_ A guide to Adaptive Force Matching (AFM) using the tools developed in the Wang Group

Ying Yuan, Ryan Rogers, Feng Wang

Contents

1. Introduction: 2

2. Obtain the utilities. 2

3. Extracting QM/MM configurations for the sampling step 2

3.1 Introduction 2

3.2 pxyz file 2

3.3 Utility scripts developed for extracting QM/MM configurations 3

4. Creation of input files for the electronic structure calculations in the QM/MM step of AFM. 7

4.1 Introduction 7

4.2 Scripts that generate or update input files for the electronic structure code 7

5. Generation of input files for the CRYOFF code that performs the fitting. 12

5.1 Introduction 12

5.2 Scripts developed to generate the ref file 13

5.3 Scripts to extract parameters from the *.off* 16

5.4 off2ff: update ff file from a off file. 19

5.5 Scripts being used by off2ff 23

6. Scripts developed to update/generate topology or tabulated potential files for Gromacs. 24

6.1 Introduction 24

6.2 Update the Gromacs topology and generate tabulated potential files 24

6.3 Scripts being used by off2top and off2tab 29

7. Known Limitations: 30

8. Language convention and terminology used in the manual 30

9. Usage example 31

9.1 Generate the QM/MM region 31

9.2 Generate QM/MM calculation input files 33

9.3 Generate .ref file for CRYOFF 34

1. Introduction:

This guide describes a set of tools that can be used to conduct force field developed following the adaptive force matching (AFM) method as described in the following key publications.

1. Akin-Ojo, O.; Song, Y.; Wang, F., Developing ab initio quality force fields from condensed phase quantum-mechanics/molecular-mechanics calculations through the adaptive force matching method. J. Chem. Phys. 129, 064108,(2008)
2. Akin-Ojo, O.; Wang, F., The quest for the best nonpolarizable water model from the adaptive force matching method. J. Comput. Chem. 32, 453-462 (2011)
3. Wang, F.; Akin-Ojo, O.; Pinnick, E.; Song, Y., Approaching post-Hartree-Fock quality potential energy surfaces with simple pair-wise expressions: parameterising point-charge-based force fields for liquid water using the adaptive force matching method. Molecular Simulation 37, 591-605 (2011).
4. Li, J. C.; Wang, F., Pairwise-additive force fields for selected aqueous monovalent ions from adaptive force matching. J. Chem. Phys. 143,194505,(2015)

A typical execution of AFM iterates through the three steps: the sampling step, the QM/MM step and the force matching step. A set of utilities have been developed using perl to facilitate the development of force fields following the three steps.

2. Obtain the utilities.

The utilities can be downloaded from
https://wanglab.uark.edu/CRYOFF/AFM_scripts_vx.tgz
To use them, execute (note replace vx with the actual version number)
tar -xvzf AFM_scripts_vx.tgz
and put the path containing the scripts in the system PATH.

You can edit the setenv.script file that is distributed as part of the utilities to help you set up the system PATH.
If you choose to edit the setenv.script file, use it by typing
source setenv.script

3. Extracting QM/MM configurations for the sampling step.

3.1 Introduction

AFM start from the sampling step. The sampling step typically contains several MD simulations at various conditions the force field is expected to model. The current script supports extracting QM/MM information when the MD simulations were done with the Gromacs suite of program. (http://www.gromacs.org/)

3.2 pxyz file

We used an intermediate pxyz (periodic xyz) file format to facilitate the support of other MD or MC programs in the future.

The pxyz file has the following format.

Number of atoms
periodic boundary condition
atomname xxx yyy zzz molname mark
…….
Example

```
1536
43.1   43.1   43.1
C   23.01    2.2     8.57   1CO2   -9
O   22.19    1.56    8.05   1CO2   -9
O   23.85    2.88    9.07   1CO2   -9
C   39.9    24.84   13.52   2CO2   -9
O   38.86   24.96   14.02   2CO2   -9
O   41.00   24.83   13.14   2CO2   -9
C   38.67   12.75   21.05   3CO2   -9
O   39.28   13.54   20.41   3CO2   -9
O   38.08   11.94   21.65   3CO2   -9
…….
```

The number of atoms is a free format integer.
The periodic boundary condition line contains the box vectors that follows the format Gromacs used for specifying box vectors in the gro file. It is space separated real values in the sequence:
v1(x) v2(y) v3(z) v1(y) v1(z) v2(x) v2(z) v3(x) v3(y)
We note that currently the CRYOFF code and toolset scripts only implemented support for orthorhombic boxes with v1(y)=v1(z)=v2(x)=v2(z)=v3(x)=v3(y)=0. Thus the last 6 numbers are typically omitted from the input file.

The atom specification line is space delimited similar to that in a standard xyz file.
Two additional fields are required for each line, molname, is the same of the molecule and should be different for different molecules. It could be for example 1H2O, 2H2O.
The sixth field is a mark that is updated by the scripts. Typically, a script will only increase the mark value with only few exceptions. The typical convention is to have QM atoms being fit to have a higher mark than QM atoms not being fit, the MM atoms has the lowest mark.

3.3 Utility scripts developed for extracting QM/MM configurations

A general rule of the mark scripts is that the script will only mark up and will not mark down. In other words, if a script is supposed to mark a certain atom to a value n. If the original mark is already greater than n, the script will not reduce the original mark.

Unless otherwise specified, all scripts below assume there is only one frame of configuration in the input file.

-h (-help) option is supported for these scripts. With this option, the usage will be printed.

An argument in [] is optional. When missing the file will be taken from the standard input. All counting of atoms or molecules start from one. The distance unit is Angstrom.

gro2pxyz [grofile]

gro2pxyz reads a grofile as argument or as standard input and write a pxyz as standard output. The default mark of -9 will be used when creating the pxyz file.

pxyz_2gro [pxyzfile]

pxyz_2gro reads a pxyz file and converts it to a gro file. The box vector in pxyz will be used for the PBC information for the gro file.

mark_byname targetname val [pxyzfile]

markbyname marks up all the atoms where the molecule name contains the string targetname. The molecule will be marked up to val (the second argument).

The script takes the pxyzfile either as an argument or through standard input.

For example,

markbyname solute 2 test.pxyz,

will mark up any molecule with solute as part of its name to a mark of 2.

mark_within ido rcu val [pxyzfile]

markwithin mark up any molecule with a distance of *rcu* Å of atom *ido* to a value *val*.

The script will only cause the mark value to increase and will not decrease the mark value even if it is within rcu of atom ido.

The script takes the pxyzfile either as an argument or standard input.

mark_within_range ido ide rcut val pxyzfile

This script will mark up any molecules within a distance of *rcut* Å of any atom within the range of atomic indices *ido* to *ide*. It calls markwithin for each of the atom within the range. pxyzfile is taken as an argument and it will be overwritten with the molecules marked.

mark_within_list rcut val pxyzfile atomno1 [atomno2 …]

This script (markup within list of atoms) will mark up any atom within rcut Å of the list of atoms, atomno1, atomno2 etc.

pxyzfile is the name of the pxyz file, which should be part of the argument list and will be overwritten after execution.

mark_boundary rcu valm val pxyzfile

This script will mark up any molecule within a distance *rcu* Å of the atoms whose mark equals to *valm*. The atoms matched will be marked up to val.

