohr, AU) have coordinate units in atomic units.
↳ and inputs are not case-sensitive
```



## 8.1 示例 1: 计算 SCF 能量梯度、结构优化

算例下载链接 [test003.zip](#)

```
$COMPASS
Title
  H2O Molecule test run, cc-pvdz
Basis
cc-pvdz
Geometry
O  0.000000000  0.000000000  0.369372944
H  0.000000000 -0.783975899 -0.184686472
H  0.000000000  0.783975899 -0.184686472
End geometry
$END

$XUANYUAN
$END

$SCF
RHF          #Restricted Hartree-Fock
Occupied
3 0 1 1      # 对应每个不可约表示分子轨道中双电子占据的轨道数分别为 3、0、1、1
              # 注: 如果只需要指定总电子数, 而不关心每个不可约表示分别的占据数, 则建议用
              #Charge、SpinMulti 而非 Occupied 来指定, 见后续示例 4 等
$END

$GRAD        # 计算 HF 梯度。注意 DFT 梯度需要用$RESP 而非$GRAD, 具体见示例 11
$END

$BDFOPT     # 结构优化。$BDFOPT 模块既可以写在最后, 也可以写在$COMPASS 块和$XUANYUAN 块之间
$END
```

## 8.2 示例 2: 自动识别对称性 & 指认对称性

算例下载链接 [test006.zip](#)

```

$COMPASS
Title
  C6H6 Molecule test run, CC-PVDZ
Basis
  CC-PVDZ
Geometry
C    0.000000000000000    1.394991000000000    0.000000000000000
C   -1.20809764405066    0.697495500000000    0.000000000000000
C    0.000000000000000   -1.394991000000000    0.000000000000000
C   -1.20809764405066   -0.697495500000000    0.000000000000000
C    1.20809764405066   -0.697495500000000    0.000000000000000
C    1.20809764405066    0.697495500000000    0.000000000000000
H    0.000000000000000    2.494601000000000    0.000000000000000
H   -2.16038783830606    1.247300500000000    0.000000000000000
H    0.000000000000000   -2.494601000000000    0.000000000000000
H   -2.16038783830607   -1.247300500000000    0.000000000000000
H    2.16038783830607   -1.247300500000000    0.000000000000000
H    2.16038783830606    1.247300500000000    0.000000000000000
End geometry
# 默认用最高点群计算, 即 D(6h)
$END

$xuanyuan
$end

$scf
RHF          #Restricted Hartree-Fock
$end

$COMPASS
Title
  C6H6 Molecule test run, CC-PVDZ
Basis
  CC-PVDZ
Geometry
C    0.000000000000000    1.394991000000000    0.000000000000000
C   -1.20809764405066    0.697495500000000    0.000000000000000
C    0.000000000000000   -1.394991000000000    0.000000000000000
C   -1.20809764405066   -0.697495500000000    0.000000000000000
C    1.20809764405066   -0.697495500000000    0.000000000000000

```

(continues on next page)

(continued from previous page)

```

C    1.20809764405066    0.69749550000000    0.00000000000000
H    0.00000000000000    2.49460100000000    0.00000000000000
H   -2.16038783830606    1.24730050000000    0.00000000000000
H    0.00000000000000   -2.49460100000000    0.00000000000000
H   -2.16038783830607   -1.24730050000000    0.00000000000000
H    2.16038783830607   -1.24730050000000    0.00000000000000
H    2.16038783830606    1.24730050000000    0.00000000000000

```

**End geometry**

Group

D(6h) # 指定 D6h 点群

\$END

\$xuanyuan

\$end

\$scf

RHF #Restricted Hartree-Fock

\$end

\$COMPASS

Title

C6H6 Molecule test run, CC-PVDZ

Basis

CC-PVDZ

**Geometry**

```

C    0.00000000000000    1.39499100000000    0.00000000000000
C   -1.20809764405066    0.69749550000000    0.00000000000000
C    0.00000000000000   -1.39499100000000    0.00000000000000
C   -1.20809764405066   -0.69749550000000    0.00000000000000
C    1.20809764405066   -0.69749550000000    0.00000000000000
C    1.20809764405066    0.69749550000000    0.00000000000000
H    0.00000000000000    2.49460100000000    0.00000000000000
H   -2.16038783830606    1.24730050000000    0.00000000000000
H    0.00000000000000   -2.49460100000000    0.00000000000000
H   -2.16038783830607   -1.24730050000000    0.00000000000000
H    2.16038783830607   -1.24730050000000    0.00000000000000
H    2.16038783830606    1.24730050000000    0.00000000000000

```

**End geometry**

Group

D(3h) # 指定 D3h 点群

\$END

\$xuanyuan

(continues on next page)

(continued from previous page)

```
$end

$scf
RHF
$end

$COMPASS
Title
  C6H6 Molecule test run, CC-PVDZ
Basis
  CC-PVDZ
Geometry
C    0.000000000000000    1.394991000000000    0.000000000000000
C   -1.20809764405066    0.697495500000000    0.000000000000000
C    0.000000000000000   -1.394991000000000    0.000000000000000
C   -1.20809764405066   -0.697495500000000    0.000000000000000
C    1.20809764405066   -0.697495500000000    0.000000000000000
C    1.20809764405066    0.697495500000000    0.000000000000000
H    0.000000000000000    2.494601000000000    0.000000000000000
H   -2.16038783830606    1.247300500000000    0.000000000000000
H    0.000000000000000   -2.494601000000000    0.000000000000000
H   -2.16038783830607   -1.247300500000000    0.000000000000000
H    2.16038783830607   -1.247300500000000    0.000000000000000
H    2.16038783830606    1.247300500000000    0.000000000000000
End geometry
Group
  C(6v)          # 指定 C6v 点群
$END

$xuanyuan
$end

$scf
RHF
$end

$COMPASS
Title
  C6H6 Molecule test run, CC-PVDZ
Basis
  CC-PVDZ
Geometry
C    0.000000000000000    1.394991000000000    0.000000000000000
```

(continues on next page)

(continued from previous page)

```

C   -1.20809764405066   0.69749550000000   0.00000000000000
C    0.00000000000000  -1.39499100000000   0.00000000000000
C   -1.20809764405066  -0.69749550000000   0.00000000000000
C    1.20809764405066  -0.69749550000000   0.00000000000000
C    1.20809764405066   0.69749550000000   0.00000000000000
H    0.00000000000000   2.49460100000000   0.00000000000000
H   -2.16038783830606   1.24730050000000   0.00000000000000
H    0.00000000000000  -2.49460100000000   0.00000000000000
H   -2.16038783830607  -1.24730050000000   0.00000000000000
H    2.16038783830607  -1.24730050000000   0.00000000000000
H    2.16038783830606   1.24730050000000   0.00000000000000

```

**End geometry**

Group

D(3d) # 指定 D3d 点群

\$END

\$xuanyuan

\$end

\$scf

RHF

\$end

\$COMPASS

Title

C6H6 Molecule test run, CC-PVDZ

Basis

CC-PVDZ

**Geometry**

```

C    0.00000000000000   1.39499100000000   0.00000000000000
C   -1.20809764405066   0.69749550000000   0.00000000000000
C    0.00000000000000  -1.39499100000000   0.00000000000000
C   -1.20809764405066  -0.69749550000000   0.00000000000000
C    1.20809764405066  -0.69749550000000   0.00000000000000
C    1.20809764405066   0.69749550000000   0.00000000000000
H    0.00000000000000   2.49460100000000   0.00000000000000
H   -2.16038783830606   1.24730050000000   0.00000000000000
H    0.00000000000000  -2.49460100000000   0.00000000000000
H   -2.16038783830607  -1.24730050000000   0.00000000000000
H    2.16038783830607  -1.24730050000000   0.00000000000000
H    2.16038783830606   1.24730050000000   0.00000000000000

```

**End geometry**

Group

(continues on next page)

(continued from previous page)

```
D(2h)          # 指定 D2h 点群
$END

$xuanyuan
$end

$scf
RHF
$end

$COMPASS
Title
  C6H6 Molecule test run, CC-PVDZ
Basis
  CC-PVDZ
Geometry
C    0.000000000000000    1.394991000000000    0.000000000000000
C   -1.20809764405066    0.697495500000000    0.000000000000000
C    0.000000000000000   -1.394991000000000    0.000000000000000
C   -1.20809764405066   -0.697495500000000    0.000000000000000
C    1.20809764405066   -0.697495500000000    0.000000000000000
C    1.20809764405066    0.697495500000000    0.000000000000000
H    0.000000000000000    2.494601000000000    0.000000000000000
H   -2.16038783830606    1.247300500000000    0.000000000000000
H    0.000000000000000   -2.494601000000000    0.000000000000000
H   -2.16038783830607   -1.247300500000000    0.000000000000000
H    2.16038783830607   -1.247300500000000    0.000000000000000
H    2.16038783830606    1.247300500000000    0.000000000000000
End geometry
Group
  C(2v)          # 指定 C2v 点群
$END

$xuanyuan
$end

$scf
RHF
$end

$COMPASS
Title
  C6H6 Molecule test run, CC-PVDZ
```

(continues on next page)

(continued from previous page)

```

Basis
  CC-PVDZ
Geometry
C    0.0000000000000000    1.3949910000000000    0.0000000000000000
C   -1.20809764405066    0.6974955000000000    0.0000000000000000
C    0.0000000000000000   -1.3949910000000000    0.0000000000000000
C   -1.20809764405066   -0.6974955000000000    0.0000000000000000
C    1.20809764405066   -0.6974955000000000    0.0000000000000000
C    1.20809764405066    0.6974955000000000    0.0000000000000000
H    0.0000000000000000    2.4946010000000000    0.0000000000000000
H   -2.16038783830606    1.2473005000000000    0.0000000000000000
H    0.0000000000000000   -2.4946010000000000    0.0000000000000000
H   -2.16038783830607   -1.2473005000000000    0.0000000000000000
H    2.16038783830607   -1.2473005000000000    0.0000000000000000
H    2.16038783830606    1.2473005000000000    0.0000000000000000
End geometry
Group
  C(1)          # 指定 C1 点群
$END

