# USER'S GUIDE

# Chargemol program for performing DDEC atomic population analysis

# Version 3.5

September 26, 2017

Please use the most recent version of this program by checking for updates at

http://sourceforge.net/projects/ddec/

This manual was prepared by

**Thomas A. Manz**

Please sign up for the ddec-news mailing list on sourceforge.net. This is our only way to directly contact you when a new version of the program has been released through sourceforge.

Click here to go to webpage for signing up.

**CONTENTS:**

## PART A: General Program Information

### A1) Introduction and Overview:

The density derived electrostatic and chemical (DDEC) method partitions the electron and spin densities to compute net atomic charges (NACs), atomic spin moments (ASMs), and bond orders (BOs). The DDEC NACs are simultaneously optimized to reproduce the chemical states of atoms and the electrostatic potential, $V(\vec{r})$, outside the material's electron distribution. DDEC charges are well-suited for studying the chemical properties of materials and for constructing force-fields used in atomistic simulations. The ASMs are simultaneously optimized to reproduce atomic chemical states and the spin contribution to the magnetic field, $\vec{B}^{spin}(\vec{r})$, outside the material's electron distribution. The methodology for computing BOs is highly accurate both for individual BOs as well as for the sum of bond orders (SBOs) for each atom. The DDEC method has been designed for accurate analysis of a variety of materials including: (a) molecular systems, (b) porous and non-porous crystalline solids, (c) solid surfaces, (d) nanoclusters, nanotubes, sheets, and other nanostructures. The method can accurately treat covalent and ionic materials. The principal advantages of the DDEC method are its high accuracy, applicability to a wide range of materials, sound theoretical basis, and basis set independence. Its computational cost scales approximately linearly as the number of atoms in the unit cell increases.

This program is written in the Fortran programming language. OpenMP parallelization allows it to run on multiple processors sharing a common memory; that is, on multiple processors in a single compute node. It does not parallelize across different compute nodes. It can also be compiled and ran as a serial (i.e., one-processor) program by turning off the OpenMP parallelization. Compilation is accomplished using a recent version (≥4.6.2) of the GNU GFortran compiler. Other compilers have not been tested and should not be used at this time.

**The current code version can compute two variants of charge partitioning: DDEC6 (recommended) and DDEC/c3 (aka DDEC3). The DDEC6 method is a more recent variant that improves results in extremely dense solids with a smaller impact on molecules. In addition to improved accuracy, DDEC6 is more computationally efficient (faster) than DDEC3. The choice of DDEC6 or DDEC3 is specified in the job_control.txt file (<charge type> DDEC6 </charge type> or <charge type> DDEC3 </charge type>). If nothing is specified, the program defaults to DDEC6 partitioning.**

More information can be found in

**The DDEC6 charge partitioning method is described in:** (1) T. A. Manz and N. Gabaldon Limas, "Introducing DDEC6 atomic population analysis: part 1. Charge partitioning theory and methodology," *RSC Adv.*, 6 (2016) 47771-47801. (2) N. Gabaldon Limas and T. A. Manz, "Introducing DDEC6 atomic population analysis: part 2. Computed results for a wide range of periodic and nonperiodic materials," *RSC Adv.*, 6 (2016) 45727-45747.

**This code computes bond orders using the method described in:** T. A. Manz, "Introducing DDEC6 atomic population analysis: part 3. Comprehensive method to compute bond orders," *RSC Adv.*, 7 (2017) 45552-45581 (open access).

**This code uses the spin partitioning method described in:** T. A. Manz and D. S. Sholl, "Methods for Computing Accurate Atomic Spin Moments for Collinear and Noncollinear Magnetism in Periodic and Nonperiodic Materials," *J. Chem. Theory Comput.* 7 (2011) 4146-4164.

**The DDEC3 charge partitioning method is described in:** T. A. Manz and D. S. Sholl, "Improved Atoms-in-Molecule Charge Partitioning Functional for Simultaneously Reproducing the Electrostatic Potential and Chemical States in Periodic and Non-Periodic Materials," *J. Chem. Theory Comput.* 8 (2012) 2844-2867.

**The first generations of DDEC analysis were described in:** T. A. Manz and D. S. Sholl, "Chemically Meaningful Atomic Charges that Reproduce the Electrostatic Potential in Periodic and Nonperiodic Materials," *J. Chem. Theory Comput.* 6 (2010) 2455-2468.

