REPEAT – A Quick Guide

Carlos Campañá, Tom K. Woo

Department of Chemistry and Biomolecular Sciences
University of Ottawa, Ottawa, Canada, K1N 6N5

Version 2.0

Updated: November 2022

Introduction

This document describes how to use REPEAT code to generate electrostatic potential (ESP) derived charges for periodic systems as described in the paper Campañá, C.; Mussard, B.; Woo, T.K."Electrostatic Potential Derived Atomic Charges for Periodic Systems Using a Modified Error Functional: REPEAT Charges" *Journal of Chemical Theory and Computation*, **2009**, *5*, 2866-2878. (https://doi.org/10.1021/ct9003405) The code was originally written by Carlos Campañá and has since been updated by Tom Woo. The original version of the code was written in Fortran but now has been rewritten in C/C++. The Ewald routines have been adapted from the F22 routine provided with the book of Allen and Tildesley "Computer Simulations of Liquids". This software was created with the purpose of 'getting the work done' and less effort in making it clean and user friendly.

Important Notes

The code computes ESP fitted partial atomic charges from a tabulated ESP on grid points that are read from a Gaussian cube file. ALL VARIABLES in the cube file should be in ATOMIC UNITS. You have to make sure that the values within the cube file make sense – i.e have the right units, cell and grid symmetry, format, etc, which can be a bit tricky. Please be aware some periodic DFT codes include other terms in the stored electrostatic potential file. For example, in the popular code VASP, the exchange-correlation potential is added to the ESP in the LOCPOT and one needs to insure that the proper keywords are used to not include the XC potential. For VASP, we have written a Python code to convert the LOCPOT into a cube file. Conversion files are found in the *Conversion* directory.

Compiling the Code

There is a README file that describes how to compile the code in the *Source* directory.

Input files

To run REPEAT, one minimally needs a '*repeat.input*' file which defines the settings for the calculation and a *cube* file that contains the electrostatic potential on grid points. The name of the cube file is defined in the *repeat.input* file. If one imposes symmetry, then a *symmetry.input* file is required. If one imposes RESP-like harmonic restraints, then a *RESP_parameters.input* file is required, while if QEq restraints are used, then a *QEq_parameters.input* file is required. If any one of these files is missing a sample input will be created with the expected file name and the job will stop. The details of these input files are given below:

repeat.input:

This is the main input file that defines all of the main settings for the REPEAT calculation. Upon execution, the program looks for a *repeat.input* file in the running directory and will exit if no file can be found. Below is an example of input file which is quite self-explanatory:

Sample *repeat.input* file:

```
    Input ESP file name in cube format
a   h2o.cube
    Fit molecular(0) or periodic(1:default) system?
b   1
    van der Waals scaling factor (default = 1.0)
c   0.90
    Apply RESP-like harmonic restraints?, no(0:default), yes(1)
d   0
    Read cutoff radius? no(0), yes(1:default)
e   1
    If flag above=1 provide R_cutoff next (in Bohrs)
f   20.00
    Apply symmetry constraints? no(0:default), yes(1)
g   0
    Use QEq restraints? no(0:default), yes(1)
h   0
    If flag above=1 then provide weight next
i   0.00
    van der Waals max scaling factor
j   10.00
    System total charge
k   0.00
```

a. Name of cube file: This simply defines the name of the cube file in the current working directory the REPEAT executable is run in.

b. Molecular or periodic: In a periodic calculation the ESP from the periodic images of the charges is calculated using an Ewald summation.

c. Van der Waals scaling factor: The REPEAT methodology considers as "valid" grid points those lying inside the simulation box and outside the van der Waals atomic spheres. The van der Waal's radii for each element are taken from the UFF force field and are hard coded into the executable. This factor allows one to globally scale these radii. We have found that a value of 0.90 to work well to give reasonable charges while reducing the buried atom problem found with ESP fitted charge methods.

d. RESP restraints: Although ESP fitted charges typically give chemically intuitive charges, there are cases when the best fit charge is unusually large or of the 'wrong' sign. This most often occurs when the atom is 'buried' and has little to no exposed surface area such that there are no fitting points close to it. In these cases, one can change the charge on the atom with minimal effect on the ESP at any of the fitting grid points. Thus, usual charges can often give the best fit. Although this isn't a problem within the formalism when fitting a single species, these unusual charges are unsavory and not particularly transferable from one system to the next. The latter is particularly important say if you are using the charges for machine learning.