$xuanyuan
$end

$scf
RHF
$end

```

### 8.3 示例 3: DFT 计算

算例下载链接 [test012.zip](#)

```

$COMPASS
Title
  H2O Molecule test run, cc-pvdz
Basis
  cc-pvdz
Geometry
O  0.000000000    0.000000000    0.369372944
H  0.000000000   -0.783975899   -0.184686472
H  0.000000000    0.783975899   -0.184686472
End geometry
$END

```

(continues on next page)

(continued from previous page)

```
$XUANYUAN
RS
0.33d0      # 指定 Range-Seperated 泛函的系数
$END

$SCF
RKS          #Restricted Kohn-Sham
Occupied
3 0 1 1     # 对应每个不可约表示分子轨道中双电子占据的轨道数分别为 3、0、1、1
DFT
  CAM-B31yp  # 指定 DFT 计算的交换相关泛函
$END
```

## 8.4 示例 4: 检验非阿贝尔群和骨架矩阵法

算例下载链接 [test029.zip](#)

```
# 1st task
$COMPASS
Title
  N2 Molecule test run, CC-PVTZ
Basis
  CC-PVTZ
Geometry
N   0.0000   0.000000   1.05445
N   0.0000   0.000000  -1.05445
End geometry
Unit
  Bohr      # 指定坐标长度单位
Group
  D(2h)     # 指定 D2h 点群
$END

$xuanyuan
$end

$SCF
ROHF          #Restricted Open-shell Hartree-Fock
charge        # 电荷数 1
  1
spinmulti     # 自旋多重度 2
```

(continues on next page)

(continued from previous page)

```

2
$END

# 2nd task
$COMPASS
Title
  N2 Molecule test run, CC-PVTZ
Basis
  CC-PVTZ
# 3-21G
Geometry
N   0.0000   0.000000   1.05445
N   0.0000   0.000000  -1.05445
End geometry
Unit
  Bohr
$END

$xuanyuan
$end

$SCF
ROHF
charge
  1
spinmulti
  2
$END

```

## 8.5 示例 5: 开壳层体系

算例下载链接 [test031.zip](#)

```

$COMPASS
Title
  C2H4 Molecule test run, aug-cc-pvdz
Basis
  aug-cc-pvdz
Geometry
C           -0.66500000   0.00000000   0.00000000
C            0.66500000   0.00000000   0.00000000

```

(continues on next page)

(continued from previous page)

```
H          -1.14678878    0.96210996    0.00000000
H          -1.14678878   -0.96210996    0.00000000
H          1.14678878   -0.96210996    0.00000000
H          1.14678878    0.96210996   -0.00000000
End geometry
$END

$XUANYUAN
$END

$SCF
UHF          #Unrestricted Hartree-Fock
spinmulti
3            # 自旋多重度 3
Alpha
3 0 1 1 0 2 1 1  # 指定 alpha 或 beta 轨道每种不可约表示占据轨道数目
Beta
3 0 0 1 0 2 1 0
$END
```

## 8.6 示例 6: 势能面扫描

算例下载链接 [test032.zip](#)

```
#!/test032.bdf
HF/cc-pvdz scan

geometry
O
H 1 R1
H 1 R1 2 109.3

R1 0.8 0.05 4
end geometry
```

## 8.7 示例 7: 基于双电子积分 Cholesky 分解的 SCF 计算

算例下载链接 [test033.zip](#)

```
$COMPASS
Title
  CH2 Molecule test run, cc-pvdz
Basis
cc-pvdz
Geometry
C      0.000000      0.00000      0.31399
H      0.000000     -1.65723     -0.94197
H      0.000000      1.65723     -0.94197
End geometry
UNIT                # 指定坐标长度单位
  Bohr
Group
  C(1)              # 指定 C1 点群
$END

$XUANYUAN
$END

$SCF
RKS                #Restricted Kohn-Sham
Dft functional
SVWN5
numinttype        # 数值积分
11
$END

$XUANYUAN
Cholesky
S-CD 1.d-4        # 对双电子积分做 Cholesky 分解, 设置方法和阈值
$END

$scf
RKS
Dft functional
  SVWN5
numinttype
  11
$end
```

(continues on next page)

(continued from previous page)

```
$XUANYUAN
Cholesky
S-CD 1.d-5
$END

$scf
RKS
Dft functional
SVWN5
numinttype
11
$end

$XUANYUAN
Cholesky
S-CD 1.d-6
$END

$scf
RKS
Dft functional
SVWN5
numinttype
11
$end

$XUANYUAN
Cholesky
1C-CD 1.d-4
$END

$scf
RKS
Dft functional
SVWN5
numinttype
11
$end

$XUANYUAN
Cholesky
1C-CD 1.d-6
$END
```

(continues on next page)

(continued from previous page)

```
$scf
RKS
Dft functional
SVWN5
numinttype
11
$end
```

## 8.8 示例 8: 基于 RI-J 的 DFT 计算

算例下载链接 [test041.zip](#)

```
##### C(2v) group is used
$COMPASS
Title
  H2O Molecule test run, DEF2-SV(P)
Basis
DEF2-SV(P)
Geometry
O  0.000000000  0.000000000  0.369372944
H  0.000000000 -0.783975899 -0.184686472
H  0.000000000  0.783975899 -0.184686472
End geometry
RI-J          # 库伦拟合加速计算
  DEF2-SV(P)  # 密度拟合基组
Group
  C(2v)       # 指定 C2v 点群
$END

$XUANYUAN
$END

$SCF
RKS          #Restricted Kohn-Sham
dft functional
B3lyp
gridtype     # 指定 DFT 计算径向与角向布点方法
100
$END

$SCF
```

(continues on next page)

(continued from previous page)

```
RKS
dft functional
svwn5
gridtype
100
$END

$SCF
UKS          #Unrestricted Kohn-Sham
dft functional
B3lyp
gridtype
100
$END

$SCF
UKS
dft functional
svwn5
gridtype
100
$END

##### C(1) group is used
$COMPASS
Title
  H2O Molecule test run, DEF2-SV(P)
Basis
DEF2-SV(P)
Geometry
O  0.000000000  0.000000000  0.369372944
H  0.000000000 -0.783975899 -0.184686472
H  0.000000000  0.783975899 -0.184686472
End geometry
Check
RI-J
  DEF2-SV(P)
Group
  C(1)
$END

$XUANYUAN
$END
```

(continues on next page)

(continued from previous page)

```
$SCF
RKS
dft functional
B3lyp
gridtype
100
$END

$SCF
RKS
dft functional
svwn5
gridtype
100
$END

$SCF
UKS
dft functional
B3lyp
gridtype
100
$END

$SCF
UKS
dft functional
svwn5
gridtype
100
$END
```

## 8.9 示例 9：计算电荷转移，库仑和交换积分

算例下载链接 [test062.zip](#)

```
$COMPASS
Title
  Elecoup test run
Basis
cc-pvdz
```

(continues on next page)

```
Geometry
C      0.000000    0.000000    0.000000
C      1.332000    0.000000    0.000000
H     -0.574301   -0.928785    0.000000
H     -0.574301    0.928785    0.000000
H      1.906301    0.928785    0.000000
H      1.906301   -0.928785    0.000000

End geometry
Group
  C(1)
$END

$xuanyuan
$end

$scf
RKS                               #Restricted Kohn-Sham
dft functional
  PBE0
threshconv                         # 指定 SCF 收敛的能量和密度矩阵阈值
  1.d-10 1.d-8
$end

%cp $BDFTASK.scforb $BDF_WORKDIR/$BDFTASK.scforb1
%cp $BDFTASK.scforb $BDF_WORKDIR/$BDFTASK.scforb2

$COMPASS
Title
  Elecoup test run
Basis
  cc-pvdz
Geometry
C      0.000000    0.000000    0.000000
C      1.332000    0.000000    0.000000
H     -0.574301   -0.928785    0.000000
H     -0.574301    0.928785    0.000000
H      1.906301    0.928785    0.000000
H      1.906301   -0.928785    0.000000
C     -0.000000    0.000000    3.500000
C      1.332000   -0.000000    3.500000
H     -0.574301    0.928785    3.500000
H     -0.574301   -0.928785    3.500000
H      1.906301   -0.928785    3.500000
```

(continues on next page)

(continued from previous page)

```

H      1.906301    0.928785    3.500000
End geometry
Group
  C(1)
Nfragment
  2
$END

$xuanyuan
$end

# calculate Electron and hole transfer integrals
# Hole transfer: Donor HOMO to Acceptor HOMO
# Electron transfer: Donor LUMO to Acceptor LUMO
$elecoup
electrans
  2                                # 计算 2 对轨道间的迁移积分
  8 8 1
  9 9 1
dft
  pbe0
$END

# calculate excitation energy transfer integrals
# S-S and T-T coupling: Donor HOMO->LUMO Excitation to Acceptor HOMO->LUMO excitation
$elecoup
enertrans
  2
  8 9 8 9 1
  8 10 8 10 1
dft
  pbe0
iprint
  1
$END

$elecoup
enertrans
  2
  8 9 8 9 1
  8 10 8 10 1
dft
  pbe0

```

(continues on next page)

```
orthmo
iprint
1
$END

$xuanyuan
rs                               # 指定 Range-Seperated 泛函
0.33
$end

$elecoup
electrans
2
8 8 1
9 9 1
dft # note: this calculates CAM-B3LYP coupling matrix elements upon PBE0 orbitals
cam-b3lyp
$END

$elecoup
enertrans
2
8 9 8 9 1
8 10 8 10 1
dft
cam-b3lyp
iprint
1
$END

$elecoup
enertrans
2
8 9 8 9 1
8 10 8 10 1
dft
cam-b3lyp
orthmo
iprint
1
$END

&database
```

(continues on next page)

(continued from previous page)

```
fragment 1 6
  1 2 3 4 5 6
fragment 2 6
  7 8 9 10 11 12
&end
```

## 8.10 示例 10: 阿贝尔群对称结构的 TD-DFT 梯度计算

算例下载链接 [test063.zip](#)

```
$COMPASS
Title
  H2O Molecule test run, cc-pvdz
Basis
  cc-pvdz
Geometry
O  0.000000000  0.000000000  0.369372944
H  0.000000000 -0.783975899 -0.184686472
H  0.000000000  0.783975899 -0.184686472
End geometry
$END

