

# Users' guide to MakePES

Kiyoshi Yagi  
kiyoshi.yagi@riken.jp

Theoretical Molecular Science Laboratory  
RIKEN Cluster for Pioneering Research

2019/11/12

- In this guide, I will illustrate how to use MakePES to generate QFF, GridPES, and Multiresolution PES. Formaldehyde ( $\text{H}_2\text{CO}$ ) is used as an example, but the method is applicable to any molecule.
- MakePES is a command line based program. This guide assumes that you are familiar with basic commands in UNIX. Shell scripts are given for Bourne Shell (bash).
- This guide also assumes that you have installed the program. Change “/path/to” to your installation directory when you see a command like this,

```
sindo_jar=/path/to/sindo-4.0/jar  
java -cp '$sindo_jar/*' RunMakePES
```

- In this guide, we will use Gaussian for the electronic structure calculations. Make sure that you have configured “runGaussian.sh” and the path to it.

```
> export PATH=${PATH}:/path/to/sindo-4.0/script  
> runGaussian.sh  
USAGE: runGaussian.sh arg[1] arg[2]  
arg[1] : current directory  
arg[2] : input file
```

- If you don't use Gaussian, you may still use MakePES in generic mode; see 1.4 and 2.4.

# Contents

- 0. Harmonic analysis and generation of a minfo file  
(0.harmonic\_h2co)
- 1. Quartic force field (1.qff\_h2co)
  - 1.1. Using single node (1-1.single)
  - 1.2. Using parallel computers (1-2.parallel)
  - 1.3. Generate input files and exit (1-3.dryrun)
  - 1.4. Generic mode (1-4.generic)
- 2. Grid potential (2.grid\_h2co)
  - 2.1. 1MR-grid PES (2-1.1MR)
  - 2.2. 2MR-grid PES (2-2.2MR)
  - 2.3. 3MR-grid PES (2-3.3MR)
  - 2.4. Generic mode (2-4.1MR\_generic)
- 3. Multiresolution PES (3.mrpes\_h2co)
- 4. TIP and FAQ
- 5. References
- Appendix: List of all keys

Sample files are found in `sindo-4.0/doc/MakePES/sample_MakePES`

## 0. Harmonic analysis and generation of a minfo file

---

Proceed to 0.harmonic\_h2co and find an input file to perform harmonic vibrational analysis for formaldehyde using Gaussian.

```
> cd 0.harmonic_h2co
> ls
h2co-b3lyp-dz.inp    log/    run.sh
```

- “log” folder contains sample output files.

h2co-b3lyp-dz.inp is the input file. Run Gaussian by the following command:

```
> runGaussian.sh ./ h2co-b3lyp-dz.inp
```

- review the installation if you cannot run Gaussian with this command.
- the two arguments are (working folder) and (input file), respectively.

You will obtain the following output files when the job ends.

```
> ls h2co-b3lyp-dz.*
h2co-b3lyp-dz.chk    h2co-b3lyp-dz.fchk    h2co-b3lyp-dz.inp
h2co-b3lyp-dz.out    run.sh
```

.fchk is a formatted check point file, which archives the result of quantum chemistry calculations. Let us convert the fchk file to a minfo file,

```
> java -cp "/path/to/sindo-4.0/jar/*" Fchk2Minfo h2co-b3lyp-dz
```

- The argument after Fchk2Minfo is the basename of output files.

You will find a minfo file,

```
> ls h2co-b3lyp-dz.*  
h2co-b3lyp-dz.chk      h2co-b3lyp-dz.fchk  h2co-b3lyp-dz.inp  
h2co-b3lyp-dz.out     h2co-b3lyp-dz.minfo run.sh
```

Minfo file includes the equilibrium geometry, harmonic frequencies, and vibrational displacement vectors.

The same result can be obtained by JSindo; refer to the Users' guide to JSindo. You can find details about the format of minfo file therein, too.

“run.sh” is a script to do this work at once. Change “/path/to” to your installation directory using an editor,

```
> vi run.sh
#!/bin/bash

sindo_dir=/path/to/sindo-4.0
sindo_jar=$sindo_dir/jar
PATH=$PATH:$sindo_dir/script

runGaussian.sh ./ h2co-b3lyp-dz.inp
java -cp "$sindo_jar/*" Fchk2Minfo h2co-b3lyp-dz
```

Then, run the script.

```
> ./run.sh
> ls
h2co-b3lyp-dz.chk      h2co-b3lyp-dz.fchk    h2co-b3lyp-dz.inp
h2co-b3lyp-dz.minfo   h2co-b3lyp-dz.out     run.sh
```

# 1.qff\_h2co

---

## 1-1.single

In this section, we generate a quartic force field (QFF) for formaldehyde [1].  
Proceed to 1.qff\_h2co/1-1.single to find input files,

```
> cd 1.qff_h2co/1-1.single
> ls
GaussianTemplate  log/      makePES.xml  resources.info  run.sh
```

makePES.xml is the main file, which is structured using tags in xml format. It is divided into sections by, <makePES> ... </makePES>, <qchem> ... </qchem>, and <qff> ... </qff>. The options are specified in each section by <key value="value" />. The value is case insensitive except for filenames. Comment out is possible as usual by <!-- ... -->.

### makePES

```
<makePES>
  <minfoFile value="../../0.harmonic_h2co/h2co-b3lyp-dz.minfo" />
  <MR      value="3" />
  read a minfo file of H2CO
  the order of the mode coupling expansion.
```

## makePES

```
<qchem>
  <program value="gaussian" />   use Gaussian
  <dryrun value="false"/>        run Gaussian
  <removefiles value="true" />    remove the output of Gaussian
  <title value="B3LYP/cc-pVDZ" /> set the title
  <template value="GaussianTemplate" /> the name of template file to
                                     generate Gaussian input
</qchem>
<qff>
  <stepsize value="0.5" />       step size of numerical differentiations
  <ndifftype value="hess"/>      numerical differentiations using Hessian
  <mopfile value="prop_no_1.mop" /> name of a mop file
</qff>
</makePES>
```

resources.info provides hostname of nodes to run Gaussian. When we run on a single node, it is not important (but still, it should exist). We will later discuss this file in detail for parallel calculations in 1.2-parallel.



GaussianTemplate is a template file to generate input files for Gaussian specified by <template> in <qchem> section. It is the same as the usual input file for Gaussian, except for the red colored text.

