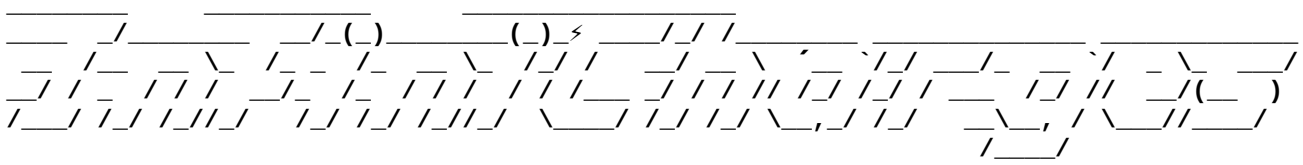

Version 1.0

Andrea Gabrieli and Marco Sant

Department of Chemistry and Pharmacy, University of Sassari, Italy

30th July 2015

InfiniCharges is a computer program for generating reliable partial charges for molecular simulations in periodic systems.
It relies on the DM-REPEAT method where the stability of the resulting charges, over a large set of fitting regions, is obtained through the simultaneous fit of multiple electrostatic potential (ESP) configurations together with the total dipole fluctuations (TDF).

This program performs the following kinds of fits:
- M-REPEAT (also standard REPEAT)
- DM-REPEAT (also D-REPEAT)
- PARABOLIC RESTRAINED M-REPEAT
- "RAPPE-GODDARD LIKE" RESTRAINED M-REPEAT

Index

Installation

1) Prerequisites, the following programs should be installed:
- GNU make
- Python 2.7 (already available in all recent Linux distributions)
- Numpy (and F2Py if not automatically installed with numpy)
- a recent Fortran 90 compiler (e.g., gfortran, ifort)

For example in Ubuntu Linux:
> sudo apt-get install make gfortran python-numpy

2) Build the atautils.so library:
> cd dir_atautils
> make

Running the Examples

- enter the wanted directory (“esp”, for example)
> cd examples/esp
- and run the program
> nohup ../../InfiniCharges.py > output &
- check the progress
> tail -f output

Running the Program

- create a working directory and enter it
> mkdir run1
> cd run1
- copy the input file InfiniInput.py
> cp ../examples/sample_InfiniInput.py InfiniInput.py
- modify the input as needed
- create a directory named “cubes” and copy here the cube files
> mkdir cubes
> cp path_to_your_cubes/cubefile.cube cubes
- alternatively, to avoid wasting space, you can create a link to the cube files
> ln -s path_to_your_cubes/cubefile.cube cubes
- only in case of “ESP+TDF” fit, create a directory named “tdf” and put here the TDF signal file (either a “.tdf” or a CP2K output file) and the trajectory in .xyz format
> mkdir tdf
> cp path_to_your_tdf/signal.tdf tdf
> cp path_to_your_tdf/trajectory.xyz tdf

- now it is possible to run
> nohup ../InfiniCharges.py > output &
- check the progress
> tail -f output

Restarting

The program automatically saves the computed data in directories called “ata_*”. Every time it is launched, it looks for such directories and if they are found the contained data will be used for the fit. This will save a lot of time if you want to perform different calculations over the same set of data.
Note: remember to remove such directories or to work in a new directory if some of the initial data (cubes or tdf) are changed.

Data Files

Electrostatic Potential
The ESP data must be provided as a set of *Gaussian cube* files, one for each system configuration to be included in the fit. By default the code works for cube files having all quantities expressed in atomic units. For ESP values computed with respect to a negative test charge (e.g., the cube files generated by CP2K through the V_HARTREE_CUBE option) the variable *esp_from_cp2k* should be set to *True*.

Total dipole fluctuations
The TDF data should be provided as a simple text file made by an header containing the periodicity of the crystal in Debye and by a list of total dipole moments having in each line the x, y, and z components, (see *examples/esp+tdf/tdf/signal.tdf*).
The program can also process directly the Berry total dipole files obtained from CP2K. To switch the two options set the *tdf_from_cp2k* variable accordingly.
At the same time, the system trajectory from which the total dipole data has been obtained should be provided as a .xyz trajectory file.
Note: the atom order in this file must be the same as in the .cube files and as in the “coor" file containing the atom types.