Although the code accepts a variety of input file types (i.e., VASP, wfx, cube, and xsf), it has been **most often used with VASP and .wfx files**. For periodic systems, a computationally efficient and accurate strategy is to use the projector augmented wave (PAW) method in VASP. To date, the code has been used in this manner to analyze >1000 periodic materials with unit cells ranging in size from one atom to more than 8000 atoms. Both the VASP and DDEC codes are extremely reliable, so you can reasonably expect this approach to perform flawlessly and yield great results. In principle, SIESTA output can be converted to an xsf file and analyzed using the Chargemol program; however, this conversion is somewhat tedious and less often used. For nonperiodic (e.g., molecular) systems, the preferred strategy is to generate a .wfx file using the latest Gaussian 09 release. The .wfx file was specifically designed to contain the appropriate information for performing atom-in-molecule population analysis.[1] The .wfx file contains the Baussian basis set specifications, nuclear positions, total net charge, basis set coefficients for the natural orbitals, natural orbital

[1] http://aim.tkgristmill.com/wfxformat.html

occupancies, and core electron densities for pseudo-potentials. In my experience, essentially all of the problems encountered when using a .wfx file have to do with it being generated improperly. The reliability of generating the .wfx file has improved between Gaussian 09 releases, so it is best to use the latest Gaussian 09 release for generating this file. For other types of quantum chemistry software (e.g., GPAW, CP2K, and ONETEP), the Chargemol program can accept input in the form of cube files containing the valence and spin densities.

**A2) Summary of the code's capabilities:**

1. Computes DDEC net atomic charges, dipoles, and quadrupoles. Also computes the effective decay exponent for each atom's electron density tail.
2. Computes DDEC atomic spin moments for collinear and non-collinear magnetic systems.
3. Computes DDEC bond orders, the sum of bond orders (SBO) for each atom, and the overlap populations.
4. Reads VASP files, wfx files, cube files, and xsf files. This allows compatibility with a number of different computational chemistry programs including but not limited to VASP, Gaussian, SIESTA, GPAW, CP2K, and ONETEP.
5. Compatible with electron density files produced from any quantum chemistry method including but not limited to density functional theory, coupled cluster theory, configuration interaction methods, and multi-reference methods like CAS-SCF.
6. Compatible with different strategies for treating core electrons during the *ab initio* calculation: (a) all-electron relaxed core, (b) all-electron frozen core (e.g., projector augmented wave method), and (c) effective core potential or pseudo-potential. In case (c), the Chargemol program includes the core electron density replaced by the pseudo-potential using core electron reference densities.
7. Performs numerous integration checks to make sure all electrons are properly accounted for. It makes sure the number of valence and core electrons integrate to the expected values, and it makes sure the grid spacing is adequate.
8. A complete set of reference densities is now available for all elements up to atomic number 109 (i.e., H to Mt).
9. Computes the relative root mean squared error (RRMSE) between the electrostatic potential of a point charge model and the electrostatic potential $V(\vec{r})$ of the *ab initio* electron density, $\rho(\vec{r})$. To compute the RRMSE, $V(\vec{r})$ must be in a Gaussian-style cube file (non-periodic system) or a VASP LOCPOT file (periodic system).
10. Computes the relative mean absolute error (RMAE) between the magnetic field of an atomic spin moment model and the spin-derived magnetic field, $\vec{B}^{spin}(\vec{r})$, of the *ab initio* spin density.
11. For non-collinear magnetic systems, two descriptors, $\Xi^{mag}$ and $\Xi^{dir}$, are automatically computed to assess the appropriateness of the exchange-correlation functional. More information on these descriptors can be found in the publication listed in section A above.
12. Computes the $\left\langle (r_A)^2 \right\rangle$, $\left\langle (r_A)^3 \right\rangle$, and $\left\langle (r_A)^4 \right\rangle$ radial moments of each atom in a material.

**A3) Version Information:**

The main differences between versions 3.5 and 3.4.x are:

1. The method for computing bond orders has been extensively reworked to make it faster and simpler. The code now uses the published method described in T. A. Manz, "Introducing DDEC6 atomic population analysis: part 3. Comprehensive method to compute bond orders," *RSC Adv.*, 7 (2017) 45552-45581 (open access).
2. In addition to the $\left\langle \left(r_A\right)^3 \right\rangle$ moments, the program now computes the $\left\langle \left(r_A\right)^2 \right\rangle$ and $\left\langle \left(r_A\right)^4 \right\rangle$ radial moments for each atom. These radial moments are computed using numerical integration. In addition, for single atom systems with .wfx file inputs, the exact (i.e., analytic) values of these moments are also printed (if the input file contains no higher than g-type basis functions).
3. The atomic radial moments files are now printed in a format that can be read using the Jmol program.
4. The program memory requirements were decreased by removing an extra large array from the spin partitioning modules. The names of some variables and arrays have also been updated to make them consistent with the published articles.
5. The integration_tolerance_percent was increased to 0.1.
6. In the job_control.txt file, the option of whether to compute bond orders is now specified using the tag <compute BOs> instead of the tag <compute EBOs>.
7. The reference data for polarizability to volume ratios have been updated to include the most recent data and additional chemical elements.
8. The core iterator has been updated to include a step at the beginning that limits the number of core electrons assigned to a single pixel to ≤ 30 electrons before core partitioning and core grid correction.
9. When reading VASP input files, the program now looks for not-a-name (NAN, ±infinity) entries (as well as entries with an absolute value greater than $10^{12}$) in the density grid files and terminates with an error if any are found.
10. A bug in the reading of spin densities from .xsf files was fixed.
11. A bug in the processing of wfx files using periodic boundary conditions was fixed.
12. Some minor typos and spelling mistakes were corrected.

Version 3.4 was first released on December 31, 2015. The main differences between versions 3.4.x and 3.2 are:

1. The computational efficiency of DDEC6 partitioning has been significantly improved. In general the DDEC6 partitioning is both more accurate and more computationally efficient than DDEC3 partitioning.
2. Support for reading the VASP CHGCAR has been added. Because the CHGCAR includes double-precision numbers as opposed to the single-precision numbers in the CHG file, it is now preferable to use the CHGCAR file instead of the CHG file as input. (This is especially important for a small number of systems where the electron density overflowed the single-precision printing width in the CHG file leading to program termination. This problem does not occur when using the CHGCAR file.)
3. Some of the OpenMP parallel routines contained racing conditions that led to program errors. This has been fixed.
4. Other minor program updates and bug fixes.
5. Minor 3.4.x version updates: Version 3.4.1 (released on March 2, 2016) includes the minor fix of allowing the number of core electrons for each element to manually changed in the job_control.txt file when running the Fortran version. The version released on May 18, 2016 includes an update to the journal references. Version 3.4.4 (released on June 26, 2016) includes updates to (a) read spin density information from valence-only cube files such as those generated by CP2K program, (b) compute and print overlap populations (see section D6), and

(c) changed the default behavior from DDEC3 to DDEC6. Other minor program updates and bug fixes. In version 3.4.5 (released on October 27, 2016), the reading of Gaussian cube files was improved to eliminate the need to have the number of grid points along the third lattice vector evenly divisible by six. The allocation and deallocation of large arrays has been optimized to reduce the overall memory requirements.

6. In version 3.4.5 (released on October 27, 2016), we added an option to choose whether to read cube file inputs in units of bohr or angstrom. For reading the input geometry and density from the cube file in bohr, no action is necessary (default behavior). For reading the input geometry and density from the cube file in angstroms, add the following lines to the job_control.txt file:

```
<density format>
e_per_angs3  <-- specifies that units in the cube file are in angstroms rather than in bohr
</density format>
```

**A4) Licensing information:**

This software is distributed under the license described below, which is compliant with the Open Source Initiative found at http://www.opensource.org/docs/osd. You are not licensed to use or modify this software unless you agree to all of the licensing terms below:

1. Derived Works and Integrity of The Author's Source Code: Derived works must be distributed under exactly the same terms as this license. **The source code is restricted from being distributed in modified form; however, this license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time.**

2. Free Redistribution: The license does not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license does not require a royalty or other fee for such a sale. This license does not restrict other software that may be distributed along with the licensed software.

3. Source Code: The program includes source code, and allows distribution in source code as well as compiled form. (Deliberately obfuscated source code and intermediate forms such as the output of a preprocessor or translator are not allowed.)

4. Distribution of License: The rights attached to the program apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

5. Scientific publications that use computational results of the program are required to properly cite the original research papers describing the methods used in the program, according to best scientific practice.

   *Note:* For the current software version, this would include citation of the following papers:

   (a) **for the calculation of DDEC6 atomic charges**, please cite: (1) T. A. Manz and N. Gabaldon Limas, "Introducing DDEC6 atomic population analysis: part 1. Charge partitioning theory and methodology," *RSC Adv.*, 6 (2016) 47771-47801. (2) N. Gabaldon Limas and T. A. Manz, "Introducing DDEC6 atomic population analysis: part 2. Computed results for a wide range of periodic and nonperiodic materials," *RSC Adv.*, 6 (2016) 45727-45747. If desired, you can also cite references listed below for the DDEC3 and earlier methods.

