CGenFF Force Field Development Tutorial

For Small Drug Molecules

Zilin Song

Before You Start

To understand the principles for small organic molecules force field development and implement all methods or technique mentioned in this tutorial, readers are required to have the following background knowledge noted in the PT 0.

Although PT 0 is distorted between contexts, the information contained in PT 0 is critical for PT 2 Practice and will be frequently used in the actual development process.

Basic skills in Python programming (for automatic process) and CHARMM scripting is prerequisite for implementation of ***ANY*** practice in this tutorial.

This tutorial orients the Python 3 and CHARMM c42 and the script provided using the oriented scripting language are completely valid. Solutions based on other techniques / software / platforms are potentially possible but not guaranteed.

All other members of TAO group @ SMU take credit on realizing this tutorial, especially Professor Peng Tao and PhD student Hongyu Zhou.

Zilin Song
14 Feb 2019

0. Basics

0.0. Coding Ethic

“Be a nice man, write your comments.”

0.0.0. Comment before code.

Comments that provides ***redundant*** explanation must be provided for each line of code that is critical to the functionality of the script.

0.0.1. Comment before code.

Comments that provides ***redundant*** explanation must be provided for each line of code that is critical to the functionality of the script.

0.0.2. Comment before code.

Comments that provides ***redundant*** explanation must be provided for each line of code that is critical to the functionality of the script.

0.0.3. Comment before code.

Comments that provides ***redundant*** explanation must be provided for each line of code that is critical to the functionality of the script.

0.0.4. Comment before code.

Comments that provides ***redundant*** explanation must be provided for each line of code that is critical to the functionality of the script.

0.0.5. Comment before code.

Comments that provides ***redundant*** explanation must be provided for each line of code that is critical to the functionality of the script.

0.0.6. Comment before code.

Comments that provides ***redundant*** explanation must be provided for each line of code that is critical to the functionality of the script.

0.0.7. Comment before code.

Comments that provides ***redundant*** explanation must be provided for each line of code that is critical to the functionality of the script.

0.1. Style Guide

This regulation contains the Python 3 grammars that would be used in the rest part of the tutorial, and is subjected to change with future development.

Naming of the local/global variables, functions, parameters, classes in Python scripting would also follow the following regulations.

0.1.0. Python functions & parameters.

```
# function comments.
def long_function_name(
                var_one,                # var_one comments
                var_two,                # var_two comments
                var_three               # var_three comments
                ):
        print(var_one)
```

0.1.1. Python list

```
# list comments.
#       a, b, c, d, e
List = [1, 2, 3, 4, 5]
```

0.1.2. Python if

```
If (expression):
        function_name(var_1)
elif(expression):
        alternative_function_name(var_3)
else:                               # else is mandatory
        pass                        # pass as well
```

0.1.3. CHARMM

The optional keyword in CHARMM are denoted with *:

```
WRITe coor *PDB UNIT 99
```

The PDB keyword is optional.

0.1.4. other situations.

Style Guide for Python 3

CHARMM Syntax

0.2. Gaussian Z-matrix opt

The Gaussian keyword

Opt=Z-matrix

would fix all other internal coordinates (IC) and optimize only the IC specified with a codename in the Z-matrix.

EXAMPLE I

```
%mem=250GB
#HF/6-31G* Opt=Z-matrix

PNMPT1...HOH, hf/6-31g*

0 1
 O          42.68996798   16.84816796   26.02809002
 C          42.69395893   18.03253157   25.60284012
 N          43.47305569   18.81674131   24.71891488
 C          44.16092565   18.19494035   23.58823633
 C          44.11922865   19.32314205   22.59471535
 S          43.80085188   20.70753577   23.63582012
 C          42.36708287   19.85541064   24.39002688
 C          41.88475343   19.23133587   25.76955544
 H          45.14013138   17.93668891   23.93372487
 H          40.82850376   19.08813836   25.86299583
 H          42.02538254   20.78469473   24.79570158
 H          44.98544735   19.36061196   21.96768971
 H          43.25354635   19.23143773   21.97255421
 H          42.19883502   19.80434515   26.61685298
 H          43.71603144   17.32443648   23.15327409
 X   15     2.1     4       90.     9       90.
 O   15     roh    16       90.     4      180.
 H   17   0.9572   15   127.74    16      180.
 H   17   0.9572   15   127.74    16        0.

 roh    2.1
```

In the Gaussian opt script above, only the IC entry specified by "**roh**" (distance between O and H15) would be changed during the optimization. Other IC entries would be fixed.

Also keep in mind that Gaussian requires empty lines in the end of the input file, otherwise Gaussian would raise the famous "Segmentation Violation" error. (Yeah, this is quite weird in terms of program design.)

0.3. Natural Internal Coordinates.

The Natural Internal Coordinates (NICs) is a coordinating system completely different from conventional Cartesian coordinates or Z-matrix coordinates or redundant internal coordinates. **NICs is the normalization of the redundant internal coordinates.**

> These coordinates involve the use of individual bond displacements as stretching coordinates, but linear combinations of bond angles and torsions as deformational coordinates.
>
> The major advantage of natural internal coordinates in geometry optimization is their ability to significantly reduce the coupling, both harmonic and inharmonic, between the various coordinates.
>
> Compared to natural internals, *Z*-matrix coordinates arbitrarily omit some angles and torsions (to prevent redundancy), and this can induce strong anharmonic coupling between the coordinates, especially with a poorly constructed *Z*-matrix. Another advantage of the reduced coupling is that successful minimizations can be carried out in natural internals with only an approximate (*e.g.*, diagonal) Hessian provided at the starting geometry. A good starting Hessian is still needed for a transition state search.
>
> (From Q-Chem 4.3 manual)

Several references listed below:
1. P. Pulay, G. Fogarasi, F. Pang, J. E. Boggs. J. Am. Chem. Soc., 1979, 101, 2550-2560;
2. P. Pulay, G. Fogarasi, G. Pongor, J. E. Boggs, A. Vargha. J. Am. Chem. Soc., 1983, 105, 7037-7047
3. Geza Fogarasi, X. Zhou, P. W. Taylor, P.Pulay. J. Am. Chem. Soc., 1992, 114, 8191-8201;

Method for constructing NICs are included in ref. 1, 2 and 3 and described in details below.