The script takes the *pxyzfile* as an argument and uses the *markwithin* script to do the marking. pxyzfile will be overwritten after execution.

mark_nextnearest ido ide val [pxyzfile]

This script mark up the nearest molecule of a reference molecule that is defined by all the atoms in the range of *ido ide* to a value of *val*. The nearest molecule to the center molecule will be marked up to *val.* The script takes the *pxyzfile* either as an argument or standard input.

mark_nextnearest_n ido ide nmark val pxyzfile

This script will mark up *nmark* nearest molecules of the atom range *[ido ide]* to a value *val.* It calls mark_nextnearest N(=nmark) times. The script takes the *pxyzfile* as an argument.

markup_random valm val [pxyzfile]

This script will pick a random molecule with a mark of *valm* and mark it up to *val*.

markup_mol rcut valm val [pxyzfile]

This script will mark up a molecule with a mark of "*valm*" to "*val*" if there are no molecule with a mark lower than "*valm*" within the cutoff *rcut*.

val should be larger than *valm*.

pxyz_dropoff val [pxyzfile]

dropoff will write to standard output a new pxyfile where all atoms with a mark no larger than *val* will be removed. The script takes the *pxyzfile* either as an argument or standard input. It will properly update the atom number in the first line of final pxyz file.

pxyz_2vxyz [pxyzfile]

pxyz2vxyz creates a new xyz file with the atomic names modified to include the mark as part of the name. The script will allow easier preparation of input files for QM/MM and also allow easy visualization with vmd. (vxyz stands for vmd xyz.)

The script takes the *pxyzfile* either as an argument or standard input.

pxyz_select valm [pxyzfile]

pxyz_select prints all the atoms with *valm*. The output will be a headless pxyz file.

In addition, the script counts the number of atoms and molecules marked with the value *valm*, and print the count to standard error. The script takes the *pxyzfile* either as an argument or standard input.

pxyz_select_n pxyzfile valm1 [valm2] …

pxyz_select_n prints all the atoms with a list of mark values valm1, valm2,… The output will be a headless pxyz file.

In addition, the script counts the number of atoms and molecules marked with the value valm1, valm2, … and print the count to standard error. The script takes the *pxyzfile* either as an argument.

pxyz_sort [pxyzfile]

pxyz_sort write a new pxyz file to the standard output with the atoms sorted according to the mark value from high to low. The script takes the *pxyzfile* either as an argument or standard input.

xyz_fix_lineno [xyz file]

xyz_fix_lineno will read in an xyz file from standard input or as an argument, recount the number of atoms and fix the atom number line of the xyz file. This is used to fix the broken atom number after processing an xyz or pxyz file.

xyz_add_lineno [xyz file]

xyz_add_lineno will read in a headless xyz file either from standard input or as an argument and add in the lineno and a comment line to convert the headless xyz into a standard xyz file, which is written to the standard output

chunk nskip nlines [filename]

chunk will first skip the first *nskip* lines and then print out *nlines*. When *nlines* is larger than the total number of lines remaining in the file, only the actual remaining lines will be printed.

4. Creation of input files for the electronic structure calculations in the QM/MM step of AFM.

4.1 Introduction

After we get the QM/MM specifications in the pxyz file, input files for QM/MM calculations have to be created. Each electronic structure code will have to follow a slightly different procedure. So far we support Molpro, GAMESS, PQS, and Gaussian.

Most scripts described below are designed to only update the geometry information of an existing input file that has been fully tested to perform the proper QM/MM calculations using the electronic structure program of choice.

-h (-help) option is supported for these scripts. With this option, the usage will be printed.

4.2 Scripts that generate or update input files for the electronic structure code.

Molpro

pxyz_molpro_upd_geom templ nametrans [pxyzfile]

pxyz_molpro_upd_geom updates the geometry section of the Molpro input file "templ". It only supports GEOMTYP=XYZ in molpro.

With Molpro, the atom name can only be element name followed by numbers. The translation file (*nametrans*) translates the atom names in the pxyzfile to the atom type needed for Molpro. The *nametrans* file is a text file with two columns. The atom name in the pxyzfile is the first column, the atom name to be used for the electronic code (in this case Molpro) will be the second column.

If the name used in the pxyzfile is compatible with Molpro, no translation will be needed. In this case, one can simply use /dev/null as the translation file.

e.g.
H H
HB1 H

HB2 H
HB3 H
OC1 O1
OC2 O2
OW O

The updated Molpro input will be printed to standard out. When updating the QM geometry information, the atom name will be created according to the nametrans file. The name in pxyz file will be used if no translation is founded. We note that the script will work with a standard *xyzfile* as input.

pxyz_molpro_gen_lattice chginfo [pxyzfile]

pxyz_molpro_gen_lattice generates the point charge lattice for Molpro to model the MM region through coulombic embedding.

The chginfo file provides the information on the partial charges for each atom type in the MM region. This is a text file with two or three column, with the atom name in the pxyz file in column 1, partial charges in column 2. The optional column 3 provides the flag value. Molpro uses the flag value to decide whether gradient on the partial charge should be calculated. If the third column is missing, the flag value will be 0.

e.g.

HW1 0.6645 0
HW2 0.6645 0
MW -1.3290 0

molpro_replace_field keyword newkey [molpro_input]

molpro_replace_field update a comma separated field in the Molpro input file.

If any field in any comma separated input line contains the “keyword” the “keyword” will be replaced by “newkey”. The newkey argument cannot have space in it.

One use of this script is to update the file name of the lattice file.

For example, molpro_upd_inp ala_mm ala_mm_22.inp molpro_input.inp

will update the field with ala_mm as part of the lattice file name to ala_mm_22.inp.

GAMESS

pxyz_gms_upd_geom templ znucinfo [pxyzfile]

pxyz_gms_upd_geom updates the QM geometry of the GAMESS input file, which is the atom input in the $DATA group. The script only supports COORD=CART and only updates the atom geometry information in the templ file with the new geometry in the pxyzfile. All the atoms in the pxyzfile will be treated as QM atoms. Thus you may need pipe the pxyzfile through the dropoff script first.

Since GAMESS input identifies atoms through nuclear charges, this information has to be provided by the translation file znucinfo. Znuinfo is a text file (free format) with two to three columns. The atom name in pxyzfile is in column 1 and the nuclear charge should be in column 2. The atom name to use for GAMESS is a combination of a basename and the mark value in the pxyzfile. If a third column exists in the nucinfo file, it will be used as the basename. Otherwise, the atom name in the pxyz file will be used as the basename.

Example nucinfo file:

```
H       1
HW1     1    H
HW2     1    H
C       6
CA      6    C
CB      6    C
N       7
```

The updated GAMESS input will be printed to standard out.

We note that the script will work with a standard xyzfile as input. Since the mark value does not exist in a standard xyz file, if a xyzfile is used as input, the mark value will simply be ignored.

pxyz_gms_upd_frag template chginfo [pxyzfile]

pxyz_gmx_upd_frag updates the fragment section ($EFRAG and $FRAGNAME) to provide location of partial charges. GAMESS limit the size of each fragment. This script will only work if all charges fit within one fragment. Only COORD=CART is supported.

The translation file (chginfo) is a text file with two columns, with the atom name in pxyzfile in column 1 partial charge in column 2.

e.g.

```
HW1   0.6645
HW2   0.6645
MW   -1.3290
```

The updated GAMESS input will be printed to standard out. The name of fragment points has to be unique for each fragment thus it is created by appending the atom name in the pxyz file by the 'M' character and a count from 0.

PQS

pxyz_pqs_gen_xyz nametrans [pxyzfile]

PQS can read in the geometry through a .xyz input file. (We note the pqs use an xyz file without the header)

pxyz_pqs_gen_xyz generates the xyz input file for use with PQS.

The translation file (nametrans) translates the atom names in the pxyzfile to the atom specification needed for the QM calculation. The nametrans file is a text file with two columns. The atom name in the pxyzfile is the first column. The atom specification in the pqs xyz file will be replaced by the second column (if exists).

If different basis set were to be used for an element, PQS requires a special character to be appended to the atom name, which is the first one or two letter of atom specification . PQS allows the use of the following special characters: !@#$%^&*+=<>? .

If no translation is needed, one can simply use /dev/null as the translation file.

A standard xyz file can be used as the pxyzfile for this script.

e.g.

```
H4          H@
HA4         H@
C4          C@
CA4         C@
N4          N@
O4          O@
OW3         O&
HW13        H&
HW23        H&
OW2         O
HW12
HW22
```

pxyz_pqs_gen_pntq chginfo [pxyzfile]

PQS has multiple ways to specify partial charges. One way (recommended by Dr. Peter Pulay) is to use the pntq file.

pxyz_pqs_gen_pntq generates the pntq for PQS.