### GaussianTemplate

```
%chk=#basename#.chk
%NprocShared=2
%mem=1GB
#P B3LYP/CC-PVDZ FREQ NOSYMMETRY MAXDISK=240GB

Frequency at B3LYP/cc-pVDZ

0 1
#coordinate#
```

MakePES replaces #basename# and #coordinate# by the filename and the coordinates, respectively, to create input files.

"FREQ" keyword is necessary because we use numerical differentiations of the Hessian matrix (i.e., ndiffdtype = hess).

MakePES is invoked by the following command

```
> sindo_jar=/path/to/sindo-4.0/jar  
> java -cp "$sindo_jar/*" RunMakePES >& makePES.out
```

The main input file is set to makePES.xml by default. You can specify a different input file with `-f` option,

```
> sindo_jar=/path/to/sindo-4.0/jar  
> java -cp $sindo_jar/*" RunMakePES -f makePES.xml >& makePES.out
```

In the output file, the options are first printed, and then the status of electronic structure calculations is printed,

#### makePES.out

Execute electronic structure calculations.

Thread0> Running minfo.files/mkqff-eq.inp on kyagi-mac3.local at ...

Thread0> Running minfo.files/mkqff0-0.inp on kyagi-mac3.local at ...

Thread0> Running minfo.files/mkqff0-1.inp on kyagi-mac3.local at ...

During this step, Gaussian jobs are carried out in a folder minfo.files. Because we've set `<removefiles>` to true, the input and output files are removed leaving only minfo files in the folder.

When this step is done, you will see an output like this,

### makePES.out

```
End of electronic structure calculations.
Storing electronic structure data in tempfile ... Done!
Generating prop_no_1.mop... Done!
Removing the tempfiles ... Done!
End of QFF generation.
```

The QFF coefficients are written to <mopfile>, i.e., prop\_no\_1.mop.

```
1.1832573615027308000000e-15 1
2.7040635655127780000000e-03 1 1
1.1590249883181242000000e-17 1 1 1
2.2875576159959352000000e-05 1 1 1 1
-2.1030452203654645000000e-16 2
```

one-body terms:  
ci, cii, ciii, ciii

```
-4.189735728085525000000e-18 1 2
-6.3982329963326570000000e-14 1 2 2
1.8221212840606143000000e-13 1 2 2 2
-1.9301098360709593000000e-15 1 1 2
2.9510063534107640000000e-05 1 1 2 2
3.2272058841394823000000e-15 1 1 1 2
9.6870281798527820000000e-19 1 3
3.1255300969821536000000e-13 1 3 3
```

two-body terms:  
cij, cijj, cijjj, ciiij, ciiijj, ciiij

```
1.443395067341624000000e-13 1 2 3
-1.6315521638821910000000e-13 1 2 3 3
-3.0819651776043526000000e-13 1 2 2 3
6.1712325565851670000000e-14 1 1 2 3
1.8082830483224921000000e-13 1 2 4
```

three-body terms:  
cijk, cijkk, cijjk, ciijk

Note that four-body terms (cijkl) are missing because <MR> was set to 3.

## 1-2.parallel

The electronic structure calculations are an intensive bottleneck for generating the PES. In the case of  $\text{H}_2\text{CO}$ , FREQ calculations at 13 grid points are required. They were carried out one by one in the previous section using one node.

MakePES can distribute the grid points to multiple nodes, and process the FREQ calculations in parallel. This function substantially speeds up the calculation. It requires that the nodes have shared disks (via NFS), where the input files as well as `sindo/gaussian` are located, and are inter-connectable with SSH without being asked for a password.

Proceed to `1-2.parallel` to find the same set of input files.

```
> cd 1.qff_h2co/1-2.parallel
> ls
GaussianTemplate  log/      makePES.xml  resources.info  run.sh
```

Assuming that we use 16 core x 2 nodes, we make the following modification to `resources.info` and `GaussianTemplate`

### resources.info

```
beluga01  
beluga01  
beluga02  
beluga02
```

} The hostname of 2 nodes.  
We will run 2 processes in each nodes.

### GaussianTemplate

```
%chk=#basename#.chk  
%NprocShared=8  
%mem=1GB  
...
```

Each Gaussian process uses 8 cores

makePES.xml is the same as before. Set an environment variable SINDO\_RSH=ssh, and then invoke MakePES as before,

```
> export SINDO_RSH=ssh  
> sindo_jar=/path/to/sindo-4.0/jar  
> java -cp "$sindo_jar/*" RunMakePES >& makePES.out
```

You can see in the output that the grid points are distributed to beluga01 and 02, each in 2 processes.

## makePES.out

Execute electronic structure calculations.

Thread2@beluga02> Running minfo.files/mkqff0-1.inp on beluga02 at ...

Thread3@beluga02> Running minfo.files/mkqff1-0.inp on beluga02 at ...

Thread0@beluga01> Running minfo.files/mkqff-eq.inp on beluga01 at ...

Thread1@beluga01> Running minfo.files/mkqff0-0.inp on beluga01 at ...

When the job is done, you will obtain the same mop file as before, but in a much faster computational time.

## 1-3.dryrun

The <dryrun> option generates input files for grid points, and then stops the program without executing Gaussian. Proceed to 1-3.dryrun,

```
> cd 1.qff_h2co/1-3.dryrun
> ls
GaussianTemplate    log1_dryrun_true/    log2_dryrun_false/
makePES.xml         resources.info       run.sh
```

The only difference is the value of <dryrun> in makePES.xml

```
makePES.xml

<qchem>
  <program value="gaussian" />
  <dryrun  value="true"/>  stop after generating the input files
```

Running the program creates input files for Gaussian in “minfo.files”,

```
> sindo_jar=/path/to/sindo-4.0/jar
> java -cp "$sindo_jar/*" RunMakePES >& makePES.out
> ls minfo.files/
mkqff-eq.inp  mkqff0-1.inp  mkqff1-1.inp  mkqff2-1.inp
mkqff3-1.inp  mkqff4-1.inp  ..
```

You may transfer these input files to other computer systems and carry out Gaussian there. Then, convert the formatted checkpoint files to minfo format using Fchk2Minfo.

```
> java -cp "/path/to/sindo-4.0/jar/*" Fchk2Minfo mkqffx-x
```

Bring back the minfo files and locate them in the minfo.files folder,

```
> ls minfo.files/  
mkqff-eq.inp      mkqff-eq.minfo  mkqff0-0.inp    mkqff0-0.minfo  
mkqff0-1.inp      mkqff0-1.minfo  ...
```

Change dryrun to false and run the program again.

```
makePES.xml  
  
<qchem>  
  <program value="gaussian" />  
  <dryrun value="false"/>  don't stop after generating the input files
```

```
> sindo_jar=/path/to/sindo-4.0/jar  
> java -cp "$sindo_jar/*" RunMakePES >& makePES.out  
> ls  
GaussianTemplate  makePES.out      makePES.xml      minfo.files/  
prop_no_1.mop     resources.info
```



You will see that the program immediately produces the mopfile.