$XUANYUAN
$END

$SCF
RKS          #Restricted Kohn-Sham
dft functional
  B3lyp
$END

#Full TDDFT
$TDDFT
iprint
  3
iroot          # 每一个不可约表示计算 1 个激发态
  1
istore          # 指定将 TDDFT 计算结果存储在第 1 个 TDDFT 结果文件里, 以备后续 TDDFT 梯度计算使用
  1
crit_vec        # 指定 TDDFT 计算波函数收敛阈值
  1.d-8
crit_e          # 指定 TDDFT 计算能量收敛阈值
```

(continues on next page)

(continued from previous page)

```

1.d-14
$END

$resp
geom
method          # 指定 TD-DFT 激发态计算
2
iroot            # 指定计算 $tdft 模块计算的最低的能量态 (即第 1 个态) 的梯度 (在本算例里为 1B2 态)
1
nfiles          # 此处的值 (1) 需要和以上 $TDDFT 模块设置的 istore 值一致
1
$end

```

## 8.11 示例 11: DFT 基态梯度计算

算例下载链接 [test065.zip](#)

```

$COMPASS
Title
  H2O+ grad
Basis
  cc-pvdz
Geometry
O  0.000000000  0.000000000  0.369372944
H  0.000000000 -0.783975899 -0.184686472
H  0.000000000  0.783975899 -0.184686472
End geometry
group          # 指定分子的对称点群
  c(2v)
$END

$XUANYUAN
$END

$SCF
UKS            #Unrestricted Kohn-Sham
dft            # DFT exchange-correlation functional B3LYP
B3LYP
charge
1
spinmulti     # 指定计算电子态的自旋多重度, 值为 2S+1=2
2

```

(continues on next page)

(continued from previous page)

\$END

\$resp

geom

\$end

## 8.12 示例 12: 非阿贝尔群对称性下进行 TD-DFT 梯度的计算

算例下载链接 [test068.zip](#)

```

$COMPASS
Title
  C6H6 SF-TD-DFT gradient, lowest & second lowest triplet state
Basis
  cc-pvdz
Geometry
C           1.20809735    0.69749533   -0.00000000
C           0.00000000    1.39499067   -0.00000000
C          -1.20809735    0.69749533   -0.00000000
C          -1.20809735   -0.69749533   -0.00000000
C           0.00000000   -1.39499067   -0.00000000
C           1.20809735   -0.69749533   -0.00000000
H           2.16038781    1.24730049   -0.00000000
H           0.00000000    2.49460097   -0.00000000
H          -2.16038781    1.24730049   -0.00000000
H          -2.16038781   -1.24730049   -0.00000000
H           0.00000000   -2.49460097   -0.00000000
H           2.16038781   -1.24730049   -0.00000000
End geometry
thresh      # 判断分子对称性的阈值
  medium
$END

$XUANYUAN
$END

$SCF
RKS
dft functional
  # for SF-TD-DFT, a larger amount of HF exchange is required than
  # for spin-conserving TD-DFT. Thus, for most organic molecules,
  # BHHLYP (cx=50%) is recommended over B3LYP (cx=20%).

```

(continues on next page)

```
BHHLYP
$END

$TDDFT
isf          # isf=1, spin flip up
1
iprint
3
iroot        # 每一个不可约表示计算 1 个激发态
1
istore       # save TDDFT wave function in 1st scratch file
1
ialda
4            # collinear kernel
crit_vec     # 指定 TDDFT 计算波函数收敛阈值
1.d-6
crit_e       # 指定 TDDFT 计算能量收敛阈值
1.d-8
$END

$resp
geom
method       # 指定 TD-DFT 激发态计算
2
iroot
1 2         # the first and the second lowest roots
nfiles
1
jahnteller
1           # follow irrep component 1
$end
```

### 8.13 示例 13: 基于 TDDFT 的非绝热耦合计算

算例下载链接 [test081.zip](#)

```
$compass
title
  PhCOMe
basis
  def2-SVP
geometry
```

(continues on next page)

(continued from previous page)

```

C      -0.3657620861      4.8928163606      0.0000770328
C      -2.4915224786      3.3493223987      -0.0001063823
C      -2.2618953860      0.7463412225      -0.0001958732
C      0.1436118499      -0.3999193588      -0.0000964543
C      2.2879147462      1.1871091769      0.0000824391
C      2.0183382809      3.7824607425      0.0001740921
H      -0.5627800515      6.9313968857      0.0001389666
H      -4.3630645857      4.1868310874      -0.0002094148
H      -3.9523568496      -0.4075513123      -0.0003833263
H      4.1604797959      0.3598389310      0.0001836001
H      3.6948496439      4.9629708946      0.0003304312
C      0.3897478526      -3.0915327760      -0.0002927344
O      2.5733215239      -4.1533492423      -0.0002053903
C      -1.8017552120      -4.9131221777      0.0003595831
H      -2.9771560760      -4.6352720097      1.6803279168
H      -2.9780678476      -4.6353463569      -1.6789597597
H      -1.1205416224      -6.8569277129      0.0002044899
end geometry
unit      # Set unit of length as Bohr
  bohr
nosymm
$end

$XUANYUAN
$END

$SCF
rks      # Restricted Kohn-Sham calculation
dft      # ask for bhhlyp functional
  bhhlyp
$END

$tdft
isf      # request for triplets (spin flip up)
  1
ialda    # use collinear kernel (NAC only supports collinear kernel)
  4
iroot    # 每一个不可约表示计算 2 个激发态
  2
crit_vec # 指定 TDDFT 计算波函数收敛阈值
  1.d-6
crit_e   # 指定 TDDFT 计算能量收敛阈值
  1.d-8

```

(continues on next page)

(continued from previous page)

```

istore      # 指定波函数存储, save TDDFT wave function in 1st scratch file
 1
iprt        # 指定输出信息的详略程度
 2
$end

# EX-EX NAC
$resp
iprt
 1
QUAD        # 指定 resp 进行二阶响应计算
FNAC        # 指定 resp 计算一阶非绝热耦合向量
double      # double 为激发态-激发态非绝热耦合向量
method      # 指定 TD-DFT 激发态计算
 2
nfiles
 1
pairs       # 指定计算哪两组激发态之间的非绝热耦合向量
 1
 1 1 1 1 1 2
noresp      # 指定在 Double 和 FNAC 计算中忽略跃迁密度矩阵的响应项
$end

```

## 8.14 示例 14: 限制性结构优化以及开壳层体系的 SA-TDDFT 计算

算例下载链接 [test085.zip](#)

```

$compass
title
  NO2 constrained geomopt
basis
  6-31GP
geometry
  N          -1.94323539    0.95929024    0.00000000
  O          -2.69323539    2.25832835    0.00000000
  O          -0.44323539    0.95929024    0.00000000
end geometry
thresh
  medium
$end

$bdfopt

```

(continues on next page)

(continued from previous page)

```
solver
1
constraint
1          # Number of constraints
1 2        # Fix the bond length between atom 1 and atom 2
# If more constraints are included at the same time, simply add more lines
# If angles are to be fixed, use 3 atom numbers
# If dihedrals are to be fixed, use 4 atom numbers
$end

$xuanyuan
$end

$scf
roks      #Restricted Open-shell Kohn-Sham
dft
  b3lyp
spinmulti
  2
$end

$TDDFT
imethod   #2 为 U-TDDFT
  2
itest     # must specified in SA-TDDFT
  1
icorrect  # spin-adapted correction to U-TDDFT, must be specified in SA-TDDFT
  1
iprt
  3
itda
  1
iroot
  2
istore    # save TDDFT wave function in 1st scratch file, must be specified
  1
crit_vec  # 指定 TDDFT 计算波函数收敛阈值
  1.d-6
crit_e    # 指定 TDDFT 计算能量收敛阈值
  1.d-8
gridtol   # 产生自适应格点的阈值
  1.d-7
$END
```

(continues on next page)

(continued from previous page)

```
$resp
geom
method      # 指定 TD-DFT 激发态计算
  2
nfiles
  1
iroot      # 指定计算 tddft 模块计算的第一个态的梯度
  1
$end
```

## 8.15 示例 15: 计算自旋翻转 (spin-flip) 的 TDA

算例下载链接 [test098.zip](#)

```
$COMPASS
Title
  N2+
Basis
  aug-cc-pvtz
Geometry
  N      0.00000      0.00000      0.5582
  N      0.00000      0.00000     -0.5582
End geometry
group
  d(2h)
$END

$XUANYUAN
$END

% echo "SVWN SCF "
$SCF
ROKS      #Restricted Open-shell Kohn-Sham
DFT
svwn5
charge
  1
spinmulti
  2
$END
```

(continues on next page)

(continued from previous page)

```
% echo "SVWN spin-flip TDA "  
$TDDFT  
IMETHOD      #ask for U-TDDFT  
2  
ISF            # ask for spin-flip up TDDFT calculation  
1  
ITDA          #ask for TDA  
1  
ialda  
2  
iroot  
20  
MemJKOP  
2048  
$END  
  
% echo "BLYP SCF "  
$SCF  
ROKS  
DFT  
blyp  
charge  
1  
spinmulti  
2  
$END  
  
% echo "BLYP spin-flip TDA "  
$TDDFT  
IMETHOD      # ask for U-TDDFT  
2  
ISF            # ask for spin-flip up TDDFT calculation  
1  
ITDA          #TDA  
1  
ialda  
2  
iroot  
20  
MemJKOP  
2048  
$END
```

(continues on next page)

(continued from previous page)

```
% echo "B3LYP SCF "  
$SCF  
ROKS  
DFT  
b3lyp  
charge  
1  
spinmulti  
2  
$END  
  
% echo "B3LYP spin-flip TDA "  
$TDDEFT  
IMETHOD  
2  
ISF  
1  
ITDA  
1  
ialda  
2  
iroot  
20  
MemJKOP  
2048  
$END  
  
$XUANYUAN  
rs  
0.33  
$END  
  
% echo "cam-B3LYP SCF "  
$SCF  
ROKS  
DFT  
cam-b3lyp  
charge  
1  
spinmulti  
2  
$END
```

(continues on next page)

(continued from previous page)