The translation file (chginfo) is a text file with two or three columns, with the atom names in pxyzfile in column 1, partial charges in column 2. The default name for point charge in the pntq is "Q". It will be replaced by the string in column 3 if column 3 exists.

e.g.

HW1 0.6645
HW2 0.6645
MW -1.3290

pqs_replace_field keyword newkey [template]

pqs_ *replace_field* will update a keyword in the pqs template file. The updated file will be printed to standard out. One use of this script is to update the geometry xyz file name. It assumes the fields in each line is delimited by space. The field contains "keyword" will be replaced by "newkey".

Gaussian

pxyz_gaussian_upd_geom templ nametrans [pxyzfile]

pxyz_gaussian_upd_geom updates the geometry in the molecule specification section of a template gaussian input file "templ". In the template input file, the molecular geometry after the charge and spin line should be replaced with the word GEOMETRY. The scripts will replace everything from the word "GEOMETRY" to the next blank line with the atom positions in the pxyz file.

The translation file (nametrans) translates the atom names in the pxyz file to the atom type needed for the QM calculation. This file has the same format as the translation file for Molpro.

pxyz_gaussian_upd_lattice templ chginfo [pxyzfile]

pxyz_gaussian_upd_lattice updates the point charges for coulombic embedding in the Gaussian input file. The Background charge distribution section in the template Gaussian input file should replaced with keyword "charge_lattice". The script will replace everything from the keyword "charge_lattice" to the next blank line with the point charges based on the coordinates in the pxyzfile.

The charges are provided with the translation file (chginfo). It is a text file with two columns, with the atom name in pxyzfile in column 1 partial charge in column 2.

5. Generation of input files for the CRYOFF code that performs the fitting.

The manual for CRYOFF can be found separately. As of writing of this manual, the link to the CRYOFF manual is https://wanglab.hosted.uark.edu/cryoff/wanglab_CRYOFF.html

5.1 Introduction

After the QM/MM calculation is performed, it has to be converted to the ref files needed for the CRYOFF code.

The CRYOFF codes needs a ff file for defining the fitting protocol and a ref file for providing the reference forces.

A ref file may contain any number of frames. For the specification of each frame, it has the following format:

N_atoms

Comment [box=(xx, xy, xz; yz, yy, yz; zx, zy, zz)]

atm1 x1 y1 z1 Fx1 Fy1 Fz1 QM_wt.1 Mol.name1

atm2 x2 y2 z2 Fx2 Fy2 Fz2 QM_wt.2 Mol.name1

……

NetF xc1 yc1 zc1 FNx1 FNy1 FNz1 QM_wt.N1 Mol.name1

Torq xc1 yc1 zc1 Fτx1 Fτy1 Fτz1 QM_wt.τ1 Mol.name1

……

N_atoms (for the next frame.)

An example of a part of the ref file is shown below:

```
      1800
Step          1
   O  -0.0270470000   2.6880000000   0.5228100000     -31.2141485233      -0.6221558195     -22.8714144624  1.000000    1Wqm
   H  -0.8848800000   2.2779000000   0.1968500000      27.7629057616       5.3757169111       5.3300668164  1.000000    1Wqm
   H  -0.2191200000   2.8333000000   1.4631000000       7.3807359919      -4.3372370081       8.0188812948  1.000000    1Wqm
   E  -0.2370300000   2.6350000000   0.6456700000       0.0000000000       0.0000000000       0.0000000000  1.000000    1Wqm
NetF  -0.3770156667   2.5997333333   0.7275866667       3.9294932303       0.4163240836      -9.5224663512  1.000000    1Wqm
Torq  -0.3770156667   2.5997333333   0.7275866667       4.0545494162       6.5308496050       6.3336025874  1.000000    1Wqm
   O  -0.3509600000   3.1092000000   7.5530000000      -6.2533459568     -27.8154870225      -1.4021749245  1.000000    2Wqm
   H  -1.0391000000   2.5583000000   7.0630000000      17.2088906736      20.0755254162      14.1087473314  1.000000    2Wqm
   H  -0.3443500000   4.0374000000   7.2613000000      -3.9115678004       1.2803537032      -9.5478009587  1.000000    2Wqm
   E  -0.4872700000   3.1846000000   7.3967000000       0.0000000000       0.0000000000       0.0000000000  1.000000    2Wqm
NetF  -0.5781366667   3.2349666667   7.2924333333       7.0439769164      -6.4596079031       3.1587714481  1.000000    2Wqm
Torq  -0.5781366667   3.2349666667   7.2924333333      -5.1384007821       3.5983786671      -1.2767885121  1.000000    2Wqm
```

N_atms in the first line specify the number of sites in this frame. NetF and Torq are also considered to be sites in this context.

The second line is a comment line. If the PBC is used when fitting, the box information should also be included in this line. This line could also include some other keywords. (see CRYOFF manual)

The following are the site information. There are 9 fields for each line.

The first 4 fields are the same as the 4 fields in a standard xyz file. (It can actually be visualized in vmd)

The next 3 fields are the reference forces.

The 8th field is the solvation factor, which is a weight for each force component. A zero weight will cause the line not to be fitted.

The last field is the molecule name.

For each molecule, another two components NetF and Torq may be included to provide total force and total torque. The coordinate for the total torque is the center for the torque calculation for that molecule.

The unit for length is Angstrom (Å) and for force is kcal/(mol·Å)

For more detail description, please refer the CRYOFF manual.

5.2 Scripts developed to generate the ref file

ref_gen_step1_cord molinfo [xyzfile]

This typically is the first step for generating a ref file.

ref_gen_step1_cord writes to standard output a .ref file with only atomic position information. It will also calculate the center needed for the torque calculations, although the same center will be written as the position of NetF, it does not affect the fit as net force is insensitive to the center. The weight used to compute the center is provided through the molinfo file.

comment line
number name solv
atomname weight
atomname weight
……
[next triggeratom]
[number name solv]
[atomname weight]
……
next

Example:
CO2 force matching
3 CO2 1.0

```
C       1.0
O       1.0
O       1.0
next    O
3    H2O    1.0
O       1.0
H       1.0
H       1.0
next
```

The comment line will be used as the comment line in .ref file.

The sequence of molecule types must be the same in “xyzfile”, however, there could be any number (greater than 1) molecules of each type, the script assume the same molecule type and keep reading until the trigger atom specified by the “next” line is encountered.

For each molecular type specification
“number” is the number of atoms in the molecule. “name” is the name of the molecule. “solv” is the solvation factor of this type of molecule.
The following lines are the name of the atoms (atomname) to be used for the ref file. It does not have to be the same as those used for the pxyz file.
The atomname is followed by the weight, which is the weight used for the center calculation for NetF and Torq. If the weights of every atom of a molecule is zero, the NetF and Torq will not be printed.

The “next” set the trigger atom to inform the script a new type of molecule is encountered. The triggeratom has to be the first atom of the new molecule in the xyz or pxyz file.

A limitation of the script is that the first atom names of two adjacent molecule types (triggeratom) in the xyz file cannot be the same.

The final next after the specification of the last molecule type is required.

The input *xyzfile* has the same format of a standard xyz file except that it does not have the first two lines. (headless) Molecule sequence in the xyzfile must the same as those in the molinfo file.
If a mistake is made, it is very likely the script never finds the correct trigger atom and generates an invalid ref file without warning.

Update the forces in ref file

ref_upd_d3_grad d3_grad [.ref]
ref_upd_dlploy_frc dlpoly_revcon [.ref]
ref_upd_gmx_frc gmx_force [.ref]
refu_upd_pqs_frc pqs_force [.ref]
ref_upd_gaussian_frc gaussian_frc [.ref]
ref_upd_gms_grad gms_grad [.ref]
ref_upd_molpro_grad molpro_grad [.ref]

The ref file output produced by *refgen_step1_cord* does not have any force information. The force information needed for CRYOFF can be provided with the *ref_upd* scripts shown above.