0.3.0. Construction of NICs.

FYI: The NICs are constructed basing on the redundant IC system, i.e., Dihedrals, Angles, Lengths

Step 0:
Identify backbone for the molecule:

Regulations for identifying the backbone.

1. Remove all (– H) (– F/Cl/Br) (– OH) groups from the molecule.
2. That is it!
3. In another words: all single valent groups can be treated as local side chains.

EXAMPLE I:
$H_3C – CH_2 – CH_3$
Backbone: C – C – C

EXAMPLE II:
Benzene: C_6H_6
Backbone: C_6 ring structure

Step 1:

Calculate the degrees of freedom of the whole molecule:

3N-6 or;
3N-5 for linear molecules

This number denotes the number of entries in the NIC for the molecules

Step 2:

Identify the NIC entries for bond stretching:

i.e. The number of bonds in the molecule: each bond is an NIC entry.

Step 3:

Identify the entries on each atom.

For molecules with ring systems, 4-, 5- and 6-member rings are considered as follows.

EXAMPLE I:
4-member ring:

```
C1 – C2
|       |
C4 – C3
```

Redundant ICs:

Dihedral (x, y, z, w)	as	*dy*
Angle (x, y, z)	as	*ay*
Dihedral (4-1-2-3)	as	*d1*
Dihedral (1-2-3-4)	as	*d2*
Dihedral (2-3-4-1)	as	*d3*
Dihedral (3-4-1-2)	as	*d4*
Angle (4-1-2)	as	*a1*
Angle (1-2-3)	as	*a2*
Angle (2-3-4)	as	*a3*
Angle (3-4-1)	as	*a4*

NICs:

a1 – a2 + a3 – a4	-> ring deformation
d1– d2 + d3 – d4	-> puckering

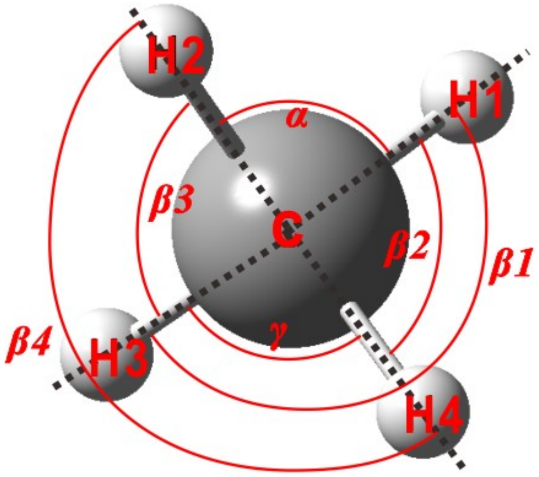
5-member ring & 6-member ring can be constructed using ref. 1. Table III

Step 4:

For each atom on backbone, mostly divalent O/S, C and N atoms, construct NIC entries under the guidance of ref. 1. Table III.

EXAMPLE I:

Methylene (sp^3): carbon connecting to 2 H atoms and 2 non-H atoms.



The H3 and H4 atoms under the C atom in the figure could be denoted as X and Y.

Specify the angles as shown in the figure as redundant ICs.

And the NICs are as follows:

1. C – H1 bond stretching;
2. C – H2 bond stretching;
3. C – H3 bond stretching;
4. C – H4 bond stretching;
5. CH_2 scissoring *5a + r*;
6. CXY scissoring *a + 5r*;
7. CH_2 rocking*b1 – b2 + b3 – b4*;
8. CH_2 wagging *b1 + b2 – b3 – b4*;
9. CH_2 twisting *b1 – b2 – b3 + b4*;

Step 5:

Identify the entries of side chains groups that contains heavy/non-H atoms.

Note that, for methylene (sp^3), or methylene (sp^2) on ring structures:

DO NOT INCLUDE the entries evolves X-C-Y mode: they are included in the ring NIC specifications.

Step 6:

For dihedrals:
X – C – C – Y and X – C = C – Y

The dihedrals itself is one entry in the NICs.

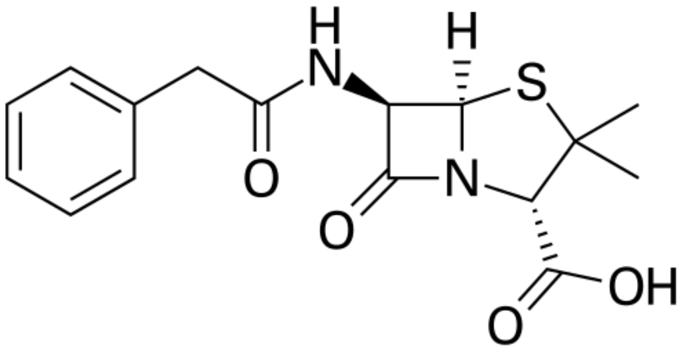
DO NOT INCLUDE the entries already included in the ring NIC specifications.