```
% echo "cam-B3LYP spin-flip TDA "
$TDDFT
IMETHOD
  2
ISF
  1
ITDA
  1
IDIAG
  1
ialda
  2
iroot
  20
MemJKOP
  2048
$END
```

## 8.16 示例 16: iOI 计算 (基于分片方法的大体系 SCF 计算)

算例下载链接 [test106.zip](#)

```
# autofrag: a Python-based automatic fragmentation driver. Automatically
# fragments an arbitrary molecule, and prepares the BDF input files of the
# fragments (xxx.fragmentyyy.inp) and the global system (xxx.global.inp).
$autofrag
method
  ioi # To request a conventional FLMO calculation, change ioi to flmo
nprocs
  2 # Use at most 2 parallel processes in calculating the subsystems
$end

$compass
Title
  hydroxychloroquine (diprotonated)
Basis
  6-31G(d)
Geometry # snapshot of GFN2-xTB molecular dynamics at 298 K
C   -4.2028   -1.1506    2.9497
C   -4.1974   -0.4473    4.1642
C   -3.7828    0.9065    4.1812
C   -3.4934    1.5454    2.9369
```

(continues on next page)

(continued from previous page)

C	-3.4838	0.8240	1.7363
C	-3.7584	-0.5191	1.7505
H	-4.6123	-0.8793	5.0715
C	-3.3035	3.0061	2.9269
H	-3.1684	1.2214	0.8030
H	-3.7159	-1.1988	0.9297
C	-3.1506	3.6292	4.2183
C	-3.3495	2.9087	5.3473
H	-2.8779	4.6687	4.2878
H	-3.2554	3.3937	6.3124
N	-3.5923	1.5989	5.4076
C1	-4.6402	-2.7763	3.0362
H	-3.8651	1.0100	6.1859
N	-3.3636	3.6632	1.7847
H	-3.4286	2.9775	1.0366
C	-3.5305	5.2960	-0.0482
H	-2.4848	5.4392	-0.0261
H	-3.5772	4.3876	-0.6303
C	-4.1485	6.5393	-0.7839
H	-3.8803	6.3760	-1.8559
H	-5.2124	6.5750	-0.7031
C	-3.4606	7.7754	-0.2653
H	-2.3720	7.6699	-0.3034
H	-3.7308	7.9469	0.7870
N	-3.8415	8.9938	-1.0424
H	-3.8246	8.8244	-2.0837
C	-2.7415	9.9365	-0.7484
H	-1.7736	9.4887	-0.8943
H	-2.8723	10.2143	0.3196
C	-2.7911	11.2324	-1.6563
H	-1.7773	11.3908	-2.1393
H	-3.5107	10.9108	-2.4646
H	-3.0564	12.0823	-1.1142
C	-5.1510	9.6033	-0.7836
H	-5.5290	9.1358	0.1412
H	-5.0054	10.6820	-0.6847
C	-6.2224	9.3823	-1.8639
H	-6.9636	10.1502	-1.7739
H	-5.8611	9.4210	-2.8855
O	-6.7773	8.0861	-1.6209
H	-7.5145	7.9086	-2.2227
C	-4.0308	4.9184	1.3736
H	-3.7858	5.6522	2.1906

(continues on next page)

(continued from previous page)

```

C      -5.5414      4.6280      1.3533
H      -5.8612      3.8081      0.7198
H      -5.9086      4.3451      2.3469
H      -6.1262      5.5024      1.0605
End geometry
MPEC+cosx
$end

$xuanyuan
rs # the range separation parameter omega (a.k.a. mu) of wB97X
  0.3
$end

$scf
rks
dft
  wB97X
iprt
  2
charge
  2
$end

$localmo
FLMO
$end

```

## 8.17 示例 17: 双杂化泛函基态单点能计算

算例下载链接 [test116.zip](#)

```

$compass
title
  NH3...H2O B2PLYP-D3/def2-TZVP
basis
  def2-TZVP
RI-C
  def2-TZVP # RI-MP2 auxiliary basis = def2-TZVP/C
geometry
      N              -0.6347196970      -2.4888833088      -0.0001987285
      H              -2.5637570606      -2.5802060356      -0.0187542806
      H              -0.0589873685      -3.4710591095      1.5591466837

```

(continues on next page)

(continued from previous page)

```
      H      -0.0283791648      -3.4872452297      -1.5375008955
      O      0.5661204194      2.8752419284      0.0000247838
      H      0.1735090569      1.0640211402      -0.0014981011
      H      2.3916890605      2.8947369696      -0.0002005778
end geometry
unit
  bohr
MPEC+cosx
$end

$xuanyuan
$end

$scf
rks
dft
  B2PLYP
D3
$end

$mp2
$end
```

## APPLICATIONS

## 9.1 穆斯堡尔谱

1958年，R. L. Mössbauer 在研究  $\gamma$  射线共振吸收现象时，发现了穆斯堡尔效应。同质异能位移 (isomer shift;  $\delta^{IS}$ ) 是穆斯堡尔谱的重要观测参量之一，它源于具有一定尺寸的原子核与周围电子分布之间的库仑相互作用。当原子处在不同的外界环境，导致原子核附近的库仑能发生变化，而  $\delta^{IS}$  对这种变化极其敏感，因此可以用来研究原子的氧化态，自旋态，以及配位环境。穆斯堡尔谱的另一个重要观测参量是四级分裂，来源于原子核的电四极矩与原子核周围电场梯度 (electric field gradient; EFG) 之间的相互作用。此外，当原子置于外磁场中，由于原子核存在核磁矩，穆斯堡尔谱会进一步发生 Zeeman 分裂。

$\delta^{IS}$  可以表示为重元素在测试体系 A 和参照体系 R 中“有效接触密度” (effective contact density; ED) 或“接触密度” (contact density; CD) 变化量的线性函数：

$$\begin{aligned}\delta^{IS} &= \alpha(\rho_A - \rho_R) \\ &= \alpha(\rho_A - C) + \beta\end{aligned}$$

以上两个公式称为校准方程 (calibration equation)，其中  $\delta^{IS}$  是测试体系 A 相对于参照系 R 的同质异能位移实验值，ED 或 CD 值  $\rho_A$  和  $\rho_R$  可以通过理论计算获得， $\alpha$  和  $\beta$  是待拟合的参数，其中  $\alpha$  也称为核标定常数 (nuclear calibration constant)， $C$  是任意量，一般取 ED 或 CD 的整数部分。考虑到  $\rho_R$  理论值存在误差，一般用后一个公式进行拟合。

**Note:** CD 假定原子核附近的电子密度是均匀分布的，因此可以用一个点的电子密度代替（一般取原子核中心位置的密度，但也有程序取一系列样点的密度做加权平均）；ED 考虑了电子密度的非均匀分布，原则上比前者更合理。很多程序计算的是 CD，而在 BDF 中计算 ED，二者近似满足换算关系（参见文献 [73] 的 Table S2、S3），换算因子可以吸收到  $\alpha$  中。

为了准确计算 ED 及其相对变化量，需要考虑以下两个因素：

- 原子核具有一定的尺寸分布，而默认的点电荷近似可能会导致几个数量级的误差！为此需要在 `xu-anyuan` 模块设定 `nuclear=1`。
- 需要考虑相对论效应。对于重元素这是显而易见的；即便对于某些轻元素也必须要考虑相对论效应，这是因为非相对论情况下的  $p$  电子在原子核附近没有分布（参见文献 [73] 的 Table S6），从而对  $p$  区元素的 ED 造成定性错误。在 BDF 中可以用 `sf-X2C` 哈密顿或其局域变体考虑标量相对论效应，通过 `xuanyuan` 模块中的 `heff=21`（标准 `sf-X2C`），`22` (`sf-X2C-AXR`)，或 `23` (`sf-X2C-AU`) 指认。

### 9.1.1 铁 ( $^{57}\text{Fe}$ ) 化合物的有效接触密度

$\gamma$  射线吸收和发射的几率正比于  $\exp(-E_\gamma^2)$ ，当核激发能  $E_\gamma$  超过 200 keV 后，一般很难观测到穆斯堡尔谱，由此只能对少数同位素探测穆斯堡尔谱。 $^{57}\text{Fe}$  就属于适合实验测量的同位素之一，不过在理论计算中，一般不对这些同位素进行区分。

计算 ED 需要非常陡峭（也就是高斯指数非常大）的  $s$  型高斯基函数才能准确描述电子在铁原子核附近的分布；对于存在  $p$  价电子的  $p$  区元素，还需要非常陡峭的  $p$  型高斯基函数，而基组库中的标准收缩基组通常不符合要求。建议采用 cc-pVnZ 型或 ANO 型全电子相对论基组，并把其中的  $s$  函数 ( $p$  区元素还有  $p$  函数) 进行非收缩处理。在以下的全电子相对论计算中，铁采用 ANO-R2 基组（具有三  $\zeta$  精度），并把  $s$  函数做非收缩处理，也就是删除  $s$  函数的收缩因子部分，并把收缩度从 6 改为 0，然后存为其它文件名（如 ANO-R2-ED）。由于铁没有 4  $p$  价电子， $p$  函数不需要修改。把 ANO-R2-ED 放到执行计算的目录下，供后面的计算调用（下载链接 [ano-r2-ed.zip](#)）。

我们对于铁的一系列模型体系化合物进行相对论密度泛函理论计算，泛函选取 PBE0，相对论哈密顿用 sf-X2C-AU。除了铁以外的轻元素全部用 def2-TZVPP 基组，它在 Kr 元素之前属于全电子基组，虽然是非相对论的，但用于前 18 号元素的相对论计算是允许的。自旋多重度和分子坐标来自文献 [74]。以  $\text{FeF}_6^{4-}$  为例，输入如下：

```
$compass
title
  FeF_6^4-
basis-block
  def2-tzvpp
  Fe = ANO-R2-ED
end basis
geometry # 分子直角坐标，单位：埃
  Fe -0.000035  0.000012  0.000014
  F   2.116808 -0.003546  0.032360
  F  -2.116824  0.001611 -0.030945
  F  -0.003602  2.164955  0.001902
  F   0.001648 -2.165219 -0.003295
  F   0.032586  0.003638  2.109790
  F  -0.030580 -0.001452 -2.109825
end geometry
MPEC+cosx      # 使用 MPEC+COSX 加速
$end