Each script requires an input file with gradient or atomic forces. Reads an input ref file either as an argument or through standard input and write a new ref file to standard output.

Each script read in the atomic force values from the input ref file, and add the value to be incorporated (updated) to the current values.

There d3 means output of Grimme's dftd3 code. dlpoly stands for DLPOLY, gmx standard from gromacs, pqs stands for PQS, gau stands for Gaussian, gms stands for GAMESS, molpro stands for Molpro.

grad in the file name indicates that it expects to read the energy gradient information.

frc in the file name indicates it expects to read the force information, which is negative of the gradient.

This is chosen to be consistent with the default output specific for each electronic structure code.

The file specified as the first argument, which contains either force or gradient information, can be extracted from the output file of the electronic structure program with the following scripts.

dftd3*:* use *getd3frc func xyzfile* .
Gromacs: follow the steps in get_gmx_frc
PQS*:* PQS write a .grad file, which is compatible. Alternatively, you can use *pqs_extrac_frc pqs_output*
DLPOLY*:* The REVCON file is used as the force file.
Gaussian*: gaussian_extract_frc Gaussian_output*
GAMESS*: gms_extract_grad GAMESS_output*
Molpro*: molpro_extract_grad molpro_output*

These scripts write the force (frc) or gradient (grad) information extracted from the respective output files to standard out. This extracted information is compatible with the ref_upd_ scripts above.

getd3force func xyzfile
getd3force will call dftd3 code to calculate D3 dispersion forces. We note that the script will only run if the dftd3 executable is in the PATH. The energies and forces will be written into the files with the same name as xyzfile with the extensions of .eng and .d3.grad.
The func is the functional type supported by dftd3. The damping method is BJ, which is set in the script and can be easily updated.

get_gmx_frc grofile templatedir
get_gmx_frc does a single point calculation for a frame in grofile and then extract the forces of this frame. The script needs mdrun_d in the PATH. The calculation will be performed in the templatedir directory. This directory should include a top file named "template.top", a mdp file named "mdrun.mdp". If table files are needed, they should also be provided and with "mdrunoptions" variable properly set in the script.

A file “input_ndx” is needed to make the index file for this frame. This file is the input to the make_ndx command of gromacs.
The name for force file will have an extention .gmxfrc.

The script calls the " top_upd_mol_nu" script to update the number of water (SOL) in the top file automatically based on the grofile.
The script may need to be tailored for different systems.

ref_upd_net [.ref]

ref_upd_net calculates the molecular forces and torques and updates the NetF and Torq field of the ref file. The center of reference is based on the x, y, z coordinate of the Torq entry. The coordinate for the NetF entry is ignored. The input .ref can be taken as an argument or from the standard input. The output is written to standard out.

The script ignores existing values of the force and torque current in the input .ref file.

xyz_add_msite atm1 atm2 atm3 M [xyzfile]

addmsite will add the position of dummy site for BLYPSP-4F water. This is the geometry shared for all water models developed to date in the Wang group. (WAIL, B3LYPSP-4F) This is needed when the QM reference calculations do not have the M site. And the M site is needed for fitting.

atom1, atom2, atom3 are the names of the three atoms used to specify the water. The oxygen must be the first atom. The fourth argument is the name of the virtual site.

“xyzfile” is the standard xyz file. Since the pxyzfile and .ref file all formatted similar to an xyz file. The script also properly add M site locations to pxyzfile. When M site information is to be added to an .ref file , the ref_fix_msite script is needed to add missing information since this script only writes the coordinate information.

ref_fix_msite M [.ref]

reffixmasite will fix the forces, solvation factor, and molecule names for M site added with xyz_add_msite.

5.3 Scripts to extract parameters from the *.off*

These scripts can be used to extra fitted parameters from the *.off* file for inspection, for updating .ff file, or for updating the .top file for gromacs.

offget_inter file.off intkey,[lnA[-B], [lnA[-B] [intkey,[lnA[-B], [lnA[-B]]…

Gets the intermolecular interaction terms from *.off* file.

file*.off* is the *.off* file.

intkey is the CRYOFF interaction names in the *.off* file. It might be a good idea to use more informational names such as EXPinter, EXPintra when writing the .ff file. This is an exact match for the interaction name not a substring match.

Optionally a line number (ln) or a range of line numbers can be specified. *[lnA[-B]]* (A and B are line numbers). More than one line ranges can be specified for each key. They should be separated by comma.

For example,

offget_inter test.off EXP POW,ln1,ln3-10 STRC,ln8

The entire EXP potentials in the *test.off* file will be extracted.

The line 1, and 3 through 10 for the POW and line 8 for the STRC will also be printed.

offget_intra file.off molname

Get the intramolecular parameters of the molecule *molname* from *.off* file.

file.off is the *.off* file name and *molename* is the molecule name in *.off* file.

This script will print a summary of all intramolecular parameters.

offget_atmtype file.off molname

Print out the all the atoms types in the molecule *molname*

offget_charge COU,atm1/atm2,chgdiv,lnA[-B][,lnA[-B]] file.off

Get the fitted charges from the off file, *file.off*.

The first argument contains comma delimited fields that describe the position of the charges to be extracted. The first field can be *COU* or *THC*(Thole damped coulombic). The second field can be either *atm1* or *atm2*, if *atm1* is specified, the name of first atom of the charge products in *.off* will be printed first, otherwise the second atoms will be printed first. The third field, *chgdiv*, determines how to obtain the charge. If *chgdiv* is a number, the charge product will be divided by the number, if *chgdiv* is *sqrtp* or *sqrtn*, the positive or negative square root of the product will be obtained.

The field *lnA* tells script to get the charge product from line A (e.g. *ln5*). One can also use more than one line or ranges, see the description for *offget_inter*

offget_tabparam file.off [atom1 atom2:atom3 …]

Extract the table file generation directives from the table potential section of the *.off* file.

By default, the *offget_tabparam* script will print table directives for all pairs listed in the *file.off*. This list can be shortened using the optional arguments. If a single atom (*atom1*) is provided as one of the optional arguments, any atom pairs involving this atom name will be omitted. If a pair of atoms (*atom2:atom3*) are provided, any pairs that can be formed by these two atoms are omitted. (For example, atom2-atom2 pair is omitted.)

offgen_tab protocol file.off

This script generates tabulated potentials in the Gromacs tabulated potential form according the parameters in *.off* file (*file.off*).

The protocol file has one or two fields in the following format

grid-definition [output]

E.g. [debug,]2.0,0.002 [pair=H1~O1][,prefix=Ala7]

The *grid-definition* field provides the cutoff and grid spacing, (2.0 nm , 0.002 nm) in the example. The first entry can also be *debug*. The *debug* entry tells the script to keep all intermediate files that contain each term of the potentials.

The *output* field is optional. It could contain a *prefix* entry or a *pair* entry. If a *pair* entry exists, only the tabulated potential for the pair (in the *atomA~atomB* format) will be created. If no *pair* entry exists, tabulated potentials for all possible pairs will be generated. The *prefix* entry controls the name for the tabulated files generated.

The POW, -6, term will be put as the attractive potential. All other short-range terms will be put in the repulsive potential columns of the table file. The coulombic columns in the table file will be computed using $q_i q_j / 4\pi\varepsilon_0 r$, where the charge product $q_i q_j$ is extracted from the *.off* file. This is mostly to make the plotting of the interactions easy since the plotting program only need access to the tabulated potential file without the need to look up charges from another source. However, such a tabulated potential cannot be used for coulombic=user in Gromacs, since Gromacs requires such user coulombic column to be 1/r.

offget_frc file.off [ref/fit/dif] [nef/torq]

This script extracts the forces from *.off* file (file*.off*). (only the forces reported in the .off file will be printed.)

The first optional argument species whether the reference (*ref*), the fitted (*fit*) or the difference of fitted of reference (*dif*) forces will be extracted. The default is fitted forces.

