MOMAP

Tutorial 10

Turbomole Calculation with MOMAP

Version 2021B

December, 2021

MOMAP Tutorial 10

Version 2021B edited by:

Dr. Qian Peng

Ms. Shiyun Lin

Dr. Qikai Li

Released by Hongzhiwei Technology (Shanghai) Co., Ltd

and Z.G. Shuai Group

The information in this document applies to version 2021B of MOMAP

Table of Contents

Turbomole Calculation with MOMAP

All TURBOMOLE modules need the control file as input file, as the control file provides the information necessary for all kinds of runs and tasks. The turbomole tool define can be used to generate step by step the control file, the control file contains the coordinates, atomic attributes (e.g. basis sets), MO start vectors, and keywords specific for the desired method of calculation *etc*.

A typical way to perform a TURBOMOLE (the version used here is 7.5) calculation from scratch is as follows:

1. Create a root directory and generate the coord file

Let us take azulene as an example. First create a root directory and enter into it,

```
$ mkdir tmole75-azulene
$ cd tmole75-azulene
```

Generate the atomic coordinates by any tool or program that you are familiar with, and save as the .xyz format which is a standard output format of all programs, e.g., coord.xyz.

Use the TURBOMOLE script x2t to convert the .xyz file to the TURBOMOLE coord file:

```
$ x2t xyzinputfile > coord
```

Next, we carry out the specific calculations in separate sub-directories.

2. Ground state geometry optimization

2.1 Create a sub-directory gsopt for ground state geometry optimization, enter into that directory, and copy the upper level coord to the current work directory:

```
$ mkdir gsopt
$ cd gsopt
$ cp ../coord  ./
```

2.2 Use the command define to generate the input files

```
$ define                        # run command define
<Enter>
ground state opt                # specifying the title

# setup the molecular structure
sy                              # add molecular symmetry
<Enter>
a coord                         # add atomic coordinates
ired                            # use the generalized internal coordinates
*                               # ends the current menu
```

```
# setup the atomic basis sets
b all def-SVP                    # set the basis sets for all atoms to def-SVP
*                                # ends the current menu

# set up an initial guess for MOs and occupation numbers
eht                              # Extended Hueckel guess
<Enter>
<Enter>
<Enter>

# method selection
dft                              # enter into the DFT menu
on                               # toggle it to on
func                             # enter into the functional menu
b3-lyp                           # use b3lyp functional
*                                # ends current menu

# setup RI approximation
ri                               # enter into RI menu
on                               # toggle the RI to on
m 4000                           # set the memory value in MB for RI
jbas                             # enter into the aug- basis sets menu
b all def-SVP                    # set the basis sets for all atoms to def-SVP
*                                # ends the auxiliary RI-J basis sets menu
*                                # ends the RI menu

*                                # ends the define running
```

If successful, the following files should be created in the current work directory:

```
auxbasis  basis  control  coord  mos
```

Note: The greyed text part is tried to be problematic, thus use with care!

In addition, we would like to also modify the convergence limits in the `control` file, for example:

```
$gridsize               m4
$scfconv                8
$scfiterlimit           200
```

Once done, then we can perform the ground state geometry optimization by using the following command in your job script:

```
jobex -np $SLURM_NPROCS -c 200 -energy 8 -gcart 4 -gexp 4 > gs-opt.out
```

Here -gcart 4 denotes the threshold of the cartesian gradient, which we set to 10^{-4} atomic units (default: 3), similarly, -energy 8 converge total energy up to 10^{-8} Hartree (default: 6), -gexp 4 converge expnt. gradient norm up to 10^{-4} atomic units, and -c 200 sets the maximum number of iterations in the SCF to 200 (default: 30) which can also be changed in the `control` file as show above.

3. Ground state frequency calculation

Create a sub-directory `gsvib`, then copy all the files in `gsopt` to `gsvib`.

```
$ mkdir gsvib
$ cd gsvib
$ cp ../gsopt/* ./
```

Modify the `control` file, add the following contents (in red color) to it,

```
$last step dscf
$hessian  file=hessian
$dipgrad  file=dipgrad
$vibrational spectrum  file=spectrum
$vibrational normal modes  file=modes
$last SCF energy change = 0.40612349E-08
```

Once done, we can perform the ground state frequency calculation by using the following command in your job script:

```
aoforce -np $SLURM_NPROCS -c 200 -energy 8 -gcart 4 -gexp 4 > gs-freq.out
```

The ground state vibration frequency information should be included in the output file `gs-freq.out`.

