Parmhess Documentation

Release v1.0 20171117

Ruixing WANG

May 10, 2018

Contents

CHAPTER 1

Overview

Manual for **Parmhess** v1.0, 20171117.

Parmhess implements Full, Partial, and Internal Hessian Fitting schemes for force-field parameterization, which are described in the following papers. Please refer to these papers for detailed theories.

1. "Analytical Hessian Fitting Schemes for Efficient Determination of Force-Constant Parameters in Molecular Mechanics" R. Wang, M. Ozhgibesov, H. Hirao. J.Comput. Chem. 2017, DOI: 10.1002/jcc.25100
2. "Partial Hessian Fitting for Determining Force Constant Parameters in Molecular Mechanics" R. Wang, M. Ozhgibesov, H. Hirao. J.Comput. Chem. 2016, 37, 23492359. DOI: 10.1002/jcc.24457

Katachi program performs *Katachi amendment* as described in these papers. **Tsubasa** is another program designed to prepare the input files for **Parmhess**.

If you found them useful in your research, please support our work by citing these papers.

1.1 Contact

- Ruixing Wang: ruixingw[%]terpmail.umd.edu
- Dr.Hajime Hirao: hhirao[%]cityu.edu.hk

Or, open an issue in the github page:

- Issue Tracker: https://github.com/ruixingw/parmhess/issues
- Source Code: https://github.com/ruixingw/parmhess

1.2 License

The programs are released under the BSD-3 license.

CHAPTER 2

Installation

Python3 (Anaconda)

Parmhess is written in Python3 and runs on Linux.

The easiest way to setup Python3 is to install **Anaconda**. It is freely available at https://www.anaconda.com/download/.

You should choose Anaconda for Linux (Python3.X), not Python2.7 nor for other operating system. Simply run `bash Anaconda3-x.x.x-Linux-x86_64.sh` to start the installation and follow the instruction. Root access is not necessary.

If you are an experienced Python3 user and prefer to use your own Python build, make sure that **numpy** and **pyyaml** be installed properly.

Gaussian 09 (G09)

QM and MM calculation is performed by Gaussian 09.

Any revision of G09 should work. However, there is a bug in G09 Rev. B.01 and thus special attention is needed in MK charge calculation. If you only have G09 B01, read the workaround in: http://ambermd.org/bugfixesat.html

AmberTools

AmberTools is used to identify the atom types and calculate RESP charges. After you have Anaconda installed, a simple command could be used to install a light version of AmberTools:

```
conda install ambertools -y -c http://ambermd.org/downloads/ambertools/conda/
```

Then, run antechamber and you should see the help of usage.

Parmhess

Download **Parmhess** from https://github.com/ruixingw/parmhess/releases

Extract the files to any folder. Create a soft-link to your $PATH. For example:

```
ln -s /PathToParmhess/parmhess.py ~/bin
```

Katachi

Download **Katachi** from https://github.com/ruixingw/katachi/releases.

Extract the files to any folder. Create a soft-link to your $PATH. For example:

```
ln -s /PathToKatachi/gokatachi.py ~/bin
ln -s /PathToKatachi/katachi.py ~/bin
```

Tsubasa

Download Tsubasa from https://github.com/ruixingw/tsubasa/releases Extract the files to any folder. Create a soft-link to your $PATH. For example:

```
ln -s /PathToTsubasa/tsubasa.py ~/bin
```

CHAPTER 3

Format of Inputs

Parmhess requires the following files as input:

1. LOG file of a QM frequency calculation by Gaussian.
2. FCHK file of a QM frequency calculation by Gaussian.
3. MM input file in Gaussian format. Includes charges, atom types and MM functions.
4. A file named "input.inp" that includes the filename of all above.

The LOG and FCHK file should be generated from a QM frequency calculation with "freq=intmodes". The FCHK file should be generated by `formchk -3 foo.chk foo.fchk`. The "-3" flag is important otherwise the internal Hessian will not be written into FCHK files.

An example of MM input file in Gaussian format is as follows.

mmH2O2.com:

```
%mem=1gb
#p amber=softonly geom=connectivity nosymm
iop(4/33=3,7/33=1)
freq=intmodes

MM

0 1
O-oh--0.410452   -0.718633164030   -0.118472295063   -0.054617618503
H-ho-0.410452   -1.023538245240    0.665457463989    0.436707052760
O-oh--0.410452    0.718637032315    0.118468958072   -0.054573365001
H-ho-0.410452    1.023507272500   -0.665430766999    0.436820814218

 1 2 1.0 3 1.0
 2
 3 4 1.0
 4

AmbTrs ho oh oh ho 0 0 0 0 0.0 XXXXXX 0.0 0.0 1.0
```

(continues on next page)

(continued from previous page)

```
HrmBnd1 ho oh oh XXXXXX 100.2486
HrmStr1 ho oh XXXXXX 0.97412
HrmStr1 oh oh XXXXXX 1.45667
Nonbon 3 1 0 0 0.0 0.0 0.5 0.0 0.0 -1.2
VDW ho  0.0000  0.0000
VDW oh  1.7210  0.2104
```

The unknown parameters to be determined should be written as "XXXXXX".