Step 7:

Check if the total number of NICs is (3N-6).

0.3.1. Example.

Determine all the NIC entries for PNM-G (41 atoms).



1. Degree of freedoms = 41 * 3 – 6 = **117**	-> the total number of entries of NIC	
2. Bonds	= 43	entries
Sum = 43		
For the Benzene ring on the left:		
3. Benzene ring	= 6	entries
4. H on benzene ring wagging	= 1 * 5	entries \
5. H on benzene ring rocking	= 1 * 5	entries from
6. Backbone chain on benzene wagging	= 1	entries methane sp^2
7. Backbone chain on benzene rocking	= 1	entries /
Sum = 18		
For 4-member ring:		
8. 4-member ring	= 2	entries
9. (C with C=O) C=O wagging	= 1	entry – from methylene sp^2
10. (C with H-C-S) CH rocking & rocking’	= 1 + 1	entries – from methine sp^3
11. (C with HN – C -H)		
CH rocking & rocking’	= 1 + 1	entries \ from
N – C – Y & N – C – Z deformation	= 1 + 1	entries / methane sp^3
** Trivalent N in the 4-member ring has being all constraint by ring entries**		
Sum = 9		
For 5-member ring:		
12. 5-member ring:	= 4	entries
13. (C with C – COOH)	= 4	entries – from methylene sp^3
14. (C with C – CH_3(– CH_3))	= 4	entries – methylene sp^3. CH_3 =H
** Divalent S in the 5-member ring has being all constraint by ring entries**		
Sum = 12		

For ring – ring butterfly:

15. ring – ring butterfly = 1 entry – from ref. 2, 1b.

Sum = 1

For (– CH_2 – C(=O) – NH –) between benzene & b-lactam ring:

16. CH_2: = 5 entries – from methylene (sp^3)
17. C=0: = 3 entries – from methane (sp^2)
18. NH: = 3 entries – from imino (sp^3)
19. Dihedral torsions:

Backbone:

Ben – C – C (= O) – N – 4-member_ring

For C – C = 2 entries

For C – N = 2 entries

Sum = 15

For COOH on 5-member ring:

20. C = 3 entries, methylene (sp^2)
21. O (in **O** – H) = 2 entries, see **0.3.2a. Appendix**

Sum = 5

For 2 methyl groups on 5-member ring:

22. C = 5 * 2 entries, from methane(sp^3)

Sum = 10

For dihedral torsions between parts:

23. Dihedral torsions cross C – C (=O) – N … 4_member_ring

Backbone

```
C – C – N – C – C –
    ‖       |   |
    O   O = C – N –
```

= 1 entry

24. Dihedral torsions cross 5_member_ring … COOH

Backbone:

```
– S – C (– me) – me
      |
  H – C – C – O – H
      |   ‖
    – N   O
```

= 2 entries

25. Dihedral torsions cross benzene_ring … $C(H_2)C(O)N(H)$ – group

Backbone (atoms in green denotes some part of the benzene ring):

```
 C –  C – C – C
 |    |   |   ‖
3Cs – C   H₂  O
```

= 1 entry

Sum = 4

Sum of entries

43 + 18 + 9 + 12 + 1 + 15 + 5 + 10 + 4 = 117

0.3.2a. Appendix: Natural Internal Coordinates for Divalent Oxygen/Sulfur Atoms.

NICs for divalent oxygen are not included in ref 1 or 2 as the authors believes these NICs are obvious. But to clarify, and they are specified below. The authors of this tutorial do not take responsibility for these specifications and **the readers should reply on their own judgment to decide whether these methods are applicable to their case.**

Divalent Oxygen/Sulfur

H – O/S – H -> #1

X – O/S – H -> #2

X – O/S – Y -> #3

For #1,

The only NIC entries is the H – O/S – H bond angles and 2 O – H bonds.

For #2,

This case requires additional attention. It is critical that the – O – H group is not included in the backbone. They should be regarded as a H atom attaching to the backbone Carbon atom and its NICs should be specified after the NICs on the backbones are settled.

The 3 NICs of divalent oxygen atoms should be specified as follows:
1. The O – H bond should be 1 independent NIC entry;
2. The X – O – H bond angle itself should be 1 independent NIC entry;
3. The Wilson wagging of H should be used as a hint, rather than the dihedrals.

    In this case, the NICs are Wilson wagging of the H atom wagging A – X – O plane, where A are **one of the atoms (including H)** connecting to X.

    Note that not all A should be included in the NIC, one NIC entry with only one of the A atoms is sufficient.

EXAMPLE I:

Methanol.

```
   H
   |
   C – O – H
  / \
 H   H
```

NICs:

i. The O – H bond;
ii. Angle H – O – C;
iii. H wagging out of H – C – O plane, should be specified for 1 of the H atoms only.

EXAMPLE II:

Ethanol.

```
    H
    |
    C – O – H
   /  \
 H   CH3
```

NICs:

i. The O – H bond;
ii. Angle H – O – C;
iii. H wagging out of (C or H) – C – O plane.

EXAMPLE III:

An extreme case:

In the following molecule, X, Y and Z are different atoms or atoms groups.

```
    X
    |
    C – O – H
   /  \
 Y   Z
```

NICs:

i. The O – H bond;
ii. Angle H – O – C;
iii. H wagging out of (X or Y or Z) – C – O plane.

For #3,

The NIC entry are the X – O – Y bond angle and relevant dihedral angles (normally should be included in the torsion & twisting of dihedral angles on the backbone).