The second optional argument species whether the net forces(netf) or the torq(torq) will be extracted. The first optional argument should be provided if the second one exists. The default is to print all forces including atomic forces and molecular net force and torques.

5.4 off2ff: update ff file from a off file.

It is recommended to generate the .ff for the initial fitting manually. The scripts are designed to update these files for subsequent fits. Not for generating a .ff from scratch.

The Gromacs command *pdb2gmx* can be used to generate the bonds, angles, dihedrals and the nonbonded interactions, including the exclusions. It is very useful for generating the initial *.ff* file.

Update a *.ff* file based on a *.off* file is needed during the fitting steps of AFM. This is accomplished by *off2ff*.

off2ff protocol file.off input.ff out.ff

Four files are provided as arguments. The first argument is the name of the protocol file, which directs the transcription of parameters. The format of the protocol file is described below. *file.off* is the name of the *.off* file used to provide parameters. The *input.ff* is the name of the input *.ff* file where the parameters will be updated. The updated *.ff* file is written to the *out.ff* file.

The protocol file uses the following format:

action off_param ff_param

or
condition conditions rm/enable off_param ff_param

action can be *copy* (copy parameters), *populate* (populate section with parameters), *charge* (update charge products), or *condition* (update parameters according to condition)

(a) An action of *copy* directs the *off2ff* script to copy parameters from *.off* file to *.ff* file.

copy	*COU,col4*	*COU,col4*
copy	*COU,col4,ln1-10,ln15*	*COU,col4,ln1-10,ln15*

The second field (*off_param*) defines the positions in the *.off* file where the parameters are taken.

It is worth noting that when counting the column in the .off file. The atom pair name such as OW~O1 is always count at two columns. This is due to the service script offget_inter always treat this as two separate columns.

And the third (*ff_param)* field defines the positions in *.ff* file where the parameters should be updated.

The parameter position fields contain comma delimited entries.

The second entry of the *off_param* is the column number (eg. col4). In the example above, the 4th column in the COU section of the *.off* file will be used to update the 4th column of the COU section in the .ff file. The script assume there is a one to one correspondence (in sequence of interaction specified) between the *.off* file and the .ff file. Use this with caution. We note this case, the *ff_param* field must also have two entries with the second entry being the column to be updated.

A third (optional) entry can be specified for the *off_param* field, listing range of lines. In this case, only the corresponding column of the specific lines are updated. The *ff_param* fields must also specify the same number of lines. A one to one correspondence in the range specified is expected. The lines that have been commented out in *.ff* file will still be counted when updating. Thus, the comment indicator will be ignored. The design is to ensure lines removed automatically by the script won't affect the of line number count.

We note that for the *copy* action corresponding line selections in the *off_param* and *ff_param* must have a one-to-one correspondence.

(b) An action of *populate* directs the *off2ff* script to populate a section of the *ff* file using parameters from *.off* file

populate	*EXP[,ln1-8,ln15]*	*EXP,fix/fit*

The second (*off_param*) field defines the section in the *.off* file where the parameters are taken.

And the third (*ff_param)* field defines the section in *.ff* file where the parameters are to be populated. The new parameters are to be added to the selected section in the .ff file.

For the *off_param,* the first entry is the interaction type. The line number specification can be ignored, in this case all interactions with this type will be copied.

The second entry for the *ff_param* field can be either *fix* or *fit*. The corresponding parameters populated will be set to either fix or fit based on this entry.

(c) action charge updates the charge products in the ff file.

charge	*neutral*	*COU,col4,ln64-156*
charge	*file=chargefile*	*COU,col4,ln64-156*
charge	*COU,atm1,chgdiv,ln8,ln12,ln16*	*COU,col4,ln64-69*

ff param has comma delimited entries.

The first entry of the *ff param* is the interaction type, which has to be either COU or THC. The second entry has to be col4, since the charge product is always the 4th column in the .ff file. The third entry are line specifications as described above.

In many cases, the charge product can be updated using the copy action. In certain scenarios, for example intramolecular coulombic derived from intermolecular fit, one has to calculate the required charge products with the charge action.

Neutral is a special *off parameter*, in this case, the charge product lines specified in the ff parameters are reset to zero. This is useful for completeness checks for later charge product updates in that any charge product of zero was not properly updated.

The *off_param* are also comma delimited entries.

The *off_param* field provides charges needed for the script to compute the charge products.

There are two ways to specify the charges. One is relying on the *.off* file, by calling the *offget_charge* script. In this case, the *off_param* field is the argument to be passed to *offget_charge.*

For the example, (*COU,atm1,*0.6688*,ln8,ln12,ln16),*

the charge used will be the charge of *atm1* computed based on lines 8, 12, and 16 in the *.off* file.

The script will go through the lines specified in the *ff param* field and update charge products for all the pairs with both atom charges available. (through *off_param).* Other charge product are left as is.

Alternatively, the *off param* field could provide a file (*file=chargefile*). chargefile is a file that contains a list of atomic charges. (free format one charge per line as the example below) The charges needed for the charge product will be read from the file.

H 0.13328
H1 0.12905
H2 0.41648
H3 0.45138

(d) *condition*: remove or enable interaction terms in .ff file based on conditions.

condition conditions rm/enable off_param ff_param

The *conditions* field can select parameters to remove or enable based on distance(*dist*), parameter value (*val,max,min*) or charge product(*chgprod*) in the *.off* file.

See example.

condition	*dist,gt,3.0*	*rm*	*STRC,ln1-10*	*STRC*
condition	*dist,gt,3.0*	*rm*	*STRC,ln1-10*	*POW*
condition	*dist,gt,3.0*	*rm*	*STRC*	*STRC*
condition	*dist,gt,3.0*	*rm*	*STRC*	*POW*

condition	*val,gt,0.0*	*rm*	*STRC,col4[,ln31-33]*	*STRC*
condition	*val,gt,0.0*	*rm*	*STRC,col4[,ln31-33]*	*POW*
condition	*val,gt,3.0*	*rm*	*STRC*	*STRC*
condition	*val,gt,3.0*	*rm*	*STRC*	*POW*
condition	*max*	*rm*	*STRC,col4[,ln31-33]*	*STRC*
condition	*max*	*rm*	*STRC,col4[,ln31-33]*	*POW*
condition	*max*	*rm*	*STRC*	*STRC*
condition	*max*	*rm*	*STRC*	*POW*
condition	*chgprod,gt,-0.25*	*rm*	*COU,atm1,chgdiv,ln8,ln12,ln16,ln20*	*STRC,ln31-33*
condition	*chgprod,gt,-0.25*	*enable*	*COU,atm1,chgdiv,ln8,ln12,ln16,ln20*	*COU,ln64,ln80*
condition	*chgprod,gt,-0.25*	*enable*	*file=chargefile*	*THC,ln1-4,ln7*

dist,gt,3.0 means if the minimum distance is greater than 3.0 A." gt" can be replaced by lt (les than), ge (greater equal) and le (less equal).

val,gt,0.0 means if the parameter value in the column specified is greater than 0.0. "gt" can be replaced by lt, ge and le.

max means the maximum value of the parameters in the range of lines specified. "max" can be replaced by "min"

chgprod,gt,-0.25 means if any charge product specified in the lines in the *ff_param* field is greater than -0.25.

rm/enable controls whether the line (or lines depending on the condition) should be enabled or removed.

For distances and value conditions, the script will check for all the lines specified in the *off_param* field, enable or remove the lines corresponding to the atomic pairs in the corresponding section in the *ff_param* field.

The *off_param* field follows the convention of interaction *type[,line number]* for distance conditions, interaction *type, column number[,line number]* for values. The column number specifies the column number of the parameter to be checked.

The *ff_param* field should be section name for the interaction type for distance, and value conditions.