$xuanyuan
heff          # sf-X2C-AU; 计算 ED 必须选 21-23 中的一个
  23
nuclear       # 高斯有限核模型; ED 必须设为 1
  1
$end
```

(continues on next page)

(continued from previous page)

```

$scf
charge
  -4
spinmulti
  5
uks
dft functional
  pbe0
grid          # DFT 计算 ED 需要用精密格点
  sg1
reled
  26          # 只计算 Fe 的 ED (对于本例, 10 至 26 的整数等价)
$end

```

计算完成后, 在 SCF 布居分析信息之后可以找到 ED 结果:

```

Relativistic effective contact densities for the atoms with Za > 25
-----
      No.      Iatm      Za      RMS (fm)      Rho (a.u.)
-----
         1         1        26        3.76842        14552.68329
-----

```

以此为例, 完成其它铁化合物分子的 ED 计算 (输入文件下载链接 [ed-fe.zip](#))。ED 结果以及  $\delta^{IS}$  实验值 [74] 列于下表:

Table 9.1: 部分铁化合物的  $\delta^{IS}$  和有效接触密度

分子	2S+1	$\delta^{IS}$ (mm/s)	ED ( $bohr^{-3}$ )
$FeCl_4^{2-}$	5	+0.90	14551.76
$Fe(CN)_6^{4-}$	1	-0.02	14555.78
$FeF_6^{4-}$	5	+1.34	14552.68
$FeCl_4^-$	6	+0.19	14553.98
$Fe(CN)_6^{3-}$	2	-0.13	14556.08
$FeF_6^{3-}$	6	+0.48	14553.01
$Fe(H_2O)_6^{3+}$	6	+0.51	14554.12
$FeO_4^{2-}$	3	-0.87	14558.17
$Fe(CO)_5$	1	-0.18	14556.37

用这些数据进行拟合，得到校准方程

$$\delta^{IS} = -0.29226(\rho_A - 14550) + 1.6089, \quad R^2 = 0.85$$

可见拟合误差比较大，这可能是以下原因造成的：

1. 样本太少
2. 穆斯堡尔谱是对固态的真实体系测量的，与计算所用的气态离子模型不一致。用团簇模型、溶剂化模型 [75]、嵌入模型 [76] 可能更合适。
3. 铁的某些化合物存在强关联，需要测试其它泛函，或者换成适合描述强关联体系的方法

有了校准方程后，就可以对一些铁的体系预测  $\delta^{IS}$ 。例如交错状的二环戊二烯基铁 [77]，通过以上密度泛函理论计算得到 ED 为 14554.25 a.u.，代入校准方程得到  $\delta^{IS}$  为 0.37 mm/s，与实验值 0.53 mm/s [77] 基本接近。

### 9.1.2 计算重元素化合物有效接触密度的注意事项

对于 4d 以上的元素，经验表明默认的高斯指数还不足以描述原子核附近的电子分布，需要额外补充一些更陡峭的高斯指数。例如，选择 cc-pVnZ 型或 ANO 型标准基组中最陡峭的 4-6 个  $s$  型高斯指数  $\alpha$  ( $p$  区重元素还要考虑  $p$  型高斯指数)，它们近似满足以下线性关系：

$$\ln \alpha_i = A + iB, \quad i = 1, 2, \dots$$

通过线性拟合得到参数 A、B，再通过外推 ( $i$  的间隔取 -0.5 或 -1)，即可得到更陡峭的高斯指数。一般加入 2-5 个更陡峭的  $s$  函数、1-3 个更陡峭的  $p$  函数即可满足要求，但是要避免用  $10^{11}$  以上的高斯指数，因为这可能会造成数值不稳定。

---

## FREQUENTLY ASKED QUESTIONS

### 10.1 Restart an interrupted computing task ?

The BDF supports breakpoints for a selection of common tasks, including:

1. SCF single point energy: Use the `guess` keyword to read the molecular orbital of the last SCF iteration of the interrupted task as the initial guess. Specifically, just specify the first guess as `readmo` in the `$scf` module, and rerun the input file.

```
$scf
...
guess
  readmo
$end
```

2. TDDFT single-point energy: When the TDDFT task is interrupted and `idiag` is not equal to 2, the TDDFT excitation vector of the last TDDFT iteration (Davidson iteration when `idiag=1`, iVI iteration when `idiag=3`) of the task can be read as a first guess. In this case, when `idiag=3`, only the computation with C(1) symmetry allows for breakpoint sequencing.

The way to continue the calculation of the TDDFT task breakpoint is as follows: The TDDFT task is broken by reading the converged SCF wave function of the interrupted task with the `guess` keyword in the `$scf` module and specifying the TDDFT excitation vector of the interrupted task with `iguess` in the `$tddft` module. Assume that the input file is

```
$scf
...
$end

$tddft
...
iguess
  21
$end
```

where the tens digit 2 of `iguess=21` indicates the selection of the tight-binding initial guess (but this is not necessary for this example, i.e. the following discussion also applies to `iguess=1` or `iguess=11`), and the single digit 1 indicates that the TDDFT excitation vector for each step is written to a file (when `idiag=1`, the excitation vector is saved (when `idiag=1`, the excitation vector is stored in a file with the suffix `.dvdsonvec*`; when `idiag=3`, the excitation vector is stored in a file with the suffix `.tdx`). If the task is interrupted, the breakpoint will be continued by changing the input file to:

```
$scf
...
guess
  readmo
$end

$tddft
...
iguess
  11 # or 10, if the user is sure that the job will not be interrupted again
$end
```

where `guess readmo` is to read the tracks of the previous SCF iteration, thus avoiding wasting time to re-run the SCF iteration, and `iguess=11` with a decimal 1 means to read the TDDFT excitation vector from within the `.dvdsonvec*` or `.tdx` file as a first guess. See `tddft` subsection for details on the meaning of the various values of `iguess`.

Note that: (1) due to the characteristics of the Davidson and iVI methods, the number of TDDFT iterations saved by the above methods is less than the number of iterations saved by SCF breakpoint sequencing under the same conditions, so unless the TDDFT iteration of the previously interrupted task is close to convergence, breakpoint sequencing may not save as many iterations as computing from scratch; (2) TDDFT does not by default The TDDFT excitation vector in the iteration is saved to the hard disk; you must specify `iguess` of 1, 11 or 21 to save the TDDFT excitation vector in the iteration to the hard disk. If the previously interrupted computation does not contain this keyword, it is not possible to perform a breakpoint continuation. The main reason why the program does not save the excitation vector for each step by default is that the resulting hard disk read/write time may not be negligible, so the user needs to weigh the need to specify saving the excitation vector when performing the calculation.

3. Structure Optimization: just add the `restart` keyword to the `$compass` module, see `compass` subsection.
4. Numerical Frequency Calculation: add `restarthess` keyword in the `$bdfopt` module, see the `bdfopt` subsection.

## 10.2 How is BDF referenced ?

The first step in using a BDF is to cite the original text of the BDF program [1, 78, 79, 80] . In addition to this, the different functions of a BDF should also be used with references to the corresponding method' s article, see the section on *Citation Notes* instructions.

## 10.3 False excitation energy/complex excitation energy problem for TDDFT calculations

If the ground state wave function is unstable or if the SCF converges to a state that is not the true ground state, the TDDFT calculation will suggest the appearance of false excitation energy and, rarely, even complex excitation energy. The false and complex excitation energies have no physical significance. When using the Davidson method, the program gives a warning **Warning: Imaginary Excitation Energy!** and gives the modes of all imaginary/complex excitation energies after convergence of the iterations; when using the iVI method, the program gives a warning **Error in ETDVSI: ABBA mat is not positive! Suggest to use nH-iVI.**, and the subsequent calculation does not try to continue solving for the imaginary/complex excitation energy, but only for the real excitation energy (so when using the iVI method, one cannot conclude that the system does not have excited states with imaginary/complex excitation energy just because the final converged excitation energy is all real). In this case, the ground state wave function should be re-optimized to find a stable solution, or the TDA should be used to calculate the excitation energy.

## 10.4 Available Memory and Computational Efficiency of J and K Operators for TDDFT

The keyword **MEMJKOP** of TDDFT module can be used to set the maximum available memory for TDDFT to compute J and K operators if the number of roots required to be solved by TDDFT is large and the default memory of the program is not enough, which makes TDDFT computation less efficient. For example, if **4** roots are required to be computed, TDDFT gives the following output.

```
Maximum memory to calculate JK operator:      1024.000 M
Allow to calculate    2 roots at one pass for RPA ...
Allow to calculate    4 roots at one pass for TDA ...
```

The maximum available memory for calculating JK operators is **1024M**, where the unit is megabytes (MB). In case of RPA (i.e. TDDFT) calculations, 2 roots are allowed per integration calculation, and 4 roots are allowed for TDA calculations. If the user requires TDA calculation, one integration calculation will get JK operators of all roots, and RPA calculation needs to calculate the integration twice, which reduces the calculation efficiency. You can set **MEMJKOP** to 2048MB to increase the memory so that only one integration is computed for each iteration step. Note that the actual physical memory used is about **2048MB\*OMP\_NUM\_THREADS** , i.e. it needs to be multiplied by the number of OpenMP threads.

## 10.5 Calculating segmentation fault with available stack area memory

If a **segmentation fault** occurs in a BDF calculation, most of the time it is caused by the user not having enough memory available in the stack area, and under Linux, the available stack area memory size can be set with the **ulimit** command.

First enter the command:

```
$ulimit -a
```

The output prompt is as follows:

```
core file size          (blocks, -c) 0
data seg size          (kbytes, -d) unlimited
scheduling priority    (-e) 0
file size              (blocks, -f) unlimited
pending signals        (-i) 256378
max locked memory      (kbytes, -l) 64
max memory size        (kbytes, -m) unlimited
open files             (-n) 4096
pipe size              (512 bytes, -p) 8
POSIX message queues   (bytes, -q) 819200
real-time priority     (-r) 0
stack size             (kbytes, -s) 4096
cpu time               (seconds, -t) unlimited
max user processes     (-u) 256378
virtual memory         (kbytes, -v) unlimited
file locks             (-x) unlimited
```

Here `stack size (kbytes, -s) 4096` means that the user has 4096KB of memory available in the stack area, which is only 4 megabytes, and can be accessed with the command