   (b) **for the calculation of DDEC3 atomic charges**, please cite: (i) T. A. Manz and D. S. Sholl, "Improved Atoms-in-Molecule Charge Partitioning Functional for Simultaneously Reproducing the Electrostatic

Potential and Chemical States in Periodic and Non-Periodic Materials," *J. Chem. Theory Comput.* 8 (2012) 2844-2867. (ii) T. A. Manz and D. S. Sholl, "Chemically Meaningful Atomic Charges that Reproduce the Electrostatic Potential in Periodic and Nonperiodic Materials," *J. Chem. Theory Comput.* 6 (2010) 2455-2468.

(c) **for the calculation of RMSE and RRMSE**, please cite: (i) T. A. Manz and D. S. Sholl, "Improved Atoms-in-Molecule Charge Partitioning Functional for Simultaneously Reproducing the Electrostatic Potential and Chemical States in Periodic and Non-Periodic Materials," *J. Chem. Theory Comput.* 8 (2012) 2844-2867. (ii) T. Watanabe, T. A. Manz, and D. S. Sholl, "Accurate Treatment of Electrostatics during Molecular Adsorption in Nanoporous Crystals without Assigning Point Charges to Framework Atoms," *J. Phys. Chem. C*, 115 (2011) 4824 – 4836.

(d) **for the calculation of atomic spin moments**, please cite: T. A. Manz and D. S. Sholl, "Methods for Computing Accurate Atomic Spin Moments for Collinear and Noncollinear Magnetism in Periodic and Nonperiodic Materials," *J. Chem. Theory Comput.* 7 (2011) 4146-4164.

(e) **for the calculation of bond orders**, please cite: T. A. Manz, "Introducing DDEC6 atomic population analysis: part 3. Comprehensive method to compute bond orders," *RSC Adv.*, 7 (2017) 45552-45581.

6. In addition to the scientific papers described in # 5 above, please cite this software program as following: T. A. Manz and N. Gabaldon Limas, Chargemol program for performing DDEC analysis, Version 3.5, 2017, ddec.sourceforge.net.
7. This license makes no discrimination against persons or groups or fields of endeavor. The rights attached to this program do not depend on the program's being part of a particular software distribution. No provision of this license is predicated on any particular style of interface.

## PART B: Using Quantum Chemistry Programs to Generate Density Files for Analysis

### B1) VASP density grid generation:

1) Set up a POSCAR file with the final set of atomic positions: Before generating the density files used by the Chargemol program, you must first have the final set of atomic positions. **If you want to optimize the atomic positions in the system you are studying, you must do this prior to attempting to generate the density files used by the Chargemol program. Once you have the final set of atomic positions, place them in a POSCAR file in the usual manner.**

2) Set up the INCAR file to compute the needed density files: Specify NSW = 0 (this tells VASP not to change the geometry) and LCHARG = .TRUE. and LAECHG = .TRUE. in the INCAR file (this tells VASP to generate the needed electron density files). **We want to re-iterate that NSW must be set to zero in this calculation.** Specify PREC = Accurate in the INCAR file to generate an appropriate grid spacing for calculating the DDEC charges. The other parameters in the INCAR file are specific to your particular system and should be specified as described in the VASP manual.

3) Set up the KPOINTS file: In the KPOINTS file, you need to specify the number of k-points along each of the three lattice directions. We recommend that the number of k-points along a lattice direction times the length of the corresponding lattice vector be approximately 16 Å or greater. For example, if the lengths of the lattice vectors for your unit cell are 3 Å, 9 Å, and 23 Å, you might want to use 6 k-points along the first lattice vector, 2 k-points

along the second lattice vector, and 1 k-point along the third lattice vector. You can specify more k-points if you wish, but this probably will not change the values of the computed charges.

4) Set up the POTCAR file: The POTCAR file is set up in the usual manner using projector augmented wave (PAW) data. For the current version, you cannot use ultrasoft pseudopotentials to generate density files needed for the Chargemol program. This is not a limitation of the Chargemol program. Rather, I've consciously chosen not to support the less reliable pseudo-potential method for codes that implement the more accurate PAW method.