For example,
condition val,gt,0.0 rm STRC,col4[,ln31-33] POW
scans line 31 to 33 in the STRC section in the off file look for any value in column 4 that is greater than 0.0, once a much is identified, the corresponding entry (for the same pair of atoms) in the POW section of the .ff is to be commented out. (remove)

For charge product, the *off_param* follow the same format of the *charge* action (see (b) above), all the charges needed for computing the charge products in the *ff_param* lines must be present, otherwise, the charge products will be assumed to be zero.

The *ff_param* specifies the interaction type and line numbers for the charge product condition. The lines commented out will be counted for the line number specifications.

For example,

condition chgprot,gt,-0.25 rm COU,atm1,chgdiv,ln8,ln12,ln16,ln20 STRC,ln31-33

Scans line 31 to 33 of the shifted-truncated-repulsion section, computes the charge products for the atom pairs encountered based on the atom charge from the *offget_charge* script and disable lines with charge product greater than -0.25.

5.5 Scripts being used by off2ff

These scripts are service script for off2ff that must be in the PATH for off2ff to function. They can be used alone. The usage for most of these script will not be elaborated since there is generally no need for the end user to use them.

offget_inter: get inter potentials from *.off* file.

offget_intra: get intra potentials from *.off* file

offget_charge: get fitted atom charges from *.off* file.

offget_tabparam: get tabulated potential parameters form *.off* file

offget_atmtype: get atom type of one molecule.

ffgetpam: get parameters from *.ff* file by line number.

ffupdbynum: update *.ff* file according line number.

ffaddremtermbynam: add or remove terms of *.ff* file according atom names.

getcharge: calculate charges based on charge product.

off2ff_copy: copy parameters from *.off* to .ff.

off2ff_charge: calculated charge product and update the .ff file.

off2ff_bymindist: add or remove terms based on the minimum distance of two atoms.

off2ff_byvalue: add or remove terms based on the fitted parameters.

off2ff_bymaxminpam: add or remove terms based on the fitted maximum or minimum parameters.

off2ff_bychgprod: add or remove terms based on the fitted charge product.

6. Scripts developed to update/generate topology or tabulated potential files for Gromacs.

6.1 Introduction

Gromacs allows the same force field to be defined in several possible ways. With our scripts, it is recommended to create Gromacs topology using the following procedure.

Below, we will use the term topology file to refer to either the .top or .itp file of Gromacs.

In gromacs, it is possible to separate the force field definition into molecular definition files and parameter files. For more complex molecules, it is probably easier to use the parameter files.

We recommend to create a separated molecular definition file for each molecule type.

To use our scripts, one has to first generate an initial topology files and our scripts are designed to update an existing topology file with new parameters. The pdb2gmx script distributed with gromacs is very useful to generate the initial topology file.

The parameter file can include the non-bonded and bonded sections. It is generally more convenient to create multiple parameter files for these sections.

The non-bonded section includes [atomtypes] and [nonbond_params].

When using our scripts, we recommended to use bonded sections in the parameter file instead of define the bonded parameters in the molecule definition files.

The bonded sections may include [bondtypes], [angletypes] and [dihedraltypes].

6.2 Update the Gromacs topology and generate tabulated potential files

The update of parameter and molecule definition files, including generating the tabulated potential files are accomplished by the script off2top.

off2top protocol file.off templtop outtop

The off2top script is the main script to update the parameter and molecule definition files.

protocol is the protocol file used to define the rules for the update. *file.off* is the CRYOFF .off file for extracting parameters. *templtop* is the template topology file based on which the output topology file *outtop* will be written.

The protocol file uses the following format:

action off_param top_param [unit] control

The *off_param* is typically passed to the off_get* scripts and the syntax is similar to that used for off2ff.

The top_param has three or more comma delimited entries. When updating parameters, the first entry is the section name in the topology file such as [atomtypes], the second entry is the column number (col4) , follow by line specifications. When updating molecular specification, the first entry is the

molecule name (ALA7), the second entry is section name, the third and fourth entries are column and line numbers. More than one line number fields are always supported.

The unit field is included for updating parameter, but not for generating lists. When the unit field is present, the parameter extracted from the off file is multiplied by the number in the unit field before it is written to the topology file.

(I) Actions for updating the "parameter files"

(a) Atomic charges

Action *pam.charge* will update the atomic charges in the parameter section [atomtypes] of a gromacs topology file.

Example:

pam.charge	*COU,atm1,0.6645,ln8,ln10,ln12*	*atomtypes,col3,ln1-3*	*1.0*	*default*
pam.charge	*file=chgfile*	*atomtypes,col3,ln1-3*	*1.0*	*type=filename*

The *off_param* field is the charge position in the *.off* file. This field has the same format as the first argument of *offget_charge.*

The top_param field is the position of charges that to be updated in parameter file. For gromacs, the first two fields should be atomtypes,col3.

The script looks up atom names for the lines specified in the *top_param* field and obtained the required charges from the lines in the *off_param* field. When the off*_param* field specify a file, the charges will be looked up from file.

For charges, the *unit* field should always be 1.

The control field can be used for atom type conversion. "default" indicates the same atom types will be used for CRYOFF and Gromacs. If *type=filename* is provided, atom type translation will be done based on the name pairs in the file *filename*. The file has two columns, the first column is the atom type used by Gromacs, and the second is the atom type in CRYOFF. If the atom type used in Gromacs is not found in the translation file, no atom type translation will be done.

In the translation file, lines start with # will be treated as comment lines.

Example translation file:

#Gromacs	CRYOFF
H1	HB1
HW	HW4f
C1	CA

(b) Nonbonded list

This action will add the list of pairwise nonbonded interactions to the topology file, to populate the [nonbond_params] section. The atom pairs that only have coulombic interactions between them will not be added.

Example:

list.nonbond	*default*	*nonbond_params[,1,1.0,1.0]*	*default*
list.nonbond	*HW~MW,HW~OW,HMM,MMM*	*Nonbond_params*	*type=filename*

The off_param field specifies the exclusions that is passed as the 2nd argument of offget_tabparam. "default" indicate all pairs without any exclusion will be added.

The top_param field is the nonbonded section name [nonbond_params] followed with the function type and parameters for functions. The function type and parameters are optional. The default function type is 1 and the parameters are 1.0. (as in the example)

The control field is used for type conversion. It can be default or type, which have the same meaning as the control field in section I (a).

(c) Nonbonded parameters:

Action pam.nonbonded will update nonbonded parameters in parameter file section [nonbond_params].

Example:

pam.nonbond	*STRC,col4*	*nonbond_params,col4,ln1-10*	*4.184*	*default*
pam.nonbond	*STRC,col4*	*nonbond_params,col4,ln1-10*	*1.0*	*type=filename*

The off_param field is the parameter position to be extracted from the .off file. This field contains the potential name in .off and the column number. The column number is actually the column position of the output of offget_inter, which was made to be the same position as the parameter in the .ff file.

The top_param, unit, and control has the same format and meaning as those for the pam.charge action described in section I (a).

(d) bonded list

Action list.bonded will add the list of nonbonded interactions to the parameter file, to the section [bondtypes], [angletypes] or [dihedraltypes].

Example:

list.bonded	*ALA7*	*bondtypes[,1]*	*default*

list.bonded	*ALA7*	*angletypes[,1]*	*type=atmtype*
list.bonded	*ALA7*	*dihedraltypes[,9]*	*intkey=har*
list.bonded	*ALA7*	*dihedraltypes[,9]*	*type=atmtype,intkey=har*

The off_param field is the molecule name in the .off file.

The top_param field contains the section name in the topology file. A number can be appended after the section name, which will be used as the function type in the topology file. The default function type is 1.

The control field can be default or type as discussed in section I(a).

The intkey keyword also in the control field allows selective construction of the list in the section selected by the top_param field. If a type of interaction is specified with intkey=inttype. Then only the specified type of interactions from all the interactions extracted according to the top_param field will be added to the list constructed.

(e) Bondtypes, angletypes and dihedraltypes parameters

Action pam.bonded will update the bonded parameters in the gromacs parameter file section [bondtypes], [angletypes] or [dihedraltypes].