```
ulimit -s unlimited
```

Set an unlimited amount of stack area memory available to the user. Many Linux distributions have limits on the stack area. Strictly speaking, there is **hard limit** and **soft limit** on the size of the stack area memory limit, and normal users only have permission to set the stack area memory smaller than the **hard limit**. If `ulimit -s unlimited` prompts an error

```
$ulimit -S
-bash: ulimit: stack size: cannot modify limit: Operation not permitted
```

You need to change the **hard limit** of memory in the available stack area with your root account or contact your system administrator to resolve the issue.

---

## 10.6 OpenMP Parallel Computing

BDF supports OpenMP parallel computing and requires the number of available OpenMP threads to be set in the run script as follows:

```
export OMP_NUM_THREADS=8
```

Here is set up to have a maximum of 8 OpenMP threads available for parallel computing.

## 10.7 OpenMP' s stack area memory size

Intel compiler available stack area memory, especially when using OpenMP parallel computing, the intel compiler puts dynamic memory from the parallel area into the stack area to obtain higher computational efficiency. Therefore, the user needs to set the size of the available stack area memory for OpenMP in the BDF run script as follows:

```
export OMP_STACKSIZE=2048M
```

Here the available stack area memory size of OpenMP is set to 2048MB. Note: If you use OpenMP for multi-thread parallelism, the total heap memory used by the system is **OMP\_STACKSIZE\*OMP\_NUM\_THREADS** .

---

**Important:** The environment variable OMP\_STACKSIZE is a general environment variable and has an overlay relationship with the special environment variables of other OpenMP runtime libraries:

KMP\_STACKSIZE (Intel OpenMP) > GOMP\_STACKSIZE (GNU OpenMP) > OMP\_STACKSIZE

Therefore, if a higher priority environment variable is set in the script, the value of OMP\_STACKSIZE will be overwritten.

---

## 10.8 Intel 2018 Edition Fortran Compiler

The Intel 2018 version of the Fortran compiler is buggy and should be avoided for compiling BDFs.

## 10.9 SCF non-convergence

See the section on dealing with non-convergence in self-consistent field calculations in the SCFTech chapter

## **10.10 SCF energy is far below the expected value (more than 1 Hartree below the expected value) or SCF energy is displayed as a string of asterisks**

This is usually caused by the basis group linear correlation problem. See the discussion of basis group linear correlation problems in the section on dealing with nonconvergence in self-consistent field calculations in the SCFTech chapter. Note that while this section focuses on solutions to problems where the SCF does not converge due to basis group linear correlation problems, these methods are also applicable in cases where the basis group linear correlation problem only causes errors in the SCF energy and does not lead to non-convergence.

## **10.11 How to use custom base groups**

See the section Custom Basis File in Gaussian-Basis-Set .

## CITATION NOTES

Use BDF for related calculations, please cite the following related literatures

1. Wenjian Liu, Gongyi Hong, Dadi Dai, Lemin Li, and Michael Dolg. The Beijing four-component density functional program package (BDF) and its application to EuO, EuS, YbO and YbS *Theoretical Chemistry Accounts*, 96(2):75–83, Jul 1997. doi:10.1007/s002140050207.
2. Yong Zhang, Bingbing Suo, Zikuan Wang, Ning Zhang, Zhendong Li, Yibo Lei, Wenli Zou, Jun Gao, Daoling Peng, Zhichen Pu, Yunlong Xiao, Qiming Sun, Fan Wang, Yongtao Ma, Xiaopeng Wang, Yang Guo, and Wenjian Liu. BDF: A relativistic electronic structure program package *The Journal of Chemical Physics*, 152(6):064113, 2020. doi:10.1063/1.5143173.
3. Wenjian Liu, Fan Wang, and Lemin Li. The Beijing Density Functional (BDF) Program Package: Methodologies and Applications *Journal of Theoretical and Computational Chemistry*, 02(02):257–272, 2003. doi:10.1142/S0219633603000471.
4. Wenjian Liu, Fan Wang, and Lemin Li. *Relativistic Density Functional Theory: The BDF Program Package*, chapter 9, pages 257–282. Volume 5. World Scientific Publishing, 2004. doi:10.1142/9789812794901\_0009.

If it involves ground state/excited state structure optimization using DFT or TDDFT methods, please additionally cite the relevant literatures below

1. Zikuan Wang, Zhendong Li, Yong Zhang, and Wenjian Liu. Analytic energy gradients of spin-adapted open-shell time-dependent density functional theory *The Journal of Chemical Physics*, 153(16):164109, 2020. doi:10.1063/5.0025428.

If it involves the use of the sf-X2C-SA-TDDFT/SOC method to calculate the spin-orbit coupling between excited states, please additionally cite the relevant literatures below

1. Zhendong Li, Bingbing Suo, Yong Zhang, Yunlong Xiao, and Wenjian Liu. Combining spin-adapted open-shell TD-DFT with spin-orbit coupling *Molecular Physics*, 111(24):3741–3755, 2013. doi:10.1080/00268976.2013.785611.

2. Zhendong Li, Yunlong Xiao, and Wenjian Liu. On the spin separation of algebraic two-component relativistic Hamiltonians *The Journal of Chemical Physics*, 137(15):154114, 2012. doi:10.1063/1.4758987.
3. Zhendong Li, Yunlong Xiao, and Wenjian Liu. On the spin separation of algebraic two-component relativistic Hamiltonians: Molecular properties *The Journal of Chemical Physics*, 141(5):054111, 2014. doi:10.1063/1.4891567.

**If it involves the calculation of non-adiabatic coupling using the NAC-TDDFT method, please additionally cite the relevant literatures below**

1. Zhendong Li and Wenjian Liu. First-order nonadiabatic coupling matrix elements between excited states: A Lagrangian formulation at the CIS, RPA, TD-HF, and TD-DFT levels *The Journal of Chemical Physics*, 141(1):014110, 2014. doi:10.1063/1.4885817.
2. Zhendong Li, Bingbing Suo, and Wenjian Liu. First order nonadiabatic coupling matrix elements between excited states: Implementation and application at the TD-DFT and pp-TDA levels *The Journal of Chemical Physics*, 141(24):244105, 2014. doi:10.1063/1.4903986.
3. Zikuan Wang, Chenyu Wu, and Wenjian Liu. NAC-TDDFT: Time-Dependent Density Functional Theory for Nonadiabatic Couplings *Accounts of Chemical Research*, 54(17):3288–3297, 2021. doi:10.1021/acs.accounts.1c00312.

**If solvation effects are involved and employed, please additionally cite the relevant literatures below**

1. Eric Cancès, Yvon Maday, and Benjamin Stamm. Domain decomposition for implicit solvation models *The Journal of Chemical Physics*, 139(5):054111, 2013. URL: <https://doi.org/10.1063/1.4816767>, arXiv:<https://doi.org/10.1063/1.4816767>, doi:10.1063/1.4816767.
2. Filippo Lipparini, Benjamin Stamm, Eric Cancès, Yvon Maday, and Benedetta Mennucci. Fast Domain Decomposition Algorithm for Continuum Solvation Models: Energy and First Derivatives *Journal of Chemical Theory and Computation*, 9(8):3637–3648, 2013. PMID: 26584117. URL: <https://doi.org/10.1021/ct400280b>, arXiv:<https://doi.org/10.1021/ct400280b>, doi:10.1021/ct400280b.
3. Filippo Lipparini, Giovanni Scalmani, Louis Lagardère, Benjamin Stamm, Eric Cancès, Yvon Maday, Jean-Philip Piquemal, Michael J. Frisch, and Benedetta Mennucci. Quantum, classical, and hybrid QM/MM calculations in solution: General implementation of the ddCOSMO linear scaling strategy *The Journal of Chemical Physics*, 141(18):184108, 2014. URL: <https://doi.org/10.1063/1.4901304>, arXiv:<https://doi.org/10.1063/1.4901304>, doi:10.1063/1.4901304.

**REFERENCES**



## BIBLIOGRAPHY

- [1] Wenjian Liu, Gongyi Hong, Dadi Dai, Lemin Li, and Michael Dolg. The Beijing four-component density functional program package (BDF) and its application to EuO, EuS, YbO and YbS *Theoretical Chemistry Accounts*, 96(2):75–83, Jul 1997. doi:10.1007/s002140050207.
- [2] Jun Gao, Wenjian Liu, Bo Song, and Chengbu Liu. Time-dependent four-component relativistic density functional theory for excitation energies *The Journal of Chemical Physics*, 121(14):6658–6666, 2004. doi:10.1063/1.1788655.
- [3] Jun Gao, Wenli Zou, Wenjian Liu, Yunlong Xiao, Daoling Peng, Bo Song, and Chengbu Liu. Time-dependent four-component relativistic density-functional theory for excitation energies. II. The exchange-correlation kernel *The Journal of Chemical Physics*, 123(5):054102, 2005. doi:10.1063/1.1940609.
- [4] Daoling Peng, Wenli Zou, and Wenjian Liu. Time-dependent quasirelativistic density-functional theory based on the zeroth-order regular approximation *The Journal of Chemical Physics*, 123(14):144101, 2005. doi:10.1063/1.2047554.