4. Excited state geometry optimization

Create a sub-directory `esopt`, the copy all the files in `gsopt` to `esopt`.

```
$ mkdir esopt
$ cd esopt
$ cp ../gsopt/* ./
```

Modify the ground state geometry optimized `control` file, add the following contents (in red color) to it,

```
$last step     dscf
$scfinstab rpas               # specify rpa as the excited state calc. method
$soes
a   3                         # calculate 3 excited states
$exopt 1                      # the state to be optimized is S1
$last SCF energy change = 0.40612349E-08
```

In addition, we would like to also modify the convergence limits in the `control` file:

```
$scfconv               9
```

Once done, we can perform the excited state geometry optimization by using the following command in your job script:

```
jobex -ex -np $SLURM_NPROCS -c 200 -energy 8 -gcart 4 -gexp 4 > es-opt.out
```

5. Excited state frequency calculation

Create a sub-directory `esvib`, the copy all the files in `esopt` to `esvib`.

```
$ mkdir esvib
$ cd esvib
$ cp ../esopt/* ./
```

Modify the `control` file, add the following contents (in red color) to it,

```
$last step     egrad
$hessian  file=hessian
$dipgrad  file=dipgrad
$vibrational spectrum  file=spectrum
$vibrational normal modes  file=modes
$last SCF energy change = 0.84278895E-08
```

Once done, we can perform the excited state frequency calculation by using the following command in your job script:

```
NumForce -ex 1 -mfile hosts > es-freq.out
```

The excited state vibrational frequency information should be included in the output file `numforce/aoforce.out`.

6. Nonadiabatic coupling calculation

The Turbomole `$nacme` flag is used to compute the cartesian non-adiabatic coupling vectors between the excited state of interest and the ground state. This option requires the use of `weight derivatives` in section `dft`. It is only implemented for the C_1 symmetry. The Hartree-Fock and DFT ground state calculations for all available DFT functionals, without the usage of `RI-J` approximation, SMP and MPI, can be done using `dscf`. The single point excited state energies for CIS, TDHF, and TDDFT methods can be calculated using `escf`, while the excited state energies, gradients, and other first order properties are provided by `egrad`. Both modules require the well converged ground state orbitals. Please refer to the Turbomole Manual for details.

Create a sub-directory `nacme`, the copy all the files in `esopt` to `nacme`.

```
$ mkdir nacme
$ cd nacme
$ cp ../esopt/* ./
```

Modify the excited state geometry optimized `control` file, add the following contents (in red color) to it,

```
$dft
     functional b3-lyp
     gridsize m3
     weight derivatives
$nacme full                          # do Nonadiabatic coupling calculation
```

Once done, we can continue to perform the nonadiabatic coupling calculation by using the following commands in your job script:

```
dscf > dscf.out
escf > escf.out
egrad > egrad.out
```

The nonadiabatic coupling information should be included in the `control` file as shown below:

```
…
$dipole from egrad
  x    -0.00000006453587    y    -0.00000014356762    z     0.10346295396548
a.u.
   | dipole | =    0.2629785855  debye
$optinfo       file=optinfo
$hessapprox   file=hessapprox
$cartesianstep
  total steps      5
  forceupdate      on
$tmole
$last excitation energy change= 0.38919659E-07
$redund_inp
    metric -3
$closed shells
 a       1-34                                   ( 2 )
$orbital_max_rnorm 0.28627509967336E-08
$gsenergy=    -385.29894
$esenergy=    -385.23999
$couplingvector
# cartesian nonadiabatic coupling vector
  0.71587481531775D+00  0.15632285732914D-05  0.18909143280755D-06
  -.37171830858308D+00  -.13145987445777D-05  0.34415755183263D+00
  0.12166702586431D+00  0.95824917549734D-06  -.34980278306214D+00
  -.37171813646616D+00  -.35401184453269D-06  -.34415771541683D+00
…
```

Now the Turbomole related calculations are done, we can continue to do the evc *etc*. calculations.