The two log folders contains the files for the first step (log1\_dryrun\_true), and the files for the second step (log2\_dryrun\_false).

Note that, in general, MakePES looks into minfo.files folder for minfo files before starting Gaussian jobs. The job is skipped if a minfo file is found, and starts from the grid point where it ended before. In this example, we provided all minfo files, and thus the electronic structure calculations were all skipped. See the TIPS in 4.2 for more detail.

## 1-4.generic

Setting <program> to generic prints the coordinates to a file in xyz format. You have to generate the input files, carry out the electronic structure calculations, and return the information in minfo format by yourself. Nevertheless, this may be useful for users who wish to use programs other than Gaussian.

Proceed to 1-4.generic to find makePES.xml.

```
> cd 1.qff_h2co/1-4.generic  
> ls  
log1_genxyz/ log2_genmop/ makePES.xml
```

The file is different only in <qchem> section,

### makePES

```
<qchem>  
  <program value="generic" />  
  <title   value="B3LYP/cc-pVDZ" />  
  <xyzfile value="makeQFF" />  
</qchem>
```

"generic" means no specific program

set the name of xyz file

Running the program creates makeQFF.xyz,

```
> sindo_jar=/path/to/sindo-4.0/jar
> java -cp "$sindo_jar/*" RunMakePES >& makePES.out
> ls
makePES.out  makePES.xml  makeQFF.xyz
```

makeQFF.xyz is written in the usual xyz format,

```
                                makeQFF
4                                The number of atoms
mkqff-eq                        name of the first point
C      0.0000000000      0.0000000000      -0.6014736819
O      0.0000000000      -0.0000000000      0.6027247362
H      0.0000000000      0.9459644267      -1.2020174143
H     -0.0000000000     -0.9459644267      -1.2020174143
4
mkqff0-0                        name of the second point
C     -0.0125897498     -0.0000000000      -0.6014736819
O      0.0031430124      0.0000000000      0.6027247362
```

The name, colored in red, is the ID of each grid points. We assume you carry out the electronic structure calculations by yourself, and collect the data in a minfo file with a name, ID.minfo.

Run the electronic structure calculations and generate minfo files by yourself.  
Locate the minfo files in a minfo.files folder,

```
> ls minfo.files/  
mkqff-eq.minfo  mkqff0-1.minfo  mkqff1-1.minfo  mkqff2-1.minfo  
mkqff3-1.minfo  mkqff4-1.minfo  mkqff5-1.minfo  mkqff0-0.minfo  
mkqff1-0.minfo  mkqff2-0.minfo  mkqff3-0.minfo  mkqff4-0.minfo  
mkqff5-0.minfo
```

Run the program again (no need to change anything in makePES.xml).

```
> sindo_jar=/path/to/sindo-4.0/jar  
> java -cp "$sindo_jar/*" RunMakePES >& makePES.out  
> ls  
makePES.out  makePES.xml  makeQFF.xyz  minfo.files/  
prop_no_1.mop
```

The program produces the mopfile.

The two log folders contain the files for the first step (log1\_genxyz), and the files for the second step (log2\_genmop).

## 2. Grid potential

### 2.1. 1MR-grid PES

In this section, we generate a grid potential [2]. Proceed to 2.grid\_h2co/2-1.1MR to find input files,

```
> cd 2.grid_h2co/2-1.1MR
> ls
GaussianTemplate  log/      makePES.xml  resources.info  run.sh
```

makePES.xml has the same <qchem> section as before. A new section <grid> replaces <qff>. We also set <dipole> to true to generate dipole moment surfaces.

```
makePES

<makePES>
  <minfoFile value="../0.harmonic_h2co/h2co-b3lyp-dz.minfo" />
  <MR      value="1" />      MR=1 for 1MR-PES
  <dipole  value="true" />   calculate dipole moment surface
  ...
  <grid>
    <ngrid value="11" />     number of grid points for each coordinates
    <fullmc value="true"/>   calculate all modes
  </grid>
</makePES>
```

The generation of a gridPES requires only the energy at grid points, so that FREQ is no longer needed in GaussianTemplate. **Don't forget to remove FREQ** if you copy the file from QFF. Note that MakePES still works as long as the energy is printed in the output; however, it would be an enormous waste of time! SCF=TIGHT is recommended for HF/DFT calculations.

### GaussianTemplate

```
%chk=#basename#.chk
%NprocShared=8   Each Gaussian process uses 8 cores
%mem=1GB
#P B3LYP/CC-PVDZ SCF=TIGHT NOSYMMETRY MAXDISK=240GB
...              Don't put a FREQ keyword!
```

Modify resource.info and %NprocShared for your system. In this sample, we run 8 processes of Gaussian with 8 cores (64 cores in total).

### resources.info

```
beluga01 }
beluga01 } 4 nodes x 2 = 8 processes
...      }
beluga04 }
beluga04 }
```

Set an environment variable SINDO\_RSH=ssh, and then run MakePES,

```
> export SINDO_RSH=ssh
> sindo_jar=/path/to/sindo-4.0/jar
> java -cp "$sindo_jar/*" RunMakePES >& makePES.out
```

You will find in the output,

#### makePES.out

Setup MakeGrid module

o ngrid = 11

o 1MR Grid:

1 2 3 4 5 6

} 6 modes x 11 grid = 66 grid points

Enter GridPES generation:

Execute electronic structure calculations.

Thread2@beluga02.local> Running minfo.files/mkg-q1-11-1.inp on beluga02 at ...

Thread3@beluga02.local> Running minfo.files/mkg-q1-11-2.inp on beluga02 at ...

Thread0@beluga03.local> Running minfo.files/mkg-eq.inp on beluga03 at ...