To alleviate this problem one can restrain the charges as first introduced in the RESP ESP charge method. We have implemented quadratic restraints of the type $a(q_i-q_i^o)^2$ where 'a' is the strength of the restraint and q^0 is the equilibrium charge value or minimum for the quadratic restraint. In the original RESP paper, they found that centering the restraint at zero worked best, but this probably hasn't been explored thoroughly.

To use the RESP restraints, one needs a file: *RESP_parameters.input* that contains 'a' and 'q^o' values for each element in atomic units as shown below:

```
H  0.050000 0.000000
He 0.050000 0.000000
Li 0.100000 0.000000
…
```

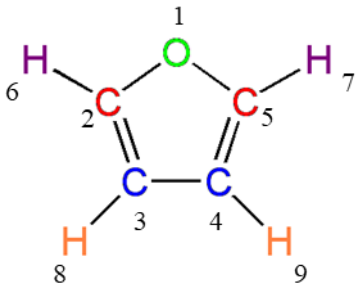
The second column is the 'a' value, while the third column is 'q^o'. It is important to note that the code does not read the element name and assumes that the first row is element 1 or H, and the 15th row is element 15 or P. If you request RESP restraints but do not have the *RESP_parameters.input* file, one will be created for you with zeros for all parameters and the job will stop. You can then modify this file with the parameters desired and rerun the REPEAT calculation.

e. Cutoff radius flag: This variable defines the cutoff to be used within the real space series of the Ewald sum. If one does not define a cutoff radius, then the default value of 0.5*Volume$^{1/3}$ is used which is only good for cubic cells. If one specifies a custom radius, then it is read in the next input line.

f. Cutoff radius: This is the cutoff radius described in (**e**) above in Bohrs.

g. Symmetry Constraints: In the unrestrained-all atoms case, the charges on the atoms are treated as individual variables to be fitted. However, if some of the charges are expected to be identical (due to symmetry) there is the possibility constraining the atoms to have identical charges. More formally, the charge on these 'equivalent' atoms are linked to the same fitting variable whether they are symmetry equivalent or not.

To apply symmetry, the user must designate which atoms are symmetry equivalent in the file: *symmetry.input*. It is important to note that the which atoms are equivalent is NOT determined from the structure. If no *symmetry.input* file is found, the code will generate an sample file for the H_2O molecule, and stop. To better understand the format used in the *symmetry.input*, here is a simple example for the molecule shown. The first line of the file is the total number of independent atoms in the structure (i.e. the number of trees). Then, for each tree there are two lines. In the first one defines the total number of atoms in the tree. The second one includes the integer indices of all identical atoms in the tree as they appear within the cube file.



symmetry.input

```
5
# first tree
1
1
# second tree
2
2 5
# third tree
2
3 4
# fourth tree
2
6 7
# fifth tree
2
8 9
```

QEq restraints: The QEq method is a parameterized method for rapidly generating charges. Instead of using harmonic restraints to tame buried atom charges, one can introduce QEq-like restraints as introduced in the original REPEAT paper. When using QEq restraints, the parameters are read from the *QEq_parameters.input* file. This file defines the χ_i^0 and $J_{ii}/2$ parameters for each element in atomic units from the original QEq paper of Rappe and Goddard (*J. Phys. Chem.* **1991**, *95*, 3358). The *QEq_parameters.input* file for the first few elements in shown below:

```
H  0.166397 0.477146
He 0.449802 0.899603
Li 0.110466 0.175364
…
```