7. Perform evc calculation

Create a sub-directory `evc`, copy all the needed files from Turbomole calculations to the directory:

```
$ mkdir evc
$ cd evc
$ cp ../gsvib/gs-freq.out ./
$ cp ../esvib/numforce/aoforce.out ./
$ cp ../nacme/control ./nacme-control
```

Create a MOMAP control file `momap.inp` with the following contents:

```
do_evc          = 1

&evc
  ffreq(1)      = "gs-freq.out"
  ffreq(2)      = "aoforce.out"
  proj_nacme    = 1
```

```
  fnacme         = "nacme-control"
  sort_mode      = 1
/
```

Finally, we can run the `momap.py` to perform the evc calculation as usual:

```
$ momap.py
```

Here we assume that you have set up the MOMAP environment properly.

The generated files should look like the following:

```
aoforce.out   evc.cart.nac  evc.dx.v.xyz  evc.out       evc.vib2.xyz  momap.inp
evc.cart.abs  evc.dint.abs  evc.dx.x.com  evc.vib1.xyz  evc.vib20.xyz nacme-control
evc.cart.dat  evc.dint.dat  evc.dx.x.xyz  evc.vib10.xyz gs-freq.out   nodefile
```

The users should check the contents carefully before performing the following tasks.

8. Job scripts

The scheduling system used in this tutorial is SLURM, a typical Slurm job script is shown as follows:

```
#!/bin/bash

#SBATCH --time=0-2:00:00
#SBATCH --output=stdout.txt
#SBATCH --job-name=turbomole
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=32
#SBATCH --partition=short

# load turbomole environments, can be [ smp | mpi | serial ]
module load turbomole/smp

#---- Very important!
scratch=1

username=`whoami`

# In this way, you can run multiple jobs on the same node
rundir=$SLURM_JOB_ID

if [ $scratch -eq "1" ]; then

  SCRATCH_DIR=/scratch/${username}
  if [ ! -a $SCRATCH_DIR ]; then
    echo "Scratch directory $SCRATCH_DIR created."
    mkdir -p $SCRATCH_DIR
  fi

  SCRATCH_DIR=$SCRATCH_DIR/${rundir}
  if [ ! -a $SCRATCH_DIR ]; then
    echo "Scratch directory $SCRATCH_DIR created."
    mkdir -p $SCRATCH_DIR
  fi

  echo "Work directory $SLURM_SUBMIT_DIR created."
```

```
  cd $SLURM_SUBMIT_DIR
  cp * $SCRATCH_DIR

  echo Working directory is $SCRATCH_DIR
  cd $SCRATCH_DIR

else

  echo Working directory is $SLURM_SUBMIT_DIR
  cd $SLURM_SUBMIT_DIR

fi

srun hostname -s | sort -n > hosts
echo Running on host `hostname`
echo Starting Time is `date`
echo Directory is `pwd`
echo This jobs runs on the following processors:
echo `cat hosts`
#NPROCS=`wc -l < hosts`
echo This job has allocated $SLURM_NPROCS cores

if [ $SLURM_NPROCS -gt 1 ]; then
   export PARNODES=$SLURM_NPROCS
fi

# Note: Uncomment the line(s) that suits your case!

# ---- GS opt ----
#jobex -np $SLURM_NPROCS -c 200 -energy 8 -gcart 4 -gexp 4 > gs-opt.out

# ---- GS freq ----
#aoforce -np $SLURM_NPROCS -c 200 -energy 8 -gcart 4 -gexp 4 > gs-freq.out

# ---- ES opt ----
#jobex -ex -np $SLURM_NPROCS -c 200 -energy 8 -gcart 4 -gexp 4 > es-opt.out

# ---- ES freq ----
#NumForce -ex 1 -mfile hosts > es-freq.out

# ---- NACME ----
#dscf > dscf.out
#escf > escf.out
#egrad > egrad.out

if [ $scratch -eq "1" ]; then
  rm -rf $SCRATCH_DIR/twoint* &> /dev/null
  cp -rf $SCRATCH_DIR/* $SLURM_SUBMIT_DIR/
  rm -rf $SCRATCH_DIR
fi
```