After the energy calculations are done, pot/dipole files are created.

## makePES.out

Generating pot files.

```
o q1.pot [OK]
o q1.dipole [OK]
...
o q6.dipole [OK]
```

} **pot and dipole files are created.**

End of GridPES generation:

```
> ls q*
q1.dipole  q2.dipole  q3.dipole  q4.dipole  q5.dipole  q6.dipole
q1.pot     q2.pot     q3.pot     q4.pot     q5.pot     q6.pot
```

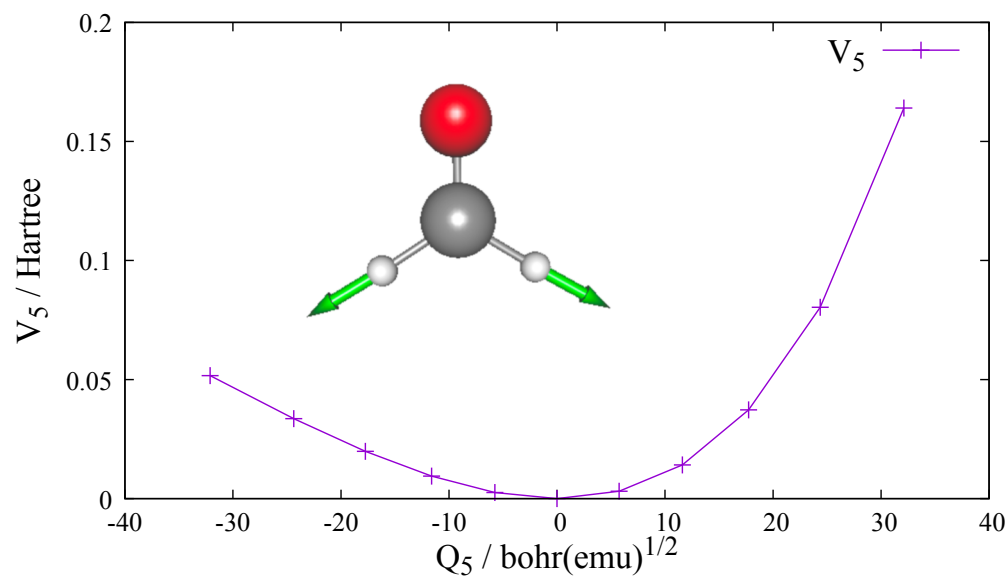
$qN$ .pot and  $qN$ .dipole contain the change of the potential energy and dipole moment, respectively, with respect to the equilibrium geometry along mode  $N$ . The values at the equilibrium geometry are written in eq.pot and eq.dipole.



grid points along  $Q_5$

```
B3LYP/cc-pVDZ
# Number of grids and data
11 1
# q5 Energy V5
-32.11897636 5.1637111962e-02
-24.36885358 3.3576631399e-02
-17.73801106 1.9823126834e-02
-11.61455479 9.4296641468e-03
-5.75063885 2.5713497506e-03
-0.00000000 0.0000000000e+00
5.75063885 3.0938756174e-03
11.61455479 1.4162827543e-02
17.73801106 3.7256435616e-02
24.36885358 8.0356295312e-02
32.11897636 1.6406728197e-01
q5.pot (END)
```

```
B3LYP/cc-pVDZ
# Number of grids and data
11 3
# q5 X dx Y dy Z dz
-32.11897636 6.5354419100e-16 -2.1581328800e-14 2.2264673800e-01
-24.36885358 1.5282778610e-15 -3.2503038080e-13 1.6380302600e-01
-17.73801106 -3.5132508090e-15 2.9134816400e-15 1.1545683200e-01
-11.61455479 6.1520087100e-16 -2.6271289280e-13 7.2992832000e-02
-5.75063885 6.2180322100e-16 9.1610204700e-14 3.4762361000e-02
-0.00000000 0.0000000000e+00 0.0000000000e+00 0.0000000000e+00
5.75063885 -5.2036891900e-16 -7.6053530000e-15 -3.1664072000e-02
11.61455479 -4.5814996000e-17 -2.2601766400e-14 -6.0408552000e-02
17.73801106 2.2404410010e-15 1.0871512036e-14 -8.6260919000e-02
24.36885358 3.7191305100e-16 1.0853080599e-14 -1.0913078900e-01
32.11897636 -1.4172511000e-16 1.1229081913e-14 -1.2867327600e-01
q5.dipole (END)
```



Plots of  $V_5$  at grid points of  $Q_5$ .  $Q_5$  (symmetric CH stretching mode) is visualized in the inset. Note that the arrow corresponds to the negative direction of  $Q_5$ , that is, the potential becomes flat as the CH bond extends.

## 2.2. 2MR-grid PES

In this section, we generate 2MR-grid PES for (Q1, Q2) and (Q5, Q6). Proceed to 2.grid\_h2co/2-2.2MR,

```
> cd 2.grid_h2co/2-2.2MR
> ls
GaussianTemplate    makePES.xml    ...
```

### makePES

```
<makePES>
  <minfoFile value="../0.harmonic_h2co/h2co-b3lyp-dz.minfo" />
  <MR value="2" />      MR=2 for 2MR-PES
  ...
  <grid>
    <ngrid value="9" />      number of grid points is reduced to 9
    <mc2 value="1,2, 5,6"/>  (Q1, Q2) and (Q5, Q6)
  </grid>
</makePES>
```

The two-mode terms are specified by <mc2> to (Q1,Q2) and (Q5,Q6). See the appendix on the details of format of mc2.

Calculating two 2MR-terms with  $n_{\text{grid}} = 9$  requires  $9 \times 9 \times 2 = 162$  grid points. However, we have already calculated the grid points along the axis. In order to re-use them, the pot and dipole files obtained in Sec. 2-1 are placed in the same directory:

```
> ls *pot *dipole
eq.dipole  q1.dipole  q2.dipole  q5.dipole  q6.dipole
eq.pot     q1.pot     q2.pot     q5.pot     q6.pot
```

MakePES make use of the previous calculation whenever possible. In this example, the information along the Q1, Q2, Q5, Q6 axis is provided by the files. Therefore, the number of grid points is reduced from 162 to  $8 \times 8 \times 2 = 128$ .