- [5] Wenhua Xu, Jianyi Ma, Daoling Peng, Wenli Zou, Wenjian Liu, and Volker Staemmler. Excited states of ReO<sub>4</sub><sup>-</sup>: A comprehensive time-dependent relativistic density functional theory study *Chemical Physics*, 356(1):219–228, 2009. Moving Frontiers in Quantum Chemistry:. doi:10.1016/j.chemphys.2008.10.011.
- [6] WenHua Xu, Yong Zhang, and WenJian Liu. Time-dependent relativistic density functional study of Yb and YbO *Science in China Series B: Chemistry*, 52(11):1945–1953, Nov 2009. doi:10.1007/s11426-009-0279-5.
- [7] Zhendong Li, Bingbing Suo, Yong Zhang, Yunlong Xiao, and Wenjian Liu. Combining spin-adapted open-shell TD-DFT with spin-orbit coupling *Molecular Physics*, 111(24):3741–3755, 2013. doi:10.1080/00268976.2013.785611.
- [8] Werner Kutzelnigg and Wenjian Liu. Quasirelativistic theory equivalent to fully relativistic theory *The Journal of Chemical Physics*, 123(24):241102, 2005. doi:10.1063/1.2137315.
- [9] Wenjian Liu and Daoling Peng. Exact two-component Hamiltonians revisited *The Journal of Chemical Physics*, 131(3):031104, 2009. doi:10.1063/1.3159445.
- [10] Wenjian Liu and Daoling Peng. Infinite-order quasirelativistic density functional method based on the exact matrix quasirelativistic theory *The Journal of Chemical Physics*, 125(4):044102, 2006. doi:10.1063/1.2222365.

- [11] Daoling Peng, Wenjian Liu, Yunlong Xiao, and Lan Cheng. Making four- and two-component relativistic density functional methods fully equivalent based on the idea of “from atoms to molecule” *The Journal of Chemical Physics*, 127(10):104106, 2007. doi:10.1063/1.2772856.
- [12] Zhendong Li, Yunlong Xiao, and Wenjian Liu. On the spin separation of algebraic two-component relativistic Hamiltonians *The Journal of Chemical Physics*, 137(15):154114, 2012. doi:10.1063/1.4758987.
- [13] Zhendong Li, Yunlong Xiao, and Wenjian Liu. On the spin separation of algebraic two-component relativistic Hamiltonians: Molecular properties *The Journal of Chemical Physics*, 141(5):054111, 2014. doi:10.1063/1.4891567.
- [14] Wenjian Liu and Ingvar Lindgren. Going beyond “no-pair relativistic quantum chemistry” *The Journal of Chemical Physics*, 139(1):014108, 2013. doi:10.1063/1.4811795.
- [15] Wenjian Liu. Advances in relativistic molecular quantum mechanics *Physics Reports*, 537(2):59–89, 2014. doi:10.1016/j.physrep.2013.11.006.
- [16] Yunlong Xiao, Daoling Peng, and Wenjian Liu. Four-component relativistic theory for nuclear magnetic shielding constants: The orbital decomposition approach *The Journal of Chemical Physics*, 126(8):081101, 2007. doi:10.1063/1.2565724.
- [17] Yunlong Xiao, Wenjian Liu, Lan Cheng, and Daoling Peng. Four-component relativistic theory for nuclear magnetic shielding constants: Critical assessments of different approaches *The Journal of Chemical Physics*, 126(21):214101, 2007. doi:10.1063/1.2736702.
- [18] Lan Cheng, Yunlong Xiao, and Wenjian Liu. Four-component relativistic theory for NMR parameters: Unified formulation and numerical assessment of different approaches *The Journal of Chemical Physics*, 130(14):144102, 2009. doi:10.1063/1.3110602.
- [19] Lan Cheng, Yunlong Xiao, and Wenjian Liu. Four-component relativistic theory for nuclear magnetic shielding: Magnetically balanced gauge-including atomic orbitals *The Journal of Chemical Physics*, 131(24):244113, 2009. doi:10.1063/1.3283036.
- [20] Qiming Sun, Wenjian Liu, Yunlong Xiao, and Lan Cheng. Exact two-component relativistic theory for nuclear magnetic resonance parameters *The Journal of Chemical Physics*, 131(8):081101, 2009. doi:10.1063/1.3216471.
- [21] Qiming Sun, Yunlong Xiao, and Wenjian Liu. Exact two-component relativistic theory for NMR parameters: General formulation and pilot application *The Journal of Chemical Physics*, 137(17):174105, 2012. doi:10.1063/1.4764042.
- [22] Yunlong Xiao, Yong Zhang, and Wenjian Liu. New Experimental NMR Shielding Scales Mapped Relativistically from NSR: Theory and Application *Journal of Chemical Theory and Computation*, 10:600–608, 01 2014. doi:10.1021/ct400950g.
- [23] Yunlong Xiao and Wenjian Liu. Body-fixed relativistic molecular Hamiltonian and its application to nuclear spin-rotation tensor: Linear molecules *The Journal of Chemical Physics*, 139(3):034113, 2013. doi:10.1063/1.4813594.
- [24] Yunlong Xiao and Wenjian Liu. Body-fixed relativistic molecular Hamiltonian and its application to nuclear spin-rotation tensor *The Journal of Chemical Physics*, 138(13):134104, 2013. doi:10.1063/1.4797496.

- [25] Yunlong Xiao, Yong Zhang, and Wenjian Liu. Relativistic theory of nuclear spin-rotation tensor with kinetically balanced rotational London orbitals *The Journal of Chemical Physics*, 141(16):164110, 2014. doi:10.1063/1.4898631.
- [26] Rundong Zhao, Yong Zhang, Yunlong Xiao, and Wenjian Liu. Exact two-component relativistic energy band theory and application *The Journal of Chemical Physics*, 144(4):044105, 2016. doi:10.1063/1.4940140.
- [27] Wenli Zou, Guina Guo, Bingbing Suo, and Wenjian Liu. Analytic Energy Gradients and Hessians of Exact Two-Component Relativistic Methods: Efficient Implementation and Extensive Applications *Journal of Chemical Theory and Computation*, 16(3):1541–1554, 2020. doi:10.1021/acs.jctc.9b01120.
- [28] Junzi Liu, Yong Zhang, and Wenjian Liu. Photoexcitation of Light-Harvesting C–P–C60 Triads: A FLMO-TD-DFT Study *Journal of Chemical Theory and Computation*, 10(6):2436–2448, 2014. doi:10.1021/ct500066t.
- [29] Fangqin Wu, Wenjian Liu, Yong Zhang, and Zhendong Li. Linear-Scaling Time-Dependent Density Functional Theory Based on the Idea of “From Fragments to Molecule” *Journal of Chemical Theory and Computation*, 7:3643–3660, 09 2011. doi:10.1021/ct200225v.
- [30] Zhendong Li, Hongyang Li, Bingbing Suo, and Wenjian Liu. Localization of Molecular Orbitals: From Fragments to Molecule *Accounts of Chemical Research*, 47(9):2758–2767, 2014. doi:10.1021/ar500082t.
- [31] Hongyang Li, Wenjian Liu, and Bingbing Suo. Localization of open-shell molecular orbitals via least change from fragments to molecule *The Journal of Chemical Physics*, 146(10):104104, 2017. doi:10.1063/1.4977929.
- [32] Minghong Yuan, Yong Zhang, Zhi Qu, Yunlong Xiao, and Wenjian Liu. Sublinear scaling quantum chemical methods for magnetic shieldings in large molecules *The Journal of Chemical Physics*, 150(15):154113, 2019. doi:10.1063/1.5083193.
- [33] Zikuan Wang and Wenjian Liu. iOI: An Iterative Orbital Interaction Approach for Solving the Self-Consistent Field Problem *Journal of Chemical Theory and Computation*, 17(8):4831–4845, 2021. doi:10.1021/acs.jctc.1c00445.
- [34] Zhendong Li and Wenjian Liu. Spin-adapted open-shell random phase approximation and time-dependent density functional theory. I. Theory *The Journal of Chemical Physics*, 133(6):064106, 2010. doi:10.1063/1.3463799.
- [35] Zhendong Li, Wenjian Liu, Yong Zhang, and Bingbing Suo. Spin-adapted open-shell time-dependent density functional theory. II. Theory and pilot application *The Journal of Chemical Physics*, 134(13):134101, 2011. doi:10.1063/1.3573374.
- [36] Zhendong Li and Wenjian Liu. Spin-adapted open-shell time-dependent density functional theory. III. An even better and simpler formulation *The Journal of Chemical Physics*, 135(19):194106, 2011. doi:10.1063/1.3660688.
- [37] Zhendong Li and Wenjian Liu. Critical Assessment of TD-DFT for Excited States of Open-Shell Systems: I. Doublet–Doublet Transitions *Journal of Chemical Theory and Computation*, 12(1):238–260, 2016. doi:10.1021/acs.jctc.5b01158.
- [38] Zhendong Li and Wenjian Liu. Critical Assessment of Time-Dependent Density Functional Theory for Excited States of Open-Shell Systems: II. Doublet-Quartet Transitions *Journal of Chemical Theory and Computation*, 12(6):2517–2527, 2016. doi:10.1021/acs.jctc.5b01219.
- [39] Zhendong Li and Wenjian Liu. Theoretical and numerical assessments of spin-flip time-dependent density functional theory *The Journal of Chemical Physics*, 136(2):024107, 2012. doi:10.1063/1.3676736.

- [40] Zhendong Li and Wenjian Liu. First-order nonadiabatic coupling matrix elements between excited states: A Lagrangian formulation at the CIS, RPA, TD-HF, and TD-DFT levels *The Journal of Chemical Physics*, 141(1):014110, 2014. doi:10.1063/1.4885817.
- [41] Zhendong Li, Bingbing Suo, and Wenjian Liu. First order nonadiabatic coupling matrix elements between excited states: Implementation and application at the TD-DFT and pp-TDA levels *The Journal of Chemical Physics*, 141(24):244105, 2014. doi:10.1063/1.4903986.