Preparing input files manually could be complicated. **Tsubasa** program was designed to automate these processes.

CHAPTER 4

Automated Preparation of Input Files

Tsubasa is designed to automatically prepare the input files for **Parmhess**. It could automatically do optimization (opt), MK charge calculation (resp), RESP charge calculation (antechamber), frequency calculation (freq) and preparation of MM input file.

The input for Tsubasa program is just coordinates and connectivity. An example is:

H2O2.gau

```
0 1
O     -0.718633164030    -0.118472295063    -0.054617618503
H     -1.023538245240     0.665457463989     0.436707052760
O      0.718637032315     0.118468958072    -0.054573365001
H      1.023507272500    -0.665430766999     0.436820814218

 1 2 1.0 3 1.0
 2
 3 4 1.0
 4
```

Save it to a file named *H2O2.gau*. Then run the following command within the same folder .. code-block:: bash

    tsubasa.py

A template config file will be copied to the same folder and named *H2O2.yml*. The format is YAML but is easy to understand. Below we attach the content of this file and make comments after a number sign (#).

```
g09rt: myg09boon      # myg09boon is the command to run Gaussian 09; change it to your␣
↪job submit
                      script. For example, running "myg09boon foo.com" should produce
                      foo.log in the same folder. If you do not use job management␣
↪system,
                    it should just be "g09".
g09a2rt: myg09a2boon          # Command to run Gaussian 09 for MK charge calculation
                                If you mainly use G09B01, which is problematic for MK␣
↪charge calculation,
```

(continues on next page)

(continued from previous page)

```
                                you can use another version for MK charge calculation.

antechamber: antechamber -c resp      # Commands to run antechamber. The charge type␣
↪can be changed
                                        # For metal system, "-j 5" may be added
clean: rm *gaussian*                  # will run at the last to clean the directory.

opthead: |                            # A block starts with a vertical bar (|)
  %mem=16gb                           # the content in the block is indented by 2␣
↪spaces.
  %nproc=12                           # This block defines the command used for␣
↪optimization
  #p b3lyp/6-31+g* geom=connectivity
  int=ultrafine symm=(loose,follow)
  opt=(verytight,maxstep=7,notrust)

  opt-title

opttail: |                            # The coordinates and connectivity will be␣
↪inserted
                                      # between the content of opthead and opttail
freqhead: |                           # This block defines the command used for␣
↪frequency calculation
  %mem=16gb                           # coordinates will be read from CHK files.
  %nproc=12
  #p b3lyp/chkbas int=ultrafine symm=loose geom=allcheck guess=tcheck freq=intmodes␣
↪iop(7/33=1)


resphead: |                           # This block defines the command used for MK␣
↪charge calculation
  %mem=16gb
  %nproc=12
  #p b3lyp/chkbas
  iop(6/33=2,6/42=17,6/41=10)
  int=ultrafine symm=loose
  pop=mk                               # If any VDW radius is missing, use pop=(mk,
↪readradii) ...
  geom=allcheck guess=tcheck


resptail: |                           # ... and add the VDW radius here


mmhead: |                             # this block should not be changed.
  %mem=12gb
  #p amber=softonly geom=connectivity nosymm
  iop(4/33=3,7/33=1)
  freq=intmodes

  MM
```

You should check the content of config file before going to the next step. After that, run the command again:

```
tsubasa.py
```

The program will start the process and finally produce all necessary files for Parmhess if all the calculations

succeed. The intermediate files will be named as: optH2O2.com/log/chk, freqH2O2.com/log/chk/fchk and respH2O2.com/log/chk/mol2.

The detailed usage is as follows.

```
> tsubasa.py --help
usage: tsubasa.py [-h] [-i GAUFILE] [-c YMLFILE] [--readvdw EXTERNALVDWFILE]
                  [--startfrom {resp,antechamber,freq,buildMMfile}]
                  [--stopafter {opt,resp,antechamber,freq}]
                  [--improper IMPROPERLIST]

optional arguments:
  -h, --help            show this help message and exit
  -i GAUFILE            Inputfile including molecular specs and
                        connectivity
  -c YMLFILE            Tsubasa config file.
  --readvdw EXTERNALVDWFILE
                        If provided, read external vdW parameters from
                        EXTERNALVDWFILE
  --startfrom {resp,antechamber,freq,buildMMfile}
                        Start from a certain step.
                    Choices=['resp','antechamber','freq','buildMMfile']
  --stopafter {opt,resp,antechamber,freq}
                        Stop after a certain step.
                        Choices=['opt','resp','antechamber','freq']
  --improper IMPROPERLIST
                        Add improper functions, IMPROPERLIST should be
a list like "h5 * c2 *,c3 * o *"
```

GAUFILE and *YMLFILE* is the .gau and .yml file aforementioned. If they are not specified, the program will search for a .gau file and a .yml file in the current folder. If a .gau file is found but not a .yml file, a template config file will be copied to the current folder. If both are found, the program will start to prepare the inputs.