GaussianTemplate and resources.info are the same as before. Run MakePES,

```
> export SINDO_RSH=ssh
> sindo_jar=/path/to/sindo-4.0/jar
> java -cp "$sindo_jar/*" RunMakePES >& makePES.out
```

You may check the settings in the output,

### makePES.out

Setup MakeGrid module

o ngrid = 9            number of grid points

o 1MR Grid:

1 2 5 6            1-mode terms

o 2MR Grid:

(1,2) (5,6)        2-mode terms

Enter GridPES generation:

and that pot/dipole files are created at the end of the calculation.

### makePES.out

Generating pot files.

o q2q1.pot [OK]

o q2q1.dipole [OK]

o q6q5.pot [OK]

o q6q5.dipole [OK]

} 2-mode terms (9x9 grid points) are created.

End of GridPES generation:

In addition to the original files with 11 grid points (q[1-6].pot/dipole), new files for 2MR-PES with 9 grid points (q2q1.pot/dipole and q6q5.pot/dipole) are created.

```
> ls q*
q1.dipole      q2.pot         q5.dipole      q6.pot
q1.pot         q2q1.dipole    q5.pot         q6q5.dipole
q2.dipole      q2q1.pot       q6.dipole      q6q5.pot
```

grid points along  $Q_5$  and  $Q_6$

B3LYP/cc-pVDZ			
# Number of grids and data			
	9	9	1
#	q5	q6	Energy
	-27.93846251	-27.68241971	-6.7200061552e-02
	-19.84484851	-27.68241971	-5.5998358343e-02
	-12.85779017	-27.68241971	-4.1951916512e-02
	-6.33498780	-27.68241971	-2.3826848408e-02
	-0.00000000	-27.68241971	0.0000000000e+00
	6.33498780	-27.68241971	3.2112374522e-02
	12.85779017	-27.68241971	7.6869380577e-02
	19.84484851	-27.68241971	1.4322717035e-01
	27.93846251	-27.68241971	2.5528810814e-01

Note that the coupling term  $V_{65}$  is

$$V_{65} = V - V_6 - V_5$$

where  $V$  is the total energy, and  $V_5$  and  $V_6$  are the 1MR potential along  $Q_5$  and  $Q_6$ , respectively.  $V_{65}$  is often called an “intrinsic” coupling term.

## 2.3. 3MR-grid PES

In this section, we generate 3MR-grid PES for (Q4, Q5, Q6). Proceed to 2.grid\_h2co/2-3.3MR,

```
> cd 2.grid_h2co/2-3.3MR
> ls *pot *dipole
eq.dipole      q4.dipole      q5.dipole      q6.dipole      q6q5.dipole
eq.pot         q4.pot         q5.pot         q6.pot         q6q5.pot
```

Again, we use the existing information (q4, q5, q6, and q6q5) to reduce the cost. By placing these files in the same folder, the grid points are reduced from 729 (=9 x 9 x 9) points to 640 points.

### makePES

```
<makePES>
  <minfoFile value="../0.harmonic_h2co/h2co-b3lyp-dz.minfo" />
  <MR      value="3" />      MR=3 for 3MR-PES
  ...
  <grid>
    <ngrid value="9" />
    <mc3 value="4,5,6"/>    (Q4, Q5, Q6)
  </grid>
</makePES>
```

GaussianTemplate and resources.info are the same as before. Run MakePES,

```
> export SINDO_RSH=ssh
> sindo_jar=/path/to/sindo-4.0/jar
> java -cp "$sindo_jar/*" RunMakePES >& makePES.out
```

You will find in the output that the 2-mode terms, (Q4, Q5) and (Q4, Q6), are automatically added,

### makePES.out

Setup MakeGrid module

o ngrid = 9

o 1MR Grid:

4 5 6

o 2MR Grid:

(4,5) (4,6) (5,6)

o 3MR Grid:

(4,5,6)

1- and 2-mode terms that are subsets of the 3-mode terms are automatically added by the program.

the 3-mode term specified in the input

Enter GridPES generation:

After iteration over the grid points, we obtain pot and dipole files for (Q4, Q5), (Q4, Q6), and (Q4, Q5, Q6).

### makePES.out

Generating pot files.

o q5q4.pot [OK]

o q5q4.dipole [OK]

o q6q4.pot [OK]

o q6q4.dipole [OK]

o q6q5q4.pot [OK]

o q6q5q4.dipole [OK]

} Automatically added 2-mode terms

} The target 3-mode term

End of GridPES generation:

```
> ls q*
```

```
q4.dipole
```

```
q4.pot
```

```
q5.dipole
```

```
q5.pot
```

```
q5q4.dipole
```

```
q5q4.pot
```

```
q6.dipole
```

```
q6.pot
```

```
q6q4.dipole
```

```
q6q4.pot
```

```
q6q5.dipole
```

```
q6q5.pot
```

```
q6q5q4.dipole
```

```
q6q5q4.pot
```



## 2.4. Generic mode

In this section, we illustrate the generic mode for grid PES. Proceed to 2-4.1MR\_generic,

```
> cd 2.grid_h2co/2-4.1MR_generic
```

makePES.xml has <qchem> section as follow,


```
                                makePES
<makePES>
...
  <qchem>
    <program value="generic" />    "generic" means no specific program
    <title   value="B3LYP/cc-pVDZ" />
    <xyzfile value="makeGrid" />   set the name of xyz file
  </qchem>
</makePES>
```


Running the program creates makeGrid.xyz,


```
> sindo_jar=/path/to/sindo-4.0/jar
> java -cp "$sindo_jar/*" RunMakePES >& makePES.out
> ls
makePES.out  makePES.xml  makeGrid.xyz
```

makeGrid.xyz has the same format as in QFF (see Sec. 1.4). For the grid points written in the file, calculate the energy and dipole moments of formaldehyde using your favorite program. Then, compile the information to a file, makeGrid.dat, in a format as follow,

makeGrid.dat				
mkg-eq,	-114.507640,	0.686660E-15,	-0.112327E-13,	-0.814248E+00
mkg-q1-11-0,	-114.468383,	-0.536891E-01,	0.321103E-13,	-0.696204E+00
mkg-q1-11-1,	-114.485563,	-0.396727E-01,	-0.420800E-14,	-0.744696E+00
mkg-q1-11-2,	-114.496189,	-0.278582E-01,	-0.494537E-13,	-0.776962E+00

  
 ID

  
 Energy

  
 Dipole moment (dx, dy, dz)

The column must be separated by comma. The length / digit is arbitrary. The order of grid points (i.e., the order of the row) is also arbitrary. A sample is found in log2\_genpot/makeGrid.dat.