The second column is the 'χ^0' value or electronegativity, while the third column is '$J_{ii}/2$' value, or hardness. It is important to note that the code does not read the element name and assumes that the first row is element 1 or H, and the 15th row is element 15 or P. If you request QEq restraints but do not have the *QEq_parameters.input* file, one will be created for you with default values for all parameters and the job will stop. You can then modify this file with the parameters desired and rerun the REPEAT calculation. The default values were taken from the Materials Studio package.

h. Weighting factor QEq restraints: This is the weighting factor or strength of the restraints. We haven't explored the use of the QEq restraints so one will have to experiment to see what values are appropriate.

i. van der Waals max scaling factor: The valid grid points used for fitting will only extend out by this factor of the van der Waals radii of each atom. To reduce the number of grid points and reduce the memory requirements for REPEAT, one can reduce this. A value of less than 2 is not recommended.

j. Net charge of system: This is the net charge of the framework. The charges are fit with this constraint imposed. Note that some periodic DFT codes allow one to perform calculations where the periodic system has a net charge by using a constant background charge that neutralizes it. Many MOFs have net framework charges and therefore one can find REPEAT charges where the counter ions are removed. See our paper *J Chem. Theory and Comp.*, **2017**, *13*, 2858–2869 (DOI:10.1021/acs.jctc.6b00998) for more details.

Cube file

The REPEAT code reads the electrostatic potential that is tabulated in the Gaussian cube file format. Here is a quick description of the format of a *.cube* file. Google it for extra info.

The first two lines are comments. Then follows the total number of atoms and the origin of the ESP mesh. The next three lines give the number of grid spaces in each direction followed by the corresponding reduced-lattice vectors in real space. The "true" lattice vectors result from multiplying the reduced ones by the factors in the first column (# of grid spaces) of the same row.
Then there is the atomic data (one row per atom) with five numbers being: 1) the atom number 2) the atom number again (the second value sometimes is not the atomic # but it doesn't matter because the code does not use it). 3-5) the three spatial atomic coordinates. Following the atomic data, one finds the volumetric data per grid point with the z index being the fastest index moving.

It is important to note that REPEAT expects all data in the cube file to be in atomic units.

Example of *.cube* file for water

```
# two first lines are comments
# specific to the software that produced the file
    3    0.000000    0.000000    0.000000
   41    0.460908    0.000000    0.000000
   41    0.000000    0.460908    0.000000
   41    0.000000    0.000000    0.460908
    1    1.000000    9.448629    9.448629    9.448629
    1    1.000000   11.177540   11.785843    9.448629
    8    8.000000    9.409701   11.285065    9.448629
  1.92244E-04  1.96419E-04  2.08900E-04  2.29554E-04  2.58149E-04  2.95014E-04
  3.38657E-04  3.88903E-04  4.44959E-04  5.05834E-04  5.70316E-04  6.36973E-04
  7.02425E-04  7.67967E-04  8.30260E-04  8.87323E-04  9.37216E-04  9.78149E-04
  1.00859E-03  1.02734E-03  1.03368E-03  1.02734E-03  1.00859E-03  9.78149E-04
  9.37216E-04  8.87322E-04  8.30259E-04  7.67966E-04  7.02424E-04  6.36972E-04
  5.70314E-04  5.05832E-04  4.44958E-04  3.88901E-04  3.38656E-04  2.95013E-04
  5.70314E-04  5.05832E-04  4.44958E-04  3.88901E-04  3.38656E-04
  3.19538E-04  3.24175E-04  3.38029E-04  3.60928E-04  3.92574E-04  4.32530E-04
  ...
          41*41*41 grid resolution, arranged in groups of six columns
```

OUTPUT FILES

Aside from the main output which is printed to the standard output (stdout), there are two output files that REPEAT creates: *Coul_ESP.dat* and *grid_ESP.dat*. The main output is self-explanatory and will not be elaborated upon here.

Coul_ESP.dat: This file contains the Cartesian coordinates (in Bohr) of each valid grids point followed by QM electrostatic potential (with the 'zero' shifted) and the ESP due to the fitted point charges at that grid point.

grid_ESP.dat: This file contains the coordinates (in Bohr) of all grid points followed by QM electrostatic potential as read from the cube file (not shifted).