qfeper

Masoud Kazemi

October 4, 2018

Introduction:

qfeper is a program to write fep files used by the **Q** package. For each state, one topology file is needed and all information is read from topology files. The program needs an input file containing the number of states, the name of each topology file, the number of q atoms, and a one to one mapping (see tests). The command *./qfeper* will print out the format of instruction file.

Program usage:

```
./qfeper h              # For instruction file format
./qfeper input          # For creating fep file
./qfeper input p        # For printing details information on std output
./qfeper input s        # For spliting the fep file to 2 states fep files
./qfeper input sp       # It is also accepted to add both functions (sp or ps)
```

How it works:

The reference state is the first topology file. All atom numbers are translated to first state numbering. Only the bond, angle, torsion, and improper will be considered that their atoms are q atoms. This implies that the user has to choose the q atoms so that its range covers all possible perturbation. Subsequently, the parameters will be compared in different states and only the ones that are changing will be printed out to fep file. The procedure is different for atom types and charges. All of q atoms type and charge will be added to fep file (it make it easier to further manipulate the fep/EVB if needed and have no effect on final results). The coupling will be suggested based on presence of breaking/forming bond in a angle, torsion and improper. The program can handle more than two states; however, the write out format might become messy in case of more than six states.

How to compile:

You can move to the src folder and just run:

```
For Intel Fortran:
  ifort qfeper_pars.f90  qfeper_analyz.f90 qfeper.f90 -o qfeper

For GCC fortran:
  gfortran qfeper_pars.f90  qfeper_analyz.f90 qfeper.f90 -o qfeper
```

```
To debug:
  ifort -check all -debug all qfeper_pars.f90  qfeper_analyz.f90 qfeper.f90 -o qfeper
```

Or you can also use make and compile using:

```
  make                       # this will show options
  make all COMP=gcc          # this will compile and organize
```

Masoud Kazemi kazemimsoud@gmail.com