The program included the vdW parameters in GAFF (gaff.dat) as a database. However, if any vdW parameter is missing, it can be provided by *–readvdw EXTERNALVDWFILE*. The format of this file should be same to the vdW parameters in AMBER(for example, gaff.dat). If the missing vdW parameters are not provided, the placeholder RADII and WELLDEPTH will be written into the generated MM input file.

startfrom and *stopafter*: control the steps. For large systems, it is better to do QM calculations manually and then provide the results to tsubasa. The name of provided files should be same to those tsubasa generated. For example, if a FOO.gau is provided as input, the files should be optFOO.com/log/chk, freqFOO.com/log/chk/fchk, respFOO.com/log/chk/mol2. Mol2 file can be modified to change atom types, which are usually not satisfactorily determined. After modifying the Mol2 files, *–startfrom buildMMfile* can be used.

If improper terms are defined, they can be provided by *–improper IMPROPERLIST*, so that the internal coordinates for improper torsions could be included to perform IHF.

CHAPTER 5

Run Parameterization

Once the input files are prepared, simply run:

```
parmhess.py input.inp
```

The determined parameters will be saved in result files whose name begin with “fhf_result_”, “phf_result_” and “ihf_result_”.

For example, the content of ihf_result_mmH2O2.com is:

```
%mem=1gb
#p amber=softonly geom=connectivity nosymm
iop(4/33=3,7/33=1)
freq=intmodes

final

0 1
O-oh--0.410452   -0.718633164030    -0.118472295063    -0.054617618503
H-ho-0.410452   -1.023538245240     0.665457463989     0.436707052760
O-oh--0.410452    0.718637032315     0.118468958072    -0.054573365001
H-ho-0.410452    1.023507272500    -0.665430766999     0.436820814218

 1 2 1.0 3 1.0
 2
 3 4 1.0
 4

AmbTrs   ho  oh  oh  ho   0   0   0   0    0.000  1.552  0.000  0.000   1.0
HrmBnd1   ho  oh  oh   68.779  100.24860
HrmStr1   ho  oh   550.931  0.97412
HrmStr1   oh  oh   346.974  1.45667
Nonbon 3 1 0 0 0.0 0.0 0.5 0.0 0.0 -1.2
VDW    ho  0.0000  0.0000
VDW    oh  1.7210  0.2104
```

CHAPTER 6

Run Katachi Amendment

After parameterization, run:

```
katachi.py ihf_result_mmH2O2.com 0 calcall 100
```

The result will be named katachi_ihf_result_mmH2O2.com.

Here are the last lines of the example:

```
AmbTrs   ho  oh  oh  ho   0   0   0   0    0.000  1.552  0.000  0.000   1.0
HrmBnd1 ho oh oh   68.779  98.2714
HrmStr1 ho oh   550.931  0.96915
HrmStr1 oh oh   346.974  1.44835
Nonbon 3 1 0 0 0.0 0.0 0.5 0.0 0.0 -1.2
VDW    ho  0.0000  0.0000
VDW    oh  1.7210  0.210
```

The detailed usage is as follows.

```
> katachi.py --help
usage: katachi.py [-h] mmresult loopid opt convthreshold
positional arguments:
  mmresult       parameterized result MM file.
  loopid         loopid
  opt            opt or calcall
  convthreshold  convergence threshold
```

loopid is the number of starting cycle. It should be 0 if it is a fresh run. If Katachi was interrupted in the middle, this number can be set to that of the latest cycle to restore the process.

opt or *calcall* controls the commands used for MM optimization. If *opt* is specified, *opt=(nomicro,cartesian)* will be used for MM optimization, and after the convergence, *opt=(nomicro,cartesian,tight,calcall)* will then be used and start

again from the previous result. If calcall is specified, all steps will use the latter keywords. In practice, we found that using calcall is usually better.

convthreshold is the cycle number threshold. If the result does not improve any more in the given cycles, the program will stop. In our tests, we used 100 for this threshold; that means, if the result does not improve in 100 cycles, the program will stop and use the best result before.