5) Run the VASP program to produce the AECCAR0, AECCAR2, and CHGCAR (or CHG) files, along with other files. The AECCAR0 file holds the core electron density while the AECCAR2 file holds the valence electron density. The CHGCAR (or CHG) file holds the valence pseudo-density and the spin densities (for spin-polarized calculations). **AECCAR0, AECCAR2, CHGCAR (or CHG), and POTCAR are the only files needed for the subsequent charge analysis.** You can use either the CHGCAR or CHG file, but if you have both available, the CHGCAR is preferable, because it contains double-precision numbers as opposed to the single-precision numbers in the CHG file.

**B2) Gaussian 09 wfx file generation:**

Generate the .wfx file using Gaussian 09 Revision B.01 or later. (The .wfx file cannot be generated using GAUSSIAN 03 or earlier versions.) Set up and run your Gaussian job in the usual manner. To generate the .wfx file, include the following keywords in route line of the Gaussian 09 input file:

for spin restricted calculations: density = current output = wfx pop=NO
for spin polarized calculations: density = current output = wfx pop=NOAB
(For recent Gaussian 09 versions, you can simply specify density = current output = wfx and Gaussian 09 will appropriately choose pop=NO or NOAB.)

The name of the .wfx file must be specified at the bottom of the Gaussian 09 input file. Examples of Gaussian 09 input files for generating the .wfx file are found in /examples_to_run/GAUSSIAN_WFX_ozone_triplet_CCSD/ and /examples_to_run/GAUSSIAN_WFX_ozone_singlet_PW91/ .

**B3) GPAW cube file generation:**

Generate a cube file named total_density.cube with a grid spacing of ~0.07 Å (0.14 bohr) along each lattice direction. This file should contain the total electron density, not the valence density. The dimensions of the cube file must correspond to the unit cell. To compute the atomic spin moments for spin-polarized (magnetic) systems, you will also need to generate a similar cube file named spin_density.cube that contains the spin density.

**B4) Siesta xsf file generation:**

1) For the computation of net atomic charges, basis sets of DZP (double zeta with polarization) or higher quality (e.g., TZP or TZDP) are recommended. Basis sets generated without a confinement potential may experience an abrupt change in value at the cutoff radius which spoils the accuracy of geometry optimizations. Basis sets prepared using a confinement potential are strongly recommended, because these basis sets decay smoothly to zero at the cutoff radius producing accurate forces and geometries. For this reason, all Siesta calculations should use basis sets generated with a confinement potential.

2) A MeshCutoff value of 300 Ry is recommended. Lower MeshCutoff values give grids that are too coarse while higher MeshCutoff values give finer grids than necessary. According to tests we performed, FilterCutoff does not need to be specified in the Siesta input file to get accurate results when using MeshCutoff = 300 Ry.

3) Include the keyword SaveRho = .true. in the Siesta input file and run the Siesta calculation in the usual manner to generate the SystemLabel.RHO file that contains the valence charge density.

4) Download the Sies2xsf program written by Andrei Postnikov from http://www.home.uni-osnabrueck.de/apostnik/download.html. (Sies2xsf converts SystemLabel.RHO into an .xsf formatted file readable by the XCrySDens and Chargemol programs). After downloading and uncompressing Sies2xsf, open the file Rho2xsf and uncomment the lines shown below by removing the C character at the line's beginning. This turns on the option for writing a periodic unit cell to the .xsf file.

(Remove C from each line shown below.)

```
C       write (6,702)
C 101   write (6,703,advance="no")
C       read (5,*) labbox
C       if (labbox.eq.'Y'.or.labbox.eq.'y') then
C         write (io1,'(a7)') 'CRYSTAL'
C         write (io1,'(a7)') 'PRIMVEC'
C         write (io1,201) (rbox(ii,1),ii=1,3)
C         write (io1,201) (rbox(ii,2),ii=1,3)
C         write (io1,201) (rbox(ii,3),ii=1,3)
C         write (io1,'(a9)') 'PRIMCOORD'
C         write (io1,'(i5,i2)') nbox,1
C       else if (labbox.eq.'N'.or.labbox.eq.'n') then
          write (io1,'(a5)') 'ATOMS'
C       else
C         write (6,*) "I do not understand:"
C         goto 101
C       endif
```

Do not uncomment any other lines in the Rho2xsf file!

After uncommenting these lines, compile the Sies2xsf program. If you have problems installing or running Sies2xsf, please contact Andrei Postnikov or ask your question to the Siesta mailing list. I am unable to answer technical questions about the Sies2xsf program.