- [42] Zikuan Wang, Chenyu Wu, and Wenjian Liu. NAC-TDDFT: Time-Dependent Density Functional Theory for Nonadiabatic Couplings *Accounts of Chemical Research*, 54(17):3288–3297, 2021. doi:10.1021/acs.accounts.1c00312.
- [43] Zikuan Wang, Zhendong Li, Yong Zhang, and Wenjian Liu. Analytic energy gradients of spin-adapted open-shell time-dependent density functional theory *The Journal of Chemical Physics*, 153(16):164109, 2020. doi:10.1063/5.0025428.
- [44] Daoling Peng, Jianyi Ma, and Wenjian Liu. On the construction of Kramers paired double group symmetry functions *International Journal of Quantum Chemistry*, 109(10):2149–2167, 2009. doi:10.1002/qua.22078.
- [45] Wenjian Liu and Mark R. Hoffmann. SDS: the 'static–dynamic–static' framework for strongly correlated electrons *Theoretical Chemistry Accounts*, 133(5):1481, Apr 2014. doi:10.1007/s00214-014-1481-x.
- [46] Wenjian Liu and Mark R. Hoffmann. iCI: Iterative CI toward full CI *Journal of Chemical Theory and Computation*, 12(3):1169–1178, 2016. doi:10.1021/acs.jctc.5b01099.
- [47] Ning Zhang, Wenjian Liu, and Mark R. Hoffmann. Iterative Configuration Interaction with Selection *Journal of Chemical Theory and Computation*, 16(4):2296–2316, 2020. doi:10.1021/acs.jctc.9b01200.
- [48] Ning Zhang, Wenjian Liu, and Mark R. Hoffmann. Further Development of iCIPT2 for Strongly Correlated Electrons *Journal of Chemical Theory and Computation*, 17(2):949–964, 2021. doi:10.1021/acs.jctc.0c01187.
- [49] Yibo Lei, Bingbing Suo, and Wenjian Liu. iCAS: Imposed Automatic Selection and Localization of Complete Active Spaces *Journal of Chemical Theory and Computation*, 17(8):4846–4859, 2021. doi:10.1021/acs.jctc.1c00456.
- [50] Yang Guo, Ning Zhang, Yibo Lei, and Wenjian Liu. iCISCF: An Iterative Configuration Interaction-Based Multi-configurational Self-Consistent Field Theory for Large Active Spaces *Journal of Chemical Theory and Computation*, 17(12):7545–7561, 2021. doi:10.1021/acs.jctc.1c00781.
- [51] Chao Huang, Wenjian Liu, Yunlong Xiao, and Mark R. Hoffmann. iVI: An iterative vector interaction method for large eigenvalue problems *Journal of Computational Chemistry*, 38(29):2481–2499, 2017. doi:10.1002/jcc.24907.
- [52] Chao Huang and Wenjian Liu. iVI-TD-DFT: An iterative vector interaction method for exterior/interior roots of TD-DFT *Journal of Computational Chemistry*, 40(9):1023–1037, 2019. doi:10.1002/jcc.25569.
- [53] H. B. Schlegel and M. J. Frisch. Transformation between Cartesian and pure spherical harmonic Gaussians *Int. J. Quant. Chem.*, 54:83–87, 1995. doi:10.1002/qua.560540202.
- [54] Benjamin P. Pritchard, Doaa Altarawy, Brett Didier, Tara D. Gibsom, and Theresa L. Windus. A New Basis Set Exchange: An Open, Up-to-date Resource for the Molecular Sciences Community *J. Chem. Inf. Model.*, 59:4814–4820, 2019. doi:10.1021/acs.jcim.9b00725.

- [55] T. Q. Teodoro, A. B. F. da Silva, and R. L. A. Haiduke. Relativistic Prolapse-Free Gaussian Basis Set of Quadruple- $\zeta$  Quality: (aug-)RPF-4Z. I. The s- and p-Block Elements *J. Chem. Theory Comput.*, 10:3800–3806, 2014. doi:10.1021/ct500518n.
- [56] T. Q. Teodoro, A. B. F. da Silva, and R. L. A. Haiduke. Relativistic Prolapse-Free Gaussian Basis Set of Quadruple- $\zeta$  Quality: (aug-)RPF-4Z. II. The d-Block Elements *J. Chem. Theory Comput.*, 10:4761–4764, 2014. doi:10.1021/ct500804j.
- [57] T. Q. Teodoro, L. Visscher, A. B. F. da Silva, and R. L. A. Haiduke. Relativistic Prolapse-Free Gaussian Basis Sets of Quadruple- $\zeta$  Quality: (aug-)RPF-4Z. III. The f-Block Elements *J. Chem. Theory Comput.*, 13:1094–1101, 2017. doi:10.1021/acs.jctc.6b00650.
- [58] J.-P. Blaudeau, S. R. Brozell, S. Matsika, Z. Zhang, and R. M. Pitzer. Atomic orbital basis sets for use with effective core potentials *Int. J. Quant. Chem.*, 77:516–520, 2000.
- [59] R. B. Ross, S. Gayen, and W. C. Ermler. Ab initio relativistic effective potentials with spin-orbit operators. V. Ce through Lu *J. Chem. Phys.*, 100:8145–8155, 1994. doi:10.1063/1.466809.
- [60] W. C. Ermler, R. B. Ross, and P. A. Christiansen. Ab initio relativistic effective potentials with spin-orbit operators. VI. Fr through Pu *Int. J. Quant. Chem.*, 40:829–846, 1991. doi:10.1002/qua.560400611.
- [61] C. S. Nash, B. E. Bursten, and W. C. Ermler. Ab initio relativistic effective potentials with spin-orbit operators. VII. Am through element 118 *J. Chem. Phys.*, 106:5133–5142, 1997. doi:10.1063/1.473992.
- [62] C. S. Nash, B. E. Bursten, and W. C. Ermler. Erratum: “Ab initio relativistic effective potentials with spin-orbit operators. VII. Am through element 118” *J. Chem. Phys.*, 111:2347–2347, 1999. doi:10.1063/1.479506.
- [63] A. Weigand, X. Cao, T. Hangele, and M. Dolg. Relativistic Small-Core Pseudopotentials for Actinium, Thorium, and Protactinium *J. Phys. Chem. A*, 118:2519–2530, 2014. doi:10.1021/jp500215z.
- [64] M. Dolg and X. Cao. Accurate Relativistic Small-Core Pseudopotentials for Actinides. Energy Adjustment for Uranium and First Applications to Uranium Hydride *J. Phys. Chem. A*, 113:12573–12581, 2009. doi:10.1021/jp9044594.
- [65] D. J. Tozer and M. J. G. Peach. Molecular excited states from the SCAN functional *Mol. Phys.*, 116:1504–1511, 2018. doi:10.1080/00268976.2018.1453094.
- [66] J. C. Boettger. Approximate two-electron spin-orbit coupling term for density-functional-theory DFT calculations using the Douglas-Kroll-Hess transformation *Phys. Rev. B*, 62:7809–7815, 2000. doi:10.1103/PhysRevB.62.7809.
- [67] M. Filatov, W. Zou, and D. Cremer. Spin-orbit coupling calculations with the two-component Normalized Elimination of the Small Component Method *J. Chem. Phys.*, 139:014106, 2013. doi:10.1063/1.4811776.
- [68] S. Koseki, M. S. Gordon, M. W. Schmidt, and N. Matsunaga. Main Group Effective Nuclear Charges for Spin-Orbit Calculations *J. Phys. Chem.*, 99:12764–12772, 1995. doi:10.1021/j100034a013.
- [69] S. Koseki, M. W. Schmidt, and M. S. Gordon. Effective Nuclear Charges for the First- through Third-Row Transition Metal Elements in Spin-Orbit Calculations *J. Phys. Chem. A*, 102:10430–10435, 1998. doi:10.1021/jp983453n.
- [70] L. Visscher and K. G. Dyall. Dirac–Fock Atomic Electronic Structure Calculations Using Different Nuclear Charge Distributions *At. Data and Nucl. Data Tables*, 67:207–224, 1997. doi:10.1006/adnd.1997.0751.

- [71] D. Andrae. Finite nuclear charge density distributions in electronic structure calculations for atoms and molecules *Phys. Rep.*, 336:413–525, 2000. doi:10.1016/S0370-1573(00)00007-7.
- [72] D. Andrae. *Relativistic Electronic Structure Theory, Part 1: Fundamentals*, pages 203–258. Elsevier, Amsterdam, 2002.
- [73] H. Zhu, C. Gao, M. Filatov, and W. Zou. Mössbauer isomer shifts and effective contact densities obtained by the exact two-component (X2C) relativistic method and its local variants *Phys. Chem. Chem. Phys.*, 22:26776–26786, 2020. doi:10.1039/d0cp04549g.
- [74] M. Römel, S. Ye, and F. Neese. Calibration of Modern Density Functional Theory Methods for the Prediction of  $^{57}\text{Fe}$  Mössbauer Isomer Shifts: Meta-GGA and Double-Hybrid Functionals *Inorg. Chem.*, 48:784–785, 2009. doi:10.1021/ic801535v.
- [75] M. Pápai and G. Vankó. On Predicting Mössbauer Parameters of Iron-Containing Molecules with Density-Functional Theory *J. Chem. Theory Comput.*, 9:5004–5020, 2013. doi:10.1021/ct4007585.
- [76] L. C. Motta and J. Autschbach. Theoretical Prediction and Interpretation of  $^{237}\text{Np}$  Mössbauer Isomer Shifts *J. Chem. Theory Comput.*, 17:6166–6179, 2021. doi:10.1021/acs.jctc.1c00687.
- [77] S. F. McWilliams, E. Brennan-Wydra, K. C. MacLeod, and P. L. Holland. Density Functional Calculations for Prediction of  $^{57}\text{Fe}$  Mössbauer Isomer Shifts and Quadrupole Splittings in  $\beta$ -Diketiminato Complexes *ACS Omega*, 2:2594–2606, 2017. doi:10.1021/acsomega.7b00595.
- [78] Yong Zhang, Bingbing Suo, Zikuan Wang, Ning Zhang, Zhendong Li, Yibo Lei, Wenli Zou, Jun Gao, Daoling Peng, Zhichen Pu, Yunlong Xiao, Qiming Sun, Fan Wang, Yongtao Ma, Xiaopeng Wang, Yang Guo, and Wenjian Liu. BDF: A relativistic electronic structure program package *The Journal of Chemical Physics*, 152(6):064113, 2020. doi:10.1063/1.5143173.
- [79] Wenjian Liu, Fan Wang, and Lemin Li. The Beijing Density Functional (BDF) Program Package: Methodologies and Applications *Journal of Theoretical and Computational Chemistry*, 02(02):257–272, 2003. doi:10.1142/S0219633603000471.
- [80] Wenjian Liu, Fan Wang, and Lemin Li. *Relativistic Density Functional Theory: The BDF Program Package*, chapter 9, pages 257–282. Volume 5. World Scientific Publishing, 2004. doi:10.1142/9789812794901\_0009.