Note that in the case of rings, if X – O – Y is on a ring structure, all the NIC from the divalent oxygen would be included in the ring NICs specification.

0.4. Schachtschneider/Snyder Format for Large Size Matrix Input

The Schachtschneider/Snyder (SS) format is the input protocol for large size 2D matrix input. The SS format only specifies the location of non-zero values in the matrix in the manner:

(column number) (row number) (value)

EXAMPLE I:

Original matrix:

$$\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 5 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 5 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 1
\end{bmatrix}$$

SS format:

1 1 1			
2 2 1			
3 3 1			
4 4 1			
5 5 5	5 6 1		
6 5 1	6 6 5		
7 7 1	7 8 -1	7 9 1	7 10 -1
8 7 1	8 8 1	8 9 -1	8 10 -1
9 7 1	9 8 -1	9 9 -1	9 10 1

only non-zero values are specified

0.5. Usage of CHARMM MOLVIB Module

The MOLVIB module is actually an independent program so that the CHARMM scripting syntax is NOT applicable.

The current section requires full understanding of section 0.3 & 0.4.

The MOLVIB module is a general-purpose vibrational analysis program for small molecule systems (< 100 atoms). Upon the consideration that most MOLVIB documentation is currently not available or not precise, the usage of MOLVIB facility in CHARMM is demonstrated individually in this tutorial.

Reference:
CHARMM MOLVIB documentation

The following syntax notations excludes those keywords that are irrelevant to the scripting in tutorial.

0.5.0. MOLVIB syntax

```
MOLVIB –
         NDI1 {int}           –
         NDI2 {int}           –
         NDI3 {int}           –
         [NATOm {int}]        –
         [NOTOpology]         –
         [SECOnd]
command-spec
   [data-structure]
command-spec
   [data-structure]
command-spec
   [data-structure]
   …
END
```

Keywords in MOLVIB

NDI1 {int}:	Number of vibrational degrees of freedom, usually 3×N – 6;
NDI2 {int}:	Number of IC's left after transformation by first U matrix.If only one U matrix is used, NDI2 should be the same as NDI1;
NDI3 {int}:	Number of primitive internal coordinates (PICs);
NATOm {int}:	Number of atoms in the molecule. Must be used together with the NOTOpology flag;

NOTOpology:	Molecular topology data from CHARMM would NOT be used, all data required is to be read inside the specifications within MOLVIB (using the CARTesian section command). Must be used with the NATOm flag.
SECOnd:	Calculate second derivatives (force constants in Cartesian coordinates) and pass them to MOLVIB. Calculations are done by CHARMM ENERgy routine, so all preconditions for ENERgy calculations must be met. The SECOnd keyword conflicts with the FMAT command-spec and in most cases only one of them should present.

command-spec & data-structure in MOLVIB

Syntax for command-spec & data-structure section:

```
Keyword [int] [int] [int] [int]
   data-structure-entries
   data-structure-entries
   data-structure-entries
```

Keywords:

GFX

Vibrational problem in Cartesian coordinates.

PRNT [int]

Set print level in MOLVIB output, 0 – 5 available. 5 suggested for testing run, 0 suggested for production run.

CART

Read in Cartesian coordinates from following data-structure. The number of entries is specified by the NATOm keyword.

data-structure:

```
X_coor          Y_coor          Z_coor      AtomicMass
```

EXAMPLE:

```
42.323951   16.875779   25.708834   15.9990
42.543261   18.054819   25.527664   12.0110
```

IC

Read in primitive internal coordinates from following data-structure.

data-structure:

```
ICType     atom1     atom2     atom3     atom4
```

Note that ICType specifies the type of ICs in each entry.

ICType = 1: atom 1 – 2 bond stretching
ICType = 2: atom 1 – 2 – 3 bond angle bending
ICType = 3: atom 1 – 4 angle with atom2 – atom3 – atom4 plane

```
                atom2
                /
atom1 – atom4                  (Wilson Wag)
                \
                atom3
```

ICType = 4: atom 1 – 2 – 3 – 4 dihedral angle
ICType = 5: atom 1 – 2 – 3 linear bend in atom 1 – 2 – 4 plane
ICType = 6: atom 1 – 2 – 3 linear bend perpendicular to atom 1 – 2 – 4 plane

```
atom1 – atom2 – atom3
        \          /
         \        /
          atom4
```

EXAMPLE:

```
1  1  2  0  0  ! denotes bond stretching between atom 1 and 2;
2  1  2  3  0  ! denotes bond angle from atom 1 2 3;
3  1  3  4  5  ! denotes improper dihedral from atom 1 – 4 (– 3) – 2;
4  9 19 12  8  ! denotes dihedral angle from atom 9 – 19 – 12 – 8.
```

UMAT int0 int1 int2

Read in U Matrix for PIC -> NIC transformation

int0: defines format of data-structure;
0: Schachtschneider/Snyder format (the only option)

int1: 1/0: do/don't normalize rows of U.

int2: Fortran unit for U matrix read;
0: read for following data-structure section

data-structure:

see section 0.4.

EXAMPLE:

see section 0.5.2. & 0.5.3

To denote the end of this section, an extra line with "–1" must be appended to the last line of data-structure.

PED int0 int1

Perform symbolic potential energy distribution (PED) analysis, each PED entry denotes a vibration frequency contributed by one or multiple NIC entries.

int0: NGRUP – number of coordinate groups to be defined, normally 0.

int1: NCUTP – cutoff level, PED contributions < NCUTP % will be ignored.

data-structure:

Number PEDName(string)

EXAMPLE:

```
1     sC1=O2      ! stretching of C1 = O2 bond
2     rcC2=O3     ! rocking of C2 = O3 bond
3     scC3CH2     ! scissoring of 2 H atoms on C3
```

To denote the end of this section, an extra line with "–1" must be appended to the last line of data-structure.