Input Files

In the following a point by point description of the input file for the program is given, a sample can be found in:
"examples/sample_InfiniInput.py"
all options should be given by setting the relevant values as Python variables.

fit_kind=

"ESP", to perform multi-frame ESP fit (M-REPEAT) which reduces to standard REPEAT if only one cube file is given;
"RESP", same as ESP with the charges restrained to desired values, requires to set *restraint_weight* and *restraints* variables;
"RG", same as ESP with Goddard-type restraints according to the original REPEAT paper, requires to set *rg_weight* variable and to set the electronegativities and the self-Coulomb interaction values inside the *rappe_goddard.py* file;
"ESP+TDF", to perform the simultaneous multi-frame ESP and TDF fit (DM-REPEAT), requires *tdf_from_cp2k*, *tdf_signal_fname*, and *tdf_coor_fname* variables to be set.

coor=
"system.xyz", system coordinates in *xyz* file format;
"system.pdb", system coordinates in *pdb* file format;
this file is used to define the atom-type groups, all atoms having the same name will attain the same charge.
Note: the order of the atoms in this file must be the same of the one used to generate the QM data. The values of the *x*, *y*, and *z* coordinates, instead, are not used.

esp_from_cp2k=
True, if ESP values have been computed with respect to a negative test charge, like in CP2K;
False, in all other cases.

g_start=
float, value indicating the first vdW scaling factor γ.

g_step=
float, separation between two consecutive γ values.

n_gammas=
int, number of γ values for which the fit will be performed.

Example: setting *g_start*=1.4, *g_step*=0.2, *n_gammas*=3 will generate charges for 3 different γ: 1.4, 1.6, 1.8.

Optional Values

ESP+TDF fit only:
tdf_from_cp2k=
True, if TDF values have to be taken from a Berry total dipole file generated with CP2K;

False, if the custom .tdf format is used (see “Data Files” section).

tdf_signal_fname=
"filename", name of the file containing TDF data.

tdf_coor_fname=
"trajectory.xyz", name of the file containing the system coordinates from which the total dipole data has been obtained. It should be provided as a *xyz* trajectory file.

weight_tdf=
float, when ESP+TDF is selected, the program performs an analysis over 101 different weights for TDF. If this variable is set, only a fit for the given value is performed.

RESP fit only:
restraint_weight=
float, single weight for all restraints given in the *restraints* variable;
list of tuples, [("type", float), …] with a specific weight value for each restraint given in the *restraints* variable.

restraints=
list of tuples, [("type", float), …] with the target value of the restraint for each atom-type group to be restrained.

RG fit only:
rg_weight=
float, single weight, the same for all atom-type groups.

Ewald Calibration:
locut=
float, magnitude of elements to be discarded from the reciprocal space part of the Ewald summation. Default value is 1.0e-10 (when variable is not set).
The default value has been tested for systems like ZIF-8 and should work in materials having similar lattice size (up to 2.5 nm) and charges magnitude (up to 1.5 *e*).
To find the correct value for other systems, one should set *locut* to a very small value (e.g., 1.0e-30) to compute the reference ESP data.
These should be used for comparison with the ESP obtained using a larger (i.e., approximated but faster) value of *locut*.

Outputs

The results are written on the *standard output* under the voice “**Final rounded charges”**

and there will be one set of charges for each γ value required.
Note: charges are rounded by a stochastic procedure, which means that the last digit may change if the fit is performed twice for the same dataset.

atautils.log: contains informations about the generation of model data from the cube files.

ESP+TDF fit only:
- **auto_weight_res.dat:** contains the results along with the γ value, the stopping weight and the number of grid points used for the ESP part;
- **res_loop_*.dat:** contains the RRMS evolution for TDF and ESP as a function of the TDF weight. There is one file for each γ;
- **results_*.dat:** contains the rounded charges for each weight. There is one file for each γ;
- **results_unrounded_*.dat:** contains the raw charges for each weight. There is one file for each γ.
Note: the last two files are mainly for check purposes.

Results Visualization

The commands here presented are valid for the examples provided with the program.

1) With gnuplot it is easy to visualize the results:
```
gnuplot> f = "auto_weight_res.dat"
gnuplot> p f u 1:2 w lp, f u 1:3 w lp, f u 1:4 w lp, f u 1:5 w lp, f u 1:6 w lp, f u 1:7 w lp, f u 1:8 w lp
```

2) To extract results for a given weight for all γ use the following command, where “W” is the line corresponding to the wanted weight in the file “results_unround*” (here W=1 means pure ESP data):
```
> W=1; for i in `ls results_unround*`; do awk -v w=$W '{if (NR==w) print $0}' $i; done > charges_for_weight_$W.dat
```
- then in gnuplot:
```
gnuplot> f = "charges_for_weight_1.dat"
gnuplot> p f u 1:2 w lp, f u 1:3 w lp, f u 1:4 w lp, f u 1:5 w lp, f u 1:6 w lp, f u 1:7 w lp, f u 1:8 w lp
```

3) To see the RRMS evolution of TDF and ESP as a function of the TDF weight for a given γ (here γ=1.4):
```
gnuplot> p "res_loop_1.4.dat" u 1:4 w lp, "res_loop_1.4.dat" u 1:5 w lp
```