Example:

pam.bonded	*ALA7,col4*	*bondtypes,col4,ln1-8*	*1.0*	*default*
pam.bonded	*ALA7,col4*	*angletypes,col4,ln1-20*	*1.0*	*type=filename*
pam.bonded	*ALA7,col4*	*dihedraltypes,col7,ln1-8,ln10*	*1.0*	*extra:col5=col8*
pam.bonded	*ALA7,col4*	*dihedraltypes,col7,ln1-8,ln10*	*57.29*	*default*

The *off_param* field specifies the parameter position in the off file.

The format is the molecule name (e.g. ALA7) followed by the column (e.g. col4) number.

The *top_param* field is the position in the parameter file. The position is specified by the section name, (e.g. bondtypes) followed by the column (col4) number and line number (ln1-8).

The *unit* field is the unit conversion from CRYOFF to Gromacs. The parameters from CRYOFF will multiply this value before updating the topology file.

The *control* field can be *default*, which means the atom types are same for Gromacs and CRYOFF, alternatively, a file can be supplied with *type*=filename. In this case, the name translation between Gromacs and CRYOFF will be done according this file.(See translation file example in Sec 1(a)).

When updating dihedrals occasionally there are more than one torsional terms for each dihedral. In order to distinguish these terms, the extra keyword can be put in the *control* field.

extra:colA=colB. This will direct the script match up colA in the .off file and colB in the gromacs parameter file in addition to the specification of the interaction (eg. C1_C2_O1_H1) before updating the parameter.

With the *extra* column constraint also works for other bonded interaction types although it was designed primarily to update dihedral parameters.

(II) To update molecular definitions:

(a) Atom charges

mol.charge	*COU,atm1,0.6645,ln8,ln10*	*Ala7,atoms,col7,ln1-13*	*1.0*	*default*
mol.charge	*file=chgfile*	*Ala7,atoms,col7,ln1-13*	*1.0*	*type=atmtype*

The mol.charge section follow the same format as the pam.charge action except this action updates the gromacs molecule definition file.

(b) Bonds, angles and dihedral

mol.bonded	*ALA7,col4*	*Ala7,bonds,col4,ln1-72*	*1.0*	*default*
mol.bonded	*ALA7,col4*	*Ala7,angle,col4,ln1-129*	*1.0*	*type=atmtype*
mol.bonded	*ALA7,col4*	*Ala7,dihedrals,col7,ln1-8,ln10*	*1.0*	*extra:col5=col8*

The mol.bonded action update the gromacs parameter file. This section is similar to the pam.bonded action except for the *top_param* field.

For this action, the first two entries of the *top_param* field are the molecule name and the section names in molecule definition files, respectively.

off2tab protocol file.off topology.top

The off2tab script is used to generate the tabulated potential files.

protocol is the protocol file used to define the rules for the tabulated file generation. *file.off* is the CRYOFF .off file for extracting parameters. *topology.top* is the topology file based on which the pairs needing tabulated potential is determined.

The protocol file uses the following format:

action grid_line_field control

The action can be *tab.nonbond or tab.bonded*

(a) Generate nonbonded tables

Action tab.nonbond will generate tabulated potentials for selected pairs in the [nonbond_params] section of parameter file.

tab.nonbond	*[debug,]3.0,0.005,ln3*	*no*
tab.nonbond	*3.0,0.0005,ln1-9,ln15*	*scale=C6,type=atmtype,prefix=Ala7*

The *grid_line* field is used to provide grid-specifications followed by a series of line numbers. (comma delimited).

The grid specifications should consist of the *cutoff* (3.0nm) and *grid spacing* (0.0005 nm). The line numbers specify the line numbers in the [nonbond_params] section of the template topology file. If a debug entry is provided before the grid specifications, the script will keep all intermediate files that was generated in the construction of the final table file.

The columns for coulombic interaction in the tabulated potential files is filled with the 1/r and $1/r^2$ for undamped coulombic interactions.

The control field support the following keywords: type, scale, prefix.

"type" specifies the name conversion file to convert names used by CRYOFF and Gromacs. "prefix" is the prefix of the generated table files. "scale" only affects the POW, BUK, and TTP types of interactions. The power law terms in these potentials will multiplied by the value specified with the scale=value entry. If entry is scale=C6, the attractive potential in the gromacs tabulated potential file is multiplied by the reciprocal of the power law prefactor.

The off2tab script also write a tab_list file, the tab_list file list pairs of table potentials generated to be used to populate the energygrp_table in the gromacs mdp file.

Example work flow for updating the topology file:

For complex molecules, one way to generate the topology file is to use the following steps:

Step 1: update the parameters in the nonbonded itp/top file. (off2top section I. c) Update the nonbonded list if necessary. (off2top section I. b)

Step 2: generate tabulated potentials according the list of nonbonded interactions. (*off2tab)*

Step 3: updated the parameters in bonded.itp. (off2top section I. e.)

Step 4: update the molecule charges in molecule definition file. The atomic charge can be updated both in the parameter file or molecular definition file. The values in the molecular definition file supersedes that in the parameter file. Thus if the molecular definition file is used, one should follow the off2top section II.a

6.3 Scripts being used by off2top and off2tab

The off2top scripts is developed based on following service scripts. The service script listed below can also be used independently.

topgetparam_paramfile: get parameters from parameter file.

topupdate_paramfile: update parameters of parameter file.

topgetparam_moldef: get parameters from molecule definition file

topupdate_moldef: update parameters of molecule definition file

topadd_nonbondedlist: add pairs to [nonbond_params]

gentab_pow_ttp_srd: generate POW/TTP/SRD potentials.

gentab_exp_buck_strc: generate EXP/BUCK/STRC potentials

gentab_thc: generate Thole damped coulombic interaction

off2top_nonbonded_charge: update the atom charges in parameter file.

off2top_nonbonded_list: update the list of nonbonded interactions in parameter file.

off2top_nonbonded_param: update the nonbonded parameters in parameter file.

off2top_nonbonded_gentab: generate the table files according the off file and the nonbonded list in parameter file.

off2top_bonded_list: update the list of bonded interactions in parameter file.

off2top_bonded_param: update the parameters of bonded interactions in parameters file.

off2top_molecule_charge: update the atom charges of the molecule.

off2top_molecule_bonded: update the bond and angle parameters of one molecule.

7. Known Limitations

(1) For .ff and Gromacs input files, comment characters can only appear at the beginning of the lines. If appear in other position, such as the middle or the end of one line, the script won't interpret anything after the comment characters as comments.

(2) For the .ff file, in order to use the scripts correctly, two sections with the same section name are not allowed, for example, you cannot have two EXP sections. But you can name these sections as EXPinter and EXPintra in the .ff file. Thus naming will not change the fitting results, but can allow the scripts to function correctly.

(3) For Gromacs topology files, the scripts do not support two sections with identical names in one parameter file. Also, for each [moleculetype], the script won't support two sections with identical names.

8. Language convention and terminology used in the manual

	explanation	example
Italic	name of scripts, input arguments for scripts, protocols,	*gro2pxyz*

	executable examples.	
colA	specify a column number	*col3*
lnA	specify a line number	*ln3*
lnA-B	specify a range of line numbers	*ln5-9*
[abc]	optional arguments	*gro2pxyz [grofile]*
/	OR	*fit/fix*
gt	Greater than	
ge	Equal or greater	
lt	less than	
le	equal or less	
rm	Comment out line(s) for a file	
enable	Delete the comment character before line(s)	
add	Add line(s) to a file	
.ref	input reference file for CRYOF	
.ff	input fitting file for CRYOFF	
.off	output file from CRYOFF	

9. Usage example

We will use the development of an alanine potential as an example.

9.1 Generate the QM/MM region

Assuming the sampling is done with gromacs, an initial conformation will be prepared where each molecule is made whole and the box is centered using the molecule to be the center of the QM region. When doing so, one does not need to worry about periodic boundary conditions later in the workflow.