5) Run the Sies2xsf program and answer questions indicated. The Sies2xsf program will ask for an origin, grid vectors, and number of points along each grid vector. The Chargemol program requires at least one grid point per 0.2 bohr (0.1 Å), so it is wise to choose enough points to give a spacing of 0.1 to 0.15 bohr along each grid vector. For periodic systems, the grid vectors should be chosen equal to the unit cell's lattice vectors with the origin chosen as (0.0, 0.0, 0.0). For non-periodic systems, the origin and grid vectors should be chosen to place the molecule in the center of cell, so that the distance between the box's surface and each atom is at least 4 Å. IMPORTANT: The Sies2xsf program will indicate the number of atoms found to be located inside the box. If this number is the same as the number of atoms in your system, then you're all set. If this number is less than the number of atoms in your system, then choose a very tiny increase in the box's size by choosing the origin to be (-0.00000001 , -0.00000001 , -0.00000001 ) or something similar until you get the correct number of atoms in the box. (This may be necessary, for example, if the nucleus for one atom is located precisely at the box's edge so

that a tiny difference in the grid vector will place the atom either inside or outside the box.) IMPORTANT: The difference between the grid vectors and the lattice vectors must be smaller than 0.000001 bohr or else the Chargemol program will complain. The Sies2xsf program will ask for a 3D property to add. Type RHO and then hit enter. The Sies2xsf will say it has added the density (RHO) to the xsf file. If available, the spin density is automatically included in the RHO information, so you don't need to add any other 3D properties. When Sies2xsf asks if you wish to add yet another 3D property, hit enter to indicate you are finished adding the desired properties.

6) Use an editor to open the .xsf generated by Sies2xsf. If your system is non-periodic the first line of the file should say ATOMS. If your system is periodic, the first line of the file should say CRYSTAL and the second line should say PRIMVEC. You cannot run a charge calculation for a periodic system using the non-periodic ATOMS notation. Therefore, if your system is supposed to be periodic but the first line of the .xsf file says ATOMS, you will need to regenerate the .xsf file. To produce proper .xsf files for periodic systems, please see the instructions above for uncommenting the appropriate lines of rho2xsf and recompiling Sies2xsf.

**B5) CP2K and ONETEP cube file generation:**

Generate a gaussian-style cube file named valence_density.cube with a grid spacing of 0.1 Å (0.2 bohr) or finer along each lattice direction. The dimensions of the cube file must correspond to the unit cell. To compute the atomic spin moments for spin-polarized (magnetic) systems, you will also need to generate a similar cube file named spin_density.cube that contains the spin density.

## PART C: Installing and Running the Fortran Program

**C1) The job_control.txt file:**

The job_control.txt file controls the job options when running the Fortran program. It must be placed in the same directory as your density files. For Windows systems, the atomic densities directory should use slashes like this: `G:\chargemol\uploaded_versions\chargemol_09_26_2017\atomic_densities\`
For Linux systems, the atomic densities directory should use slashes like this:
`/home/tamanz/bin/atomic_densities/`
Notice how the directory slashes point in different directions for the Windows and Linux systems. Also, notice that there is a slash at the end of the directory name. Examples job_control.txt files are given below:

==========Example for GAUSSIAN files==========

```
<atomic densities directory complete path>
/home/tamanz/bin/atomic_densities/   <--- specifies where the atomic densities are located
</atomic densities directory complete path>

<input filename>
ozone+1_PW91.wfx  <--- specifies the density input file
</input filename>

<charge type>
DDEC6 <-- specifies the charge type (DDEC3 or DDEC6)
</charge type>

<compute BOs>
.true. <-- specifies whether to compute bond orders or not
</compute BOs>


(For .wfx input files, there is no need to specify the periodicity or the net charge.)
```

==========Example for VASP files==========

```
<net charge>
0.0 <-- specifies the net charge of the unit cell (defaults to 0.0 if nothing specified)
</net charge>

<periodicity along A, B, and C vectors>
.true. <--- specifies whether the first direction is periodic
.true. <--- specifies whether the second direction is periodic
.true. <--- specifies whether the third direction is periodic
</periodicity along A, B, and C vectors>

<atomic densities directory complete path>
/home/tamanz/bin/atomic_densities/     <--- specifies where the atomic densities are located
</atomic densities directory complete path>

<charge type>
DDEC6 <-- specifies the charge type (DDEC3 or DDEC6)
</charge type>

<compute BOs>
.true. <-- specifies whether to compute bond orders or not
</compute BOs>
```

(For VASP files, there is no need to specify an input file. Specifying the periodicity is optional, because it defaults to .true. for VASP files.)