FMAT int0 int1 int2

Read in F matrix, the second derivatives of energy with regard to coordinates. This section must be present if SECOnd keyword is absent.

int0: Specify format:
0: Schatschneider/Snyder format
1: GAUSSIANxx format (copy from Gaussian output)

int1: 1/0: do/don't symmetrize

int2: Fortran unit for U matrix read;
0: read for following data-structure section

data-structure:

see section 0.5.2. & 0.5.3.

EXAMPLE:

see section 0.5.2. & 0.5.3.

SCALE:

Scale the vibrational frequencies by a factor, the data structure of this section is a single line input consists of the number as the factor.

data-structure:

see EXAMPLE below.

EXAMPLE:

```
SCALE
        0.89
```

END:

Denotes the end of MOLVIB command routine and launch MOLVIB calculation.

0.5.1. Construction of Natural Internal Coordinates in MOLVIB

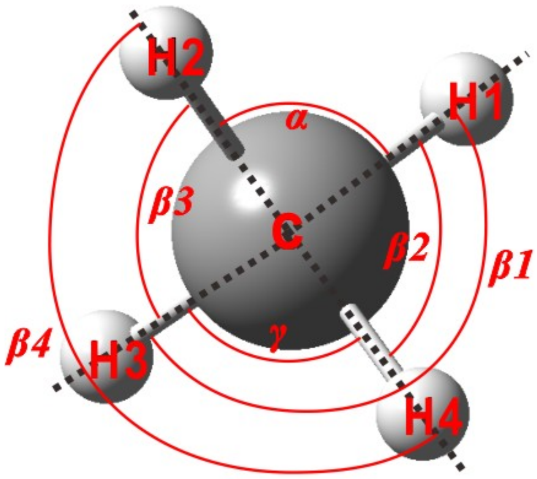
Principle:

The primitive IC entries (denoted as P) are multiplied by a matrix to construct the NIC entries (denoted as N), this matrix is called the U-Matrix.

$$P \times U = N$$

EXAMPLE:

For a methane molecule CH_4.



Entries of primitive ICs (P)
1. C – H1 bond;
2. C – H2 bond;
3. C – H3 bond;
4. C – H4 bond;
5. H1 – C – H2 angle, α;
6. H3 – C – H4 angle, γ;
7. H1 – C – H3 angle, $\beta 1$;

8. H1 – C – H4 angle, *β2*;
9. H2 – C – H3 angle, *β3*;
10. H2 – C – H4 angle, *β4*.

Entries of NICs (N), use ref 1 in section 0.3.0. as guidance.:
1. C – H1 bond stretching;
2. C – H2 bond stretching;
3. C – H3 bond stretching;
4. C – H4 bond stretching;
5. CH_2 scissoring *5a + r*;
6. CXY scissoring *a + 5r*;
7. CH_2 rocking*b1 – b2 + b3 – b4*;
8. CH_2 wagging *b1 + b2 – b3 – b4*;
9. CH_2 twisting *b1 – b2 – b3 + b4*;

So that:

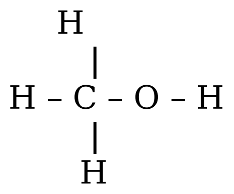
$$P \times U = N$$

$$\begin{bmatrix} C-H1 \\ C-H2 \\ C-H3 \\ C-H4 \\ \alpha \\ \gamma \\ \beta 1 \\ \beta 2 \\ \beta 3 \\ \beta 4 \end{bmatrix}^T \times \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 1 \end{bmatrix} = \begin{bmatrix} C-H1 \\ C-H2 \\ C-H3 \\ C-H4 \\ 5\alpha+\gamma \\ \alpha+5\gamma \\ \beta 1-\beta 2+\beta 3-\beta 4 \\ \beta 1+\beta 2-\beta 3-\beta 4 \\ \beta 1-\beta 2-\beta 3+\beta 4 \end{bmatrix}^T$$

0.5.2. MOLVIB scripting with Gaussian optimized QM data.

The following script illustrates the utilization of MOLVIB module basing on Gaussian QM calculation. This script would calculation the vibrational spectrum of a CH_3OH molecule.

NICs for CH_3OH



1. Degrees of freedom: 6 × 3 – 6 = 12
2. Bond stretching: 4 entries (O – H bond excluded, see section 0.3.2.)
3. For carbon atoms 5 entries from methyl sp^3
4. For divalent oxygens 3 entries see section 0.3.2.