With gromacs one way to achieve this is

gmx_d trjconv -s ../topol.tpr -n ../index.ndx -f ../traj_comp.xtc -dt 30 -pbc atom -center -o temp.xtc

gmx_d trjconv -s ../topol.tpr -n ../index.ndx -f temp.xtc -pbc whole -sep -o alpha.gro

To create a pxzy file for the QM/MM calculation for the hydrated alanine, we can do the following:

Step1: prepare pxyz file from gro file

Assume the MD was run with gromacs, we first create the pxyz file with

gro2pxyz alpha0.gro > alpha0.pxyz

Step2: Define the QM/MM region.

Let's assume we use the following protocol (underlined) to define the QM/MM region.

(a) The Ala7 is included in the QM region.

mark_bynam ALA 4 alpha0.pxyz > temp ; mv temp alpha0.pxyz

Here we will use a mark value of 4 for Alanine atoms.

(b) If a water molecule has any atom within 4.5 Å of a carbon atom or within 3.8 Å of any other Ala4 atom including the hydrogen atoms, it will be included in the QM region.

mark_within_range 1 73 3.8 2 alpha0.pxyz

This command will mark all molecules within 3.8 A of any Alanine molecule, which is atoms 1 to 73 to a value of 2.

Followed by

mark_within_list 4.5 2 alpha0.pxyz 5 7 11 15 17 21 25 27 31 35 37 41 45 47 51 55 57 61 65 67 71

This command will mark all atoms with 4.5 A of the carbons atoms 5, 7 …. 71 to a mark value of 2

(c) Randomly select five water molecules from the inner QM region. All the water molecules within 2.6 Å of the selected water molecules will be included in the QM region.

In the first step, mark 5 random water.

markup_random 2 3 alpha0.pxyz > temp ; mv temp alpha0.pxyz

five times. Or you can do

markup_random 2 3 alpha0.pxyz| markup_random 2 3 | markuprandom 2 3 | markup_random 2 3 | markup_random 2 3 > temp; mv temp alpha0.pxyz

or you can write a loop in your favorite scripting language.

In the second step, mark up all MM water with 2.6 A of these water to solvation factor 0 water.
This can be accomplished by

mark_boundary 2.6 3 2 alpha0.pxyz

(d) the Ala7 and any water without MM point charge within 2.6 Å will be hydration factor 1.

markup_mol 2.6 2 3 alpha0.pxyz > temp ; mv temp alpha0.pxyz

This script will increase (markup) the mark of molecule thus raise hydration factor 0 water to hydration factor 1.
After this step, all the solvation factor 1 water have been marked as 3, solvation factor 0 water have been marked as 2.

(e) All water molecules within 7 Å of Ala_4 heavy atoms but not included in the QM region are identified as MM molecules.

mark_within_list 7.0 1 alpha0.pxyz 1 5 7 11 12 13 15 17 21 22 23 25 27 31 32 33 35 37 41 42 43 45 47 51 52 53 55 57 61 62 63 65 67 71 72 73

After the QM/MM region are defined, the other atoms should be dropped by by

pxyz_dropoff 0 alpha0.pxyz > temp; mv temp alpha0.pxyz ,

since they are not needed for the QM/MM calculation anymore.

To make it cleaner, molecules can be rearranged by pxyzsort

pxyz_sort alpha0.pxyz > temp; mv temp alpha0.pxyz;

9.2 Generate QM/MM calculation input files

We will need to create input files for the QM/MM calculations. For some electronic structure codes, scripts have been created to update an existing input file with new QM/MM configuration specifications for each frame in the training set obtained in the sampling step.

If molpro is to be used to perform the QM/MM calculations, the **Molpro** input file (.inp in the example below) can be updated as below

pxy_dropoff 1 alpha0.pxyz|grep -v MW|xyz_fix_lineno |pxyz_upd_molpro ala_molpro_templ.inp nucinfo.molpro > alpha0.inp

In this example, the ala_molpro_templ.inp, is the molpro input file properly configured to perform the QM/MM calculation. The alpha0.inp file is the new molpro input file taking configuration from alpha0.pxyz. QM atoms with hydration factor 1 and 0 are marked with values of 2 and higher. The grep command was used to remove the M site in water.

To provide MM information for the molpro calculation, we put the point charge lattice in a file, named alpha0_mm.inp

pxyz_select 1 alpha0.pxyz|grep -v OW|xyz_add_lineno |pxyz_gen_molpro_lattice chginfo.molpro > alpha0_mm.inp

Then, the lattice file name in molpro input file can be updated with the follow command

molpro_replace_field INFILE INFILE=alpha0_mm.inp alpha0.inp > temp ; mv temp alpha0.inp

If GAMESS is to be used to perform the QM/MM calculations, the input files can be prepared/updated with the following command.

For the QM region,

pxyz_dropoff 1 alpha0.pxyz|grep -v MW| xyz_fix_lineno|pxyz_upd_gms ala_gms_templ.in nucinfo.gms > alpha0.inp

Here ala_gms_templ.in is the working GAMESS input file designed to perform the QM/MM calculation.

GAMESS supports MM partial charges. However, the way GAMESS implement the MM charges is quite cumbersome. You will have to define fragments in the fragment section of a GAMESS input file.

pxyz_select 1 alpha0.pxyz|grep -v OW|xyz_add_lineno |pxyz_upd_gmsfrag alpha0.inp chginfo.gms > temp ; mv temp alpha0.inp

If PQS is to be used to perform the QM/MM calculation, the following commands can be used to create input files.

PQS can read a headless xyz file as molecular specification. It can be generated with the following command.

pxyz_dropoff 1 alpha0.pxyz|grep -v MW |xyz_fix_lineno |pxyz_2vxyz |pxyz_gen_pqs_xyz nucinfo.pqs > alpha0.xyz

The name of the headless xyz can be updated in the PQS input file using the following command:

pqs_replace_field FILE FILE=alpha0.xyz ala_pqs_templ.inp > alpha0.inp

There are multiple ways in PQS to specify partial charges. Some older version of PQS may not support all the methods for partial charge specification. This command provides the .pntqinp that can be used for newer version of PQS.

pxyz_select 1 alpha0.pxyz|grep -v OW|xyz_add_lineno |pxyz_gen_pqs_pntq chginfo.pqs> alpha0.pntqinp

9.3 Generate .ref file for CRYOFF

An example procedure for generating the .ref file are shown below:

a) prepare a headless xyz file contains atom positions used to construct the ref file. The position unit is angstrom. You can use either the QM calculation output file to extract such an xyz or use the pxyz file you created previously for QM/MM setup.

   For example:

   pxyz_2vxyz alpha0.pxyz |grep -v MW3| grep -v MW2 |chunk 2 10000 > qmmm.xyz

b) Use refgen_step1_cord and molecule information file to generate a ref file. The forces are undefined after this step.

   refgen_step1_cord molinfo.ala qmmm.xyz > alpha0.step1.ref

c) Update the forces of the ref file generated in the last step.

There are a series commands that can do this. One can easily add up forces from different code such as DFT and dispersion correction. Since the update script always read in the initial force value in the ref file and add the new component to it.

ref_upd_pqs_frc alpha0.grad alpha0.step1.ref > temp.ref

d) Update the torque and net forces using refupdnet.

ref_upd_net *temp.ref >* temp1.ref

e) You may need to fix dummy sites with the xyz_add_miste script

f) At the end, you will have to fix the line numbers since the refgen_step1_cord script also left this as TBD (to be determined)

ref_fix_linenu temp1.ref > alpha0.ref

For example, you can do all of the above in one line: (assuming PQS was used to calculate the forces)

pxyz_2vxyz alpha0.pxyz |grep -v MW3| grep -v MW2 |chunk 2 10000|refgen_step1_cord molinfo.ala |ref_upd_pqs_frc alpha0.grad|ref_upd_net |xyz_add_msite Ow Hw Hw Mw|ref_fix_msite Mw |ref_fix_linenu > alpha0.ref