Then, run the program again to obtain the pot/dipole files.

```
> sindo_jar=/path/to/sindo-4.0/jar
> java -cp "$sindo_jar/*" RunMakePES >& makePES.out
> ls *pot
eq.pot  q1.pot  q2.pot  q3.pot  q4.pot  q5.pot  q6.pot
```

### 3. Multiresolution PES

---

In this section, we generate a multi-resolution PES [3], which combines different levels of electronic structure and functional forms. Here, we will use the following combination:

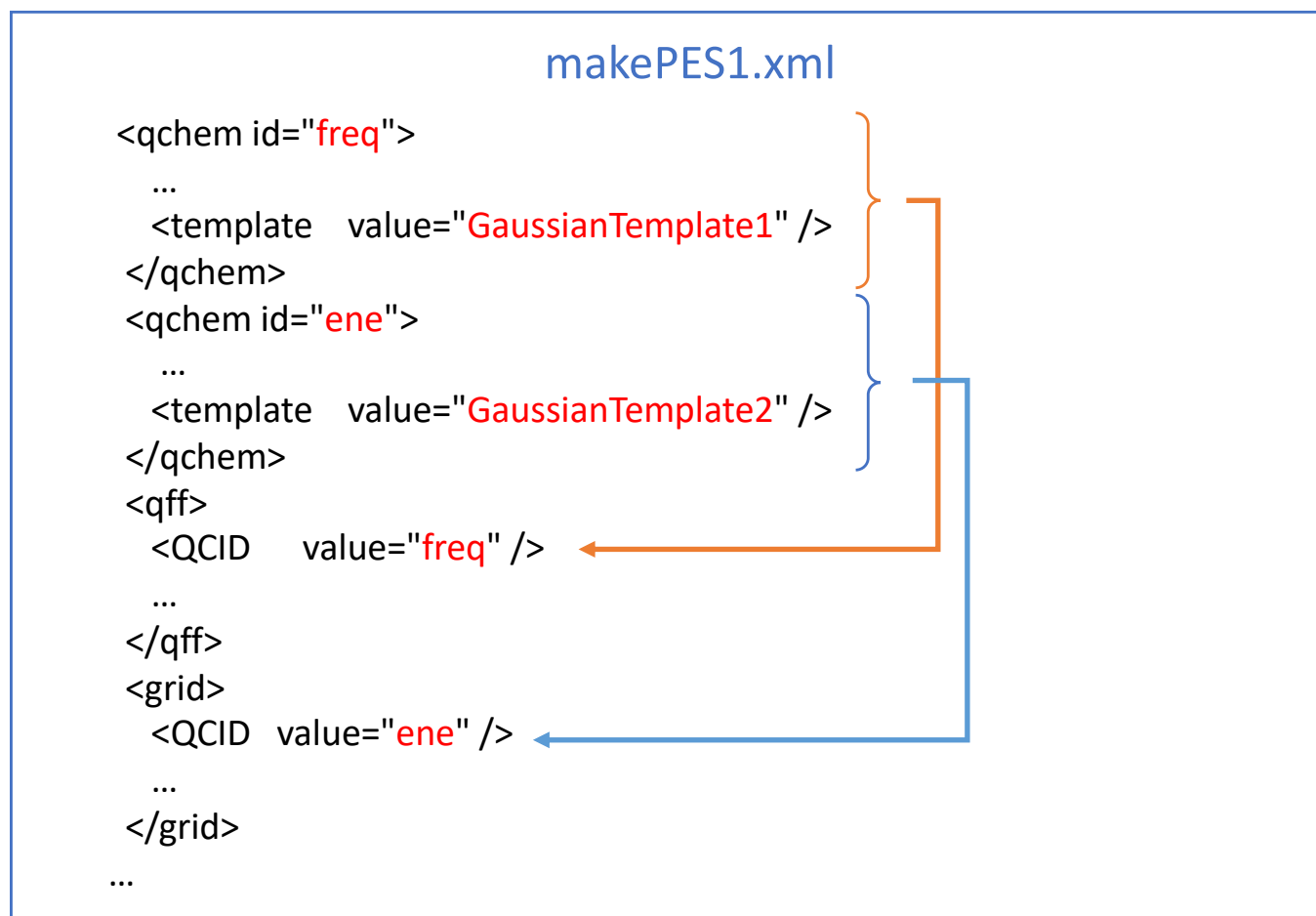
	Electronic Structure	Functional Form
1MR	CCSD(T)/cc-pVTZ	Grid-PES (11 points)
Strongly coupled terms (MCS>10)	CCSD(T)/cc-pVTZ	Grid-PES (9 points)
Weakly coupled terms (MCS>1)	B3LYP/cc-pVDZ	Grid-PES (7 points)
Other terms	B3LYP/cc-pVDZ	QFF

Mode coupling strength (MCS) is calculated from QFF coefficients [4].

Proceed to 3.mrpes\_h2co to find input files,

```
> cd 3.mrpes_h2co
> ls
GaussianTemplate1  GaussianTemplate2  GaussianTemplate3  log/
makePES1.xml      makePES2.xml      resources.info      run.sh
```

GaussianTemplate1 and GaussianTemplate2 are the template files for Gaussian to calculate the Hessian matrix (FREQ) and only the energy, respectively, at the B3LYP/cc-pVDZ level. In makePES1.xml, these files are associated with <qchem> sections with id="freq" and "ene", and the ID is associated with QCID of <qff> and <grid>.



## makePES1.xml

```
<qff>
  <QCID value="freq" />
  <mopfile value="prop_no_1.mop" />
  ...
</qff>
<grid>
  <QCID value="ene" />
  <ngrid value="11" />
  <mc1 value="1-6"/>
</grid>
<grid>
  <QCID value="ene" />
  <ngrid value="9" />
  <mcstrength value="10"/>
  <mopfile value="prop_no_1.mop"/>
</grid>
<grid>
  <QCID value="ene" />
  <ngrid value="7" />
  <mcstrength value="1"/>
  <mopfile value="prop_no_1.mop"/>
</grid>
```

1MR-grid PES

Grid PES with MCS > 10

Grid PES with MCS > 1

<qchem> sections are followed by one <qff> and three <grid> sections. MakePES processes these sections **in the order they appear** in the input file. In the present example,

1. Generate QFF  
prop\_no\_1.mop is created.
2. Generate 1MR-grid PES with ngrid = 11  
q1 - q6.pot / dipole files are created.
3. Generate 3MR-grid PES with ngrid = 9  
pot/dipole files with MCS > 10 are created.
4. Generate 3MR-grid PES with ngrid = 7  
pot/dipole files with MCS > 1 are created.