Sum of NIC entries
4 + 5 + 3 = 12

CHARMM input script

(some redundant comments should be deleted before run.):

```
* PROJ Force Field Dev - small drug molecules
* Molecular vibration frequency calculation CH3OH
* Based on Gaussian MP2/6-31Gd vibrational modes
* Zilin Song, 13 Jan 2019
*

BOMLev -1
FASTer 1

! {  CHARMM MOLVIB Module is used for molecular vibrational calculation  }

MOLVIB -
  NDI1 12 -       ! number of vibrational degrees of freedom
  NDI2 12 -       ! number of ICs left after first UMAT transformation
  NDI3 13 -       ! number of primitive ICs
  NATOM 6 -       ! number of atoms/ entries in the CART section
  NOTO                  ! read topology from CART section
GFX               ! vibrational analysis in Cartesian coordinates.
PRNT 5                  ! print level 5 -> for testing
CART
  -0.2977970   -0.0148610    0.0058630     12.0110
   0.0781570   -1.0376170    0.0300120     1.0080
   0.0624760    0.4671780   -0.9114460     1.0080
  -1.3941310   -0.0471860   -0.0190910     1.0080
   0.1975130    0.6205440    1.1806240     15.9990
  -0.1284430    1.5342610    1.1751150     1.0080
IC                           ! specify redundant ICs
    1    1    2    0    0    !  1    BOND    C1=H2
    1    1    3    0    0    !  2    BOND    C1=H3
    1    1    4    0    0    !  3    BOND    C1=H4
    1    1    5    0    0    !  4    BOND    C1=O
    1    5    6    0    0    !  5    BOND    O-H5
    2    5    1    2    0    !  6    ANGL    a1
    2    4    1    3    0    !  7    ANGL    b1
    2    5    1    3    0    !  8    ANGL    a2
    2    2    1    4    0    !  9    ANGL    b2
    2    5    1    4    0    !  10   ANGL    a3
    2    2    1    3    0    !  11   ANGL    b3
    2    1    5    6    0    !  12   ANGL    C-O-H
    3    6    1    3    5    !  13   H on plane(O-C-H) Wilson WAGGING
UMAT    0   1   0            ! u – mat
  1  1  1.
  2  2  1.
  3  3  1.
  4  4  1.
  5  5  1.
  6  6  1.
  6  8  1.
  6 10  1.
  6  7 -1.
  6  9 -1.
  6 11 -1.
  7  6  2.
  7  8 -1.
  7 10 -1.
  8  8  1.
  8 10 -1.
  9  7  2.
  9  9 -1.
  9 11 -1.
 10  9  1.
 10 11 -1.
 11 12  1.
 12 13  1.
-1
PED     0   15         ! PED
  1  sC1-H2
  2  sC1-H3
  3  sC1-H4
  4  sC1-O
  5  sO-H
  6  c1
  7  c2
  8  c3
  9  c4
  10 c5
  11 aOH
  12 dOH
-1
FMAT    1   1   0
 Force constants in Cartesian coordinates:         ! copied from Gaussian output
             1              2              3              4              5
  1  0.597071D+00
  2 -0.333027D-01  0.625440D+00
  3 -0.255709D-01 -0.402645D-01  0.536029D+00
  4 -0.945913D-01  0.100545D+00 -0.238576D-02  0.924274D-01
  5  0.942038D-01 -0.300248D+00  0.139197D-01 -0.101128D+00  0.326328D+00
  6 -0.604048D-02  0.242699D-01 -0.534958D-01  0.832155D-02 -0.672729D-02
  7 -0.925597D-01 -0.437944D-01  0.769922D-01  0.581839D-02  0.668752D-02
  8 -0.421050D-01 -0.118412D+00  0.102346D+00 -0.132890D-01 -0.122322D-01
  9  0.757396D-01  0.999028D-01 -0.222449D+00  0.123134D-02  0.884810D-03
 10 -0.317802D+00 -0.119947D-04  0.808707D-02 -0.100371D-01 -0.117105D-02
 11  0.778388D-03 -0.597213D-01  0.301534D-02  0.329113D-01  0.328409D-02
 12  0.107686D-01  0.141089D-02 -0.558964D-01 -0.233994D-02  0.100818D-02
 13 -0.948588D-01 -0.398285D-02 -0.513783D-01  0.648266D-02  0.306437D-02
 14 -0.297797D-01 -0.127739D+00 -0.848919D-01 -0.163327D-01 -0.175004D-01
 15 -0.662455D-01 -0.427781D-01 -0.197587D+00 -0.207369D-02 -0.622869D-02
 16  0.274055D-02 -0.194529D-01 -0.574431D-02 -0.100015D-03 -0.165675D-02
 17  0.102051D-01 -0.193201D-01  0.587498D-02 -0.270654D-02  0.368387D-03
 18  0.113487D-01 -0.425409D-01 -0.660015D-02 -0.275349D-02 -0.285673D-02
             6             7             8             9            10
  6  0.613236D-01
  7 -0.109854D-01  0.891400D-01
  8  0.260780D-01  0.486285D-01  0.116292D+00
  9  0.116622D-02 -0.781684D-01 -0.105323D+00  0.254150D+00
 10  0.174613D-02 -0.105062D-01  0.181327D-02  0.794390D-03  0.344096D+00
 11 -0.115982D-02 -0.177145D-01  0.183845D-02  0.201813D-02  0.563951D-02
 12  0.150525D-02  0.284167D-01 -0.190962D-03  0.210589D-02  0.109360D-02
 13  0.910631D-02  0.838922D-02  0.416120D-02  0.927370D-03 -0.605391D-02
 14 -0.378915D-01  0.536280D-02  0.105568D-01  0.645863D-03 -0.695697D-02
 15 -0.498922D-02 -0.173043D-01 -0.225451D-01 -0.355247D-01 -0.113758D-01
 16 -0.214813D-02 -0.281699D-03  0.790947D-03 -0.524286D-03  0.302752D-03
 17 -0.456934D-02  0.830065D-03  0.195694D-02  0.187143D-02  0.687234D-03
 18 -0.551005D-02  0.104920D-02 -0.365235D-03  0.552160D-03 -0.345383D-03
            11            12            13            14            15
 11  0.541389D-01
 12  0.415805D-02  0.613457D-01
 13 -0.200406D-01 -0.378410D-01  0.159040D+00
 14 -0.538163D-03 -0.862191D-02 -0.128513D+00  0.583290D+00
 15 -0.874936D-02 -0.998586D-02  0.108715D+00  0.992416D-01  0.281738D+00
 16 -0.157409D-02 -0.979160D-04 -0.729992D-01  0.176219D+00 -0.117161D-01
 17  0.998111D-03  0.223575D-02  0.145310D+00 -0.448069D+00 -0.189403D-01
 18  0.717664D-03  0.925407D-03 -0.295297D-01  0.315178D-01 -0.336506D-01
            16            17            18
 16  0.703376D-01
 17 -0.154326D+00  0.464065D+00
 18  0.202307D-01  0.135275D-01  0.442832D-01
SCALE       ! Scale factor for Gaussian MP2 calculation, scaled to experimental.
  0.89
END

STOP
```