==========Example for SIESTA files==========

```
<net charge>
0.0 <-- specifies the net charge of the unit cell (defaults to 0.0 if nothing specified)
</net charge>

<periodicity along A, B, and C vectors>
.true.  <--- specifies whether the first direction is periodic
.true. <--- specifies whether the second direction is periodic
.true. <--- specifies whether the third direction is periodic
</periodicity along A, B, and C vectors>

<atomic densities directory complete path>
/home/tamanz/bin/atomic_densities/   <--- specifies where the atomic densities are located
</atomic densities directory complete path>

<input filename>
chabazite.XSF     <--- specifies the density input file
</input filename>

<charge type>
DDEC6 <-- specifies the charge type (DDEC3 or DDEC6)
</charge type>

<compute BOs>
.true. <-- specifies whether to compute bond orders or not
</compute BOs>

<number of core electrons>
36 28    <-- First number is the element number and second number is num_core
47 30    <-- First number is the element number and second number is num_core
</number of core electrons>
```

==========Example for CP2K files==========

```
<net charge>
0.0 <-- specifies the net charge of the unit cell (defaults to 0.0 if nothing specified)
</net charge>

<periodicity along A, B, and C vectors>
.true.  <--- specifies whether the first direction is periodic
.true. <--- specifies whether the second direction is periodic
.true. <--- specifies whether the third direction is periodic
</periodicity along A, B, and C vectors>

<atomic densities directory complete path>
/home/tamanz/bin/atomic_densities/   <--- specifies where the atomic densities are located
</atomic densities directory complete path>

<charge type>
DDEC6 <-- specifies the charge type (DDEC3 or DDEC6)
</charge type>

<compute BOs>
.true. <-- specifies whether to compute bond orders or not
</compute BOs>

<number of core electrons>
36 28    <-- First number is the element number and second number is num_core
47 30    <-- First number is the element number and second number is num_core
</number of core electrons>

(For CP2K files, there is no need to specify an input file.)
```

==========Example for GPAW files==========

```
<net charge>
0.0 <-- specifies the net charge of the unit cell (defaults to 0.0 if nothing specified)
</net charge>

<periodicity along A, B, and C vectors>
.true.  <--- specifies whether the first direction is periodic
.true. <--- specifies whether the second direction is periodic
.true. <--- specifies whether the third direction is periodic
</periodicity along A, B, and C vectors>

<atomic densities directory complete path>
/home/tamanz/bin/atomic_densities/   <--- specifies where the atomic densities are located
</atomic densities directory complete path>

<charge type>
DDEC6 <-- specifies the charge type (DDEC3 or DDEC6)
</charge type>

<compute BOs>
.true. <-- specifies whether to compute bond orders or not
</compute BOs>

(For GPAW files, there is no need to specify an input file.)
```

==========Example for ONETEP files==========

```
<net charge>
0.0 <-- specifies the net charge of the unit cell (defaults to 0.0 if nothing specified)
</net charge>

<periodicity along A, B, and C vectors>
.true.  <--- specifies whether the first direction is periodic
.true. <--- specifies whether the second direction is periodic
.true. <--- specifies whether the third direction is periodic
</periodicity along A, B, and C vectors>
```

```
<atomic densities directory complete path>
/home/tamanz/bin/atomic_densities/   <--- specifies where the atomic densities are located
</atomic densities directory complete path>

<charge type>
DDEC6 <-- specifies the charge type (DDEC3 or DDEC6)
</charge type>

<compute BOs>
.true. <-- specifies whether to compute bond orders or not
</compute BOs>

<number of core electrons>
36 28    <-- First number is the element number and second number is num_core
47 30    <-- First number is the element number and second number is num_core
</number of core electrons>

<density format>
e_per_angs3  <-- specifies that units in the cube file are in angstroms rather than in bohr
</density format>
```

**Note the specification of units in that last example. If you have cube file generated by CP2K, ONETEP, GPAW, GAUSSIAN, etc. that is in angstrom and e/angstrom$^3$ then you have to add the above three lines to the job_control.txt file. Omit these if your cube file is in atomic units (i.e., bohr and e/bohr$^3$).**

**C2) Installing and running the program on Windows operating systems:**

**Installation:** Two sets of precompiled binaries are provided. The one set uses a command-line window and asks for the directory containing the job_control.txt file to be entered as a text string; this version can be useful for Windows scripting programs that automate the passing of the text string to the command-line window. The other set opens a graphical user interface (GUI) and asks the user to browse to the directory and select the job_control.txt file; this version is more convenient for a human. The only difference between the two versions is the manner in which the folder containing the job_control.txt file is selected; all other aspects of the programs are identical. The following serial and parallel precompiled binaries are provided for 32-bit and 64-bit Windows operating systems:
Chargemol_09_26_2017_windows_32bits_parallel_command_line.exe,
Chargemol_09_26_2017_windows_32bits_serial_command_line.exe,
Chargemol_09_26_2017_windows_64bits_parallel_command_line.exe,
Chargemol_09_26_2017_windows_64bits_serial_command_line.exe,
Chargemol_09_26_2017_windows_32bits_parallel_GUI.exe,
Chargemol_09_26_2017_windows_32bits_serial_GUI.exe,
Chargemol_09_26_2017_windows_64bits_parallel_GUI.exe,
Chargemol_09_26_2017_windows_64bits_serial_GUI.exe .
To install the program on a Windows computer, simply copy one or more of the above executable files to any folder on your computer. For the parallel codes, you will also need to copy the OpenMP dynamically linked library for Windows into the same directory as the program: pthreadGC2-32.dll for 32 bit codes and pthreadGC2-64.dll for 64 bit codes. Computers running the 64-bit Windows operating system can run both the 32-bit and 64-bit codes, with the 64-bit codes running slightly faster than the 32-bit codes. Computers running the 32-bit Windows operating system can run only the 32-bit code and cannot run the 64-bit code. If you are unsure whether your computer is running a 32-bit or 64-bit version of the Windows operating system, choose the 32-bit codes. The serial code creates a single thread and runs on a single processor. The parallel code creates many parallel threads and runs on as many processors as your computer has available.

**Running the program:** Double click on the executable to start the Chargemol program. If desired, you can place a shortcut to the executable on your desktop. Double-clicking the shortcut will also start the program. Once the program starts, it will open a window asking you to select the job_control.txt file. If using the GUI, browse to the directory your job files are in, select the job_control.txt file and hit the "open file" button. If using the command line interface, enter the directory name as a string (either by typing manually or by passing via a scripting program). The program will run and produce the output files. When the program has finished running, it will automatically close the command window.

**Compatibility**: Both the serial and parallel Fortran executables work on Windows XP, 7, 8, and 10.

**FAQS:** 1) *What if I don't have a Fortran compiler installed on my Windows computer?* **Answer:** No problem. Use the precompiled executable files listed above.

2) *What if I want to compile the program from sourcecode on my Windows computer?* **Answer:** There is probably no need to compile the program, because the precompiled executables are provided. However, you can compile the program using Simply Fortran version 2 (http://simplyfortran.com/download/). When compiling the program on a Windows machine, make sure to set `run_interactive=.TRUE.` in the `module_global_parameters.f08` file.

3) *What if my Windows computer already has a different Fortran compiler installed on it?* **Answer:** There is probably no need to compile the program, because the precompiled executables are provided. However, you can compile the program using Simply Fortran version 2 (http://simplyfortran.com/download/). We believe SimplyFortran2 can be installed without interfering with other previously installed Fortran compilers, but to be sure you should check with the compiler vendors first.

4) *What if I want to compile the program from sourcecode using a different Fortran compiler on my Windows computer?* **Answer:** If you encounter problems or questions about compiling the program's sourcecode on Windows machines using a compiler besides SimplyFortran2, since we do not have expertise in this area, please send the question to the CCL.net mailing list. See http://www.ccl.net/ for instructions on how to join the Computational Chemistry List (CCL).

**C3) Installing and running the program on Linux operating systems:**

First try this: The following precompiled binaries are provided for the Linux operating system: Chargemol_09_26_2017_linux_parallel and Chargemol_09_26_2017_linux_serial. There is a decent probability that you can use these executables without having to recompile the program. Make sure these files are transferred to your Linux system as type **binary** using your file transfer program—transferring them as type **text** (or **ASCII**) will corrupt them and make them unusable. We recommend placing these files in the /bin/ subdirectory of your home directory, which you can create if it does not exist; however, you can place them in any directory you choose. Make sure the file permissions are set as 'executable' using the chmod command or by right-clicking on the file and setting its properties.

Compiling the program: If the above does not work, you can compile the program using the Gfortran compiler. Gfortran is the name of the GNU Fortran compiler, which is part of the GNU Compiler Collection, also known as GCC. You can find more information about it at http://gcc.gnu.org/wiki/GFortran . You should be able to compile it using a recent version of the GFortran compiler. To compile the program, first make sure the GFortran