Step 1 (QFF) must precede Step 3 and 4, because prop\_no.1.mop is needed to calculate MCS. Note that we only need step4 for the final mrpes; however, we carry out step2 and 3 for a reference.

Modify resource.info and %NprocShared in GaussianTemplate1/2 for your system. In this sample, we run 8 processes of Gaussian with 8 cores (64 cores in total).

### GaussianTemplate1/2

```
%chk=#basename#.chk  
%NprocShared=8  
%mem=1GB
```

### resources.info

```
beluga01  
beluga01  
...  
beluga04  
beluga04
```

} 4 nodes x 2 = 8 processes

Run the program by typing,

```
> sindo_jar=/path/to/sindo-4.0/jar  
> java -cp "$sindo_jar/*" RunMakePES -f makePES1.xml >&  
makePES1.out
```

Find in the makePES1.out that the calculation is performed in the above order.

### makePES1.out

```
...
Enter QFF generation:
...
Generating prop_no_1.mop... Done!
} QFF generation

Setup MakeGrid module
...
Generating pot files.
o q1.pot [OK]
} 1MR-grid PES generation
...

Setup MakeGrid module

o Setup MCS: Read QFF Data via prop_no_1.mop ... [OK]
o ngrid = 9
o 1MR Grid:
  5 1 2 3 6 4
o 2MR Grid:
  (1,5) (2,5) (3,5) (1,6) (2,6) (3,6) (5,6) (2,3) (2,4) (4,6)
o 3MR Grid:
  (2,3,6) (2,4,6)
  Reads the output of QFF calc.
  Coupling terms with MCS > 10.0
```



When the calculation is done, you will find the following pot files in the current directory.

```
> ls *.pot
eq.pot      q3q2.pot    q5q1.pot    q6.pot      q6q4.pot
q1.pot      q4.pot      q5q2.pot    q6q1.pot    q6q4q2.pot
q2.pot      q4q2.pot    q5q3.pot    q6q2.pot    q6q5.pot
q2q1.pot    q4q3.pot    q5q4.pot    q6q3.pot
q3.pot      q5.pot      q5q4q3.pot  q6q3q2.pot
```

Note that the red ones are strongly coupled terms ( $\text{MCS} > 10$ ) generated by 9 grid points, and the blue ones are weakly coupled terms ( $\text{MCS} > 1$ ) generated by 7 grid points.

Before moving on to generate the PES at the CCSD(T)/cc-pVTZ level, we must save the PES@B3LYP data and clean up the current directory,

```
> mkdir pes_b3lyp
> mv minfo.files *.pot *dipole pes_b3lyp
```

This is because MakePES looks for pot/dipole files and minfo files in the current directory to restart the job.

The input file for MakePES looks as follow:

### makePES2.xml

```
<makePES>
...
<dipole value="false" />
<qchem id="ene">
...
  <title value="CCSD(T)/cc-pVTZ" />
  <template value="GaussianTemplate3" />
</qchem>
<grid>
  <QCID value="ene" />
  <ngrid value="11" />
  <mc1 value="1-6"/>
</grid>
<grid>
  <QCID value="ene" />
  <ngrid value="9" />
  <mcstrength value="10"/>
  <mopfile value="prop_no_1.mop"/>
</grid>
</makePES>
```

Dipole is turned off because CCSD(T) does not provide a dipole moment.

1MR-grid PES

Grid PES with MCS > 10

use the mop file generated in the last step.

The template input file for CCSD(T)/cc-pVTZ is as follow

### GaussianTemplate3

```
...  
#P CCSD(T)/CC-PVTZ NOSYMMETRY MAXDISK=240GB
```

Run the program by typing,

```
> sindo_jar=/path/to/sindo-4.0/jar  
> java -cp "$sindo_jar/*" RunMakePES -f makePES2.xml >&  
makePES2.out
```

When the calculation is done, you will find the following pot files in the current directory.

```
> ls *pot  
eq.pot      q3q2.pot    q5q1.pot    q6q1.pot    q6q4.pot  
q1.pot      q4.pot      q5q2.pot    q6q2.pot    q6q4q2.pot  
q2.pot      q4q2.pot    q5q3.pot    q6q3.pot    q6q5.pot  
q3.pot      q5.pot      q6.pot      q6q3q2.pot
```

Again, we will save all the CCSD(T) data to a directory,

```
> mkdir pes_ccsdt  
> mv minfo.files *pot pes_ccsdt
```

We now compile the PES data in one directory,

```
> mkdir pes_mrpes
> mv prop_no_1.mop      pes_mrpes    QFF
> cp $b3lypdir/*pot      pes_mrpes    2MR-, 3MR-Grid PES (MCS > 1)
> cp $ccsdtdir/*pot      pes_mrpes    1MR-grid and 2MR-, 3MR-grid PES (MCS > 10)
> cp $b3lypdir/*dipole   pes_mrpes    Dipole moment surface
```

We will use this PES for vibrational calculations. See the Users' guide to FSindo.

## 4. TIP and FAQ

---

### 4.1. How to terminate jobs.

When you want to stop the job, a safe way to terminate all Gaussian jobs is to create a file with a name “terminate”.

```
> touch terminate
```

Then, the job stops after the electronic structure jobs that are currently running are all finished.

If you want to immediately stop the job, you have to kill the main process, i.e., the java process responsible for RunMakePES. In that case, however, you may have to kill all child processes (= electronic structure jobs) manually.

### 4.2. How to restart the job.

MakePES looks for PES data (pot files and mopfile) and minfo files in a “minfo.files” directory before starting electronic structure jobs. The job is skipped if a minfo file is found, and starts from a point where it ended before.

Note that, for this reason, you have to move the PES data and the “minfo.files” directory to start a fresh new job.

## 5. References

---

### QFF

- [1] Ab initio vibrational state calculations with a quartic force field: Applications to  $\text{H}_2\text{CO}$ ,  $\text{C}_2\text{H}_4$ ,  $\text{CH}_3\text{OH}$ ,  $\text{CH}_3\text{CCH}$ , and  $\text{C}_6\text{H}_6$ , K. Yagi, K. Hirao, T. Taketsugu, M. W. Schmidt, and M. S. Gordon, J. Chem. Phys. **121**, 1383 (2004).

### Grid-PES

- [2] Direct vibrational self-consistent field method: Applications to  $\text{H}_2\text{O}$  and  $\text{H}_2\text{CO}$ , K. Yagi, T. Taketsugu, K. Hirao, and M. S. Gordon, J. Chem. Phys. **113**, 1005 (2000).

### Multiresolution PES

- [3] Multiresolution potential energy surfaces for vibrational state calculations, K. Yagi, S. Hirata, and K. Hirao, Theor. Chem. Acc. **118**, 681 (2007).
- [4] On the coupling strength in potential energy surfaces for vibrational calculations, P. Seidler, T. Kaga, K. Yagi, O. Christianse, and K. Hirao, Chem. Phys. Lett. **483**, 138 (2009).

# Appendix: List of all keys

## General keys

- minfoFile: file name  
The name of minfo file containing the vibrational data. The value is case sensitive.
- minfo\_folder: folder name  
The name of a folder where generated minfo files will be stored. The value is case sensitive.
- MR: 1/2/3  
The order of mode coupling expansion. Can take 1, 2, or 3. (default = 3)
- dipole: true/false  
Generates the dipole moment surface in addition to the PES, when true. (default = false)

- **activemode:** string of mode index  
Specifies active modes for PES generation. All modes are active by default. The mode numbers should be separated by comma or space. A hyphen can be used for a sequence of mode number. For example,

`<activemode value="1,2,3,5">`

is equivalent to,

`<activemode value="1-3 5">`

which means  $Q_1, Q_2, Q_3$ , and  $Q_5$  are active, and  $Q_4$  isn't.

Note that the modes can be set to inactive later in the vibrational calculations. Therefore, it is recommended to include as many modes as possible during the PES generation step.

Nevertheless, there are obvious cases where we want to select vibrational modes. For example, a solute in solvent, ligands in a protein, etc. A model that separates the inter- and intra-molecular modes is often used for cluster systems. `<activemode>` is useful for such purposes.



- interdomain: true / false

Calculates inter-domain coupling terms, when true. (default = false)

QFF計算では、実行された量子化学計算の情報から、Gradientで数値微分の場合はt<sub>ii</sub>j, u<sub>iii</sub>j、Hessianで数値微分の場合は3MRまでドメイン間カップリングが計算可能だが、interdomain = falseではこれらは出力されない。出力したい場合は、interdomain=trueで再計算する。

Hessianの場合、minfo.filesからデータを読み込めば、追加の量子化学計算なく3MRまで計算できたため、実は計算負荷の軽減にはならない。4MRは追加計算が必要となる。

## QCHEM section

ID: string

- program: string
  - gaussian ... Interface with Gaussian
  - generic ... Print the coordinates to a xyz file
- title: string
  - A title line that will be printed to PES files.

options for non-generic (=Gaussian)

- removefiles: true/false
  - Removes the input/output files of the quantum chemistry program, when true. (default = false)
- dryrun: true/false
  - Generates the input files for the quantum chemistry program and exits without execution, when true. (default = false)
- template: file name
  - The name of a template file to generate the input files for quantum chemistry jobs.

options for generic

- xyzfile: filename
  - Basename of a xyz file, where the coordinates of grid points are written. ".xyz" is automatically added. The PES data is read from minfo files in QFF and from a dat file, *filename.dat*, in GRID. (default = makeQFF for QFF and makeGrid for GRID)

## QFF section

- QCID: string  
The ID of associated <qchem>
- stepsize: real number  
The step size for numerical differentiations in dimensionless unit. (default = 0.5)
- ndifftype: grad or hess  
The type of numerical differentiations.  
grad : Numerical 3rd-order diff. of gradient.  
hess (default) : Numerical 2nd-order diff. of hessian.
- mopfile: file name  
The name of mop file, in which the QFF coefficients are written. (default = prop\_no\_1.mop)  
This format is compatible with the MIDAS software developed by Christiansen and coworkers.
- interdomain\_hc: true/false  
Prints the harmonic coupling, when true. (default = true)

- gradient and hessian: input/current  
Specifies where the gradient and Hessian are retrieved.  
input (default) : From the input minfo file.  
current : From the current calculation. (mkqff-eq.minfo)

“input” is useful for combining accurate geometry, gradient, and Hessian, read from the input minfo file, with lower-level cubic and quartic terms, which are calculated by MakePES module.

On the other hand, one might think of another strategy, where the geometry and coordinates are derived from a low-level of theory, and the QFF at a higher-level of theory. In that case, this option should be set to “current”, which incorporates the gradient and Hessian obtained from the current calculation.

- genhs: true/false  
Generate the 001.hs file, when true. (default = false)  
001.hs is a file which contains the QFF coefficients in the old format; however, this format is deprecated and not recommended to use unless for a debugging purpose to compare the result with the previous version of SINDO.

## GRID section

- QCID: string  
The ID of associated <qchem>
- ngrid: integer number  
The number of grid points along each coordinates. (default = 11)
- fullmc: true/false  
All the mode coupling up to the MR-th order is generated, when true. (default = false)
- mc1, mc2, mc3: string of mode index  
The 1, 2, or 3MR terms separated by comma or space.

Examples:

- `<mc1 value="1,2,3,5" />` or `<mc1 value="1-3 5" />`

generates grid points for Q1,Q2,Q3, and Q5.

- `<mc2 value="1,2, 1,4, 2,4, 3,4" />` or `<mc2 value="1,2, 1-3,4" />`

generates the grid points for (Q2, Q1),(Q4, Q1),(Q4, Q2), and (Q4, Q3).

- `<mc3 value="1,2,3, 1,2,4" />`

generates the grid points for (Q3, Q2, Q1) and (Q4, Q2, Q1).

- **mcsstrength:** real number  
The threshold value (in  $\text{cm}^{-1}$ ) to select the mode coupling term for generating the grid potential.  
The coupling terms with MCS larger than this value are generated.
- **mopfile:** file name  
The name of mop file to obtain MCS.

NOTE: One of fullmc, mc1, mc2, mc3, or mcsstrength must be present to specify the coupling terms.