The corresponding output:

```
 REDUCED POTENTIAL ENERGY DISTRIBUTION MATRIX [%]

  1      331.0    12 100.
  2     1021.3     4  51.      7  27.     11  21.
  3     1049.2     4  50.      7  35.
  4     1135.3     8  96.
  5     1337.1     7  32.     11  64.
  6     1451.5     6  98.
  7     1476.2    10  96.
  8     1488.6     9  89.
  9     2901.4     2  45.      3  45.
 10     2964.7     2  50.      3  50.
 11     3039.3     1  90.
 12     3579.8     5 100.

   Symbolic PED matrix [%] (sorted)

  1     331.0   dOH      100.
  2    1021.3   sC1-O     51.   c2         27.   aOH        21.
  3    1049.2   sC1-O     50.   c2         35.
  4    1135.3   c3        96.
  5    1337.1   aOH       64.   c2         32.
  6    1451.5   c1        98.
  7    1476.2   c5        96.
  8    1488.6   c4        89.
  9    2901.4   sC1-H4    45.   sC1-H3     45.
 10    2964.7   sC1-H3    50.   sC1-H4     50.
 11    3039.3   sC1-H2    90.
 12    3579.8   sO-H     100.
GFX option finished
```

0.5.3. MOLVIB scripting with CHARMM data

The following script illustrates the utilization of MOLVIB module basing on CHARMM MM calculation. This script would calculation the vibrational spectrum of a CH_3OH molecule.

Note that force constant is calculated by CHARMM (SECOnd specification in MOLVIB section) so that FMAT key word in MOLVIB must be absent.

To use this script for practice, the topology and parameter files for CH_3OH as well as CHARMM general force field must be present.

CHARMM input script

```
* PROJ Force Field Dev - small drug molecules
* Molecular vibration frequency calculation CH3OH
* Based on Gaussian MP2/6-31Gd vibrational modes
* Zilin Song, 13 Jan 2019
*

BOMLev -1
FASTer 1

SET optff ./ch4o.strm   ! toppar for CH3OH
SET residue CH4O

! { read force field Cgenff force fields to support generated force field }

OPEN UNIT 10 READ CARD NAME @topparDir/top_all36_cgenff.rtf
  READ rtf CARD UNIT 10
  CLOSe UNIT 10

BOMLev -2   ! required for passing NBFIX error.
            ! NBFIX error pops up when:
            ! No protein force field read before cgenff.
OPEN UNIT 10 READ CARD NAME @topparDir/par_all36_cgenff.prm
  READ param CARD FLEX UNIT 10
  CLOSe UNIT 10
BOMLev 0
STREam @optff           ! read optimized force field

! { target molecule psf & crd }

READ SEQUence CARD
* @residue
*
1
@residue
```

```
GENErate @residue FIRSt none LAST none WARN SETUp

OPEN UNIT 20 READ FORM NAME @crdname
  READ coor CARD UNIT 20 APPEnd
  CLOSE UNIT 20

! { do minimization & save minimized crd }
    MINI CONJ NSTEp 200 NPRINt 20 INBFrq 1000 CUTNb 999
    MINI NRAP NSTEp 50 TOLGrd 0.00001

! {  CHARMM MOLVIB Module is used for molecular vibrational calculation  }

MOLVIB -
  NDI1 12 -       ! number of vibrational degrees of freedom
  NDI2 12 -       ! number of ICs left after first UMAT transformation
  NDI3 13 -       ! number of primitive ICs
  SECOnd          ! calculate force constants and pass to MOLVIB
GFX               ! vibrational analysis in Cartesian coordinates.
PRNT 0                  ! print level 5 -> for testing
IC                      ! specify redundant ICs
    1    1    2    0    0    !  1    BOND    C1=H2
    1    1    3    0    0    !  2    BOND    C1=H3
    1    1    4    0    0    !  3    BOND    C1=H4
    1    1    5    0    0    !  4    BOND    C1=O
    1    5    6    0    0    !  5    BOND    O-H5
    2    5    1    2    0    !  6    a1
    2    4    1    3    0    !  7    b1
    2    5    1    3    0    !  8    a2
    2    2    1    4    0    !  9    b2
    2    5    1    4    0    !  10   a3
    2    2    1    3    0    !  11   b3
    2    1    5    6    0    !  12   C-O-H
    3    6    1    3    5    !  13   H - plane(O-C-H) WAGGING
UMAT    0   1   0       ! u – mat
  1  1  1.
  2  2  1.
  3  3  1.
  4  4  1.
  5  5  1.
  6  6  1.
  6  8  1.
  6 10  1.
  6  7 -1.
  6  9 -1.
  6 11 -1.
  7  6  2.
  7  8 -1.
  7 10 -1.
  8  8  1.
  8 10 -1.
  9  7  2.
  9  9 -1.
  9 11 -1.
 10  9  1.
 10 11 -1.
 11 12  1.
 12 13  1.
-1
PED      0   1    ! PED
  1  sC1-H2
  2  sC1-H3
  3  sC1-H4
  4  sC1-O
  5  sO-H
  6  c1
  7  c2
  8  c3
  9  c4
  10 c5
  11 aOH
  12 dOH
-1
END

STOP
```

0.6. Python Subprocess Module.

The *subprocess* module in Python 2.4 and later versions intends to replace following old packages:

os.sys;
os.spawn;*
os.popen;*
popen2;*
commands..*

It provides higher level of subprocess managements in python programs. Likely to Linux process, in Python programs, a subprocess could be forked by the main process via the *subprocess* module and to execute an external program.

In addition to the several methods/functions to create subprocess in different manner, this module also provides tools for standard stream and pipe management, so as to achieve inter-process communication (text).

Although the complete functionality of *subprocess* module is not one of the topics of this tutorial, there is one method/function from this module is important in the following sections:

subprocess.call()

The main process would wait until the subprocess been called is completed, and return the return-code, which is equivalent to the Linux exit code.

In practice of this tutorial, this method/function is applied in the following manner, note that the return-code is not critical of this method/function:

import subprocess as subp
subp.call('linux-shell-command-args', shell=True)

the *shell* argument in *subprocess.call()* method/function is *False* by default.

Under the Linux context, when *shell=False*, *Popen* would call *os.execvp()* to execute the programs that the args refers to. Meanwhile, if *shell=True* and the args is a string, *Popen* would directly call the Shell from the system to execute the shell command.

In most cases, *shell=True* is suggested for comprehending the scripts provided in this tutorial, and this *subprocess* module is mandatory for automation and serialization of the parametrization routine.

0.7 Principles of Decrypting Gaussian Output Logs

Decrypting a Gaussian output by Python scripting is not mandatory for the general purpose of the parametrization of the force field. But such skill would surprisingly increase our efficiency, as the output log file produced by Gaussian is extremely redundant and full of garbage, even trash talks.

There are several ways of coding (not necessarily Python codes) to extract useful information from Gaussian log files, some typical examples using Python 3 are listed below, note that each example listed would behave very differently in some special situations:

EXAMPLE I:

```
    with open(str(file_name), ‘r’) as file_in:
        lines = file_in.readlines()
        for l in lines:
            words = l.spilit
            …       # here applies
            …       # operations
            …       # for each line
```

EXAMPLE I:

```
    file_in = open(str(file_name), ‘r’)
    line = file_in.readline()
    while line:
        …       # here applies
        …       # operations
        …       # for each line
        line = file_in.readline()
```

0.7.0a appendix: the usage of *open()* method/function in Python 3

The *open()* method/function in Python 3 offers the support of operating/interacting with file management system in Python programs, regardless to the operating system (Windows, Linux or MacOS). The *open()* method would return a *TextIOWrapper* object which could provide access for investigating and operating the corresponding file. The syntax and technical details of this method/function is not a main theme of this tutorial but several applications of this method are listed below for the reader’s reference.

Open a file using read-only mode, no permission to change the content:

```
fi = open(‘directory-of-the-file’, ’r’)
# operations on fi.
```

Append content to the end of an existing file, if not exist, create this file:

```
fi = open(‘directory-of-the-file’, ’a’)
```

```
# operations on fi.
```

Open a file using read & write mode, if file exists, wipe all the content and write; if not exist, create the corresponding file and start writing:

```
fi = open('directory-of-the-file', 'w')
# operations on fi.
```

Other operators of this method would provide different behaviors on manipulation a specific file, and the readers should rely on their own capability to find the best practice for their specific scenario and context.

1. Principles

1.0. Understanding CHARMM force field.

Although connectivity is important

Specification of bond orders or types is not mandatory in CHARMM force field files but useful for distinguishing or assigning atom types.

1.0.0. Format of CHARMM force field

1.1. The CHARMM General force field (CGenFF) and Parametrization

1.1.0. Objectives for Parametrization

1.1.1. Atomic Charge

1.1.2. Equilibrium Bond length and Angles

1.1.3. Force Constant of Bond Stretching and Bond Angles

1.1.4. Force Constant of Dihedral Angles

1.1.5. Force Constant of Improper Angles

2. Practice

2.0. Guidelines

Although, section 1 described the principles of parametrization basing on CGenFF as well as its principle. The parametrization is quite plain forward but on the other hand, manually searching for parameters under the guidance of section 1 would be extremely complicate and fallible, which means, such process could be extremely time-consuming. Since the process of parametrization is mostly redundant operations searching for the ideal parameters, it is natural to come up with the idea to use serial scripting methods to repeat the parameter testing process and filter for ideal set of parameters, which is also the main topic of the current section.

Before starting the current section, the readers are assumed to have fruitful experience for CHARMM scripting and python programming, and if necessary, basic knowledges for operating under Linux environment.

Although section 2 contains numerous technical details remains unexplained (in a very detailed manner), the authors would assume that, (without spending too much effort,) our readers are capable of investigating and implementing these principles and practices illustrated by pseudo-codes or partial/incomplete scripts throughout this section.

The authors ensure that all principles or practice in this section had been ***actually*** implemented and had been applied for production run.

2.1. Parametrization of Atomic Charges

2.2. Parametrization of Equilibrium Bond Length & Angles

2.3. Parametrization of Force Constant on Bond Length & Angles

2.4. Parametrization of Force Constant on Dihedral Angles/Improper Angles

3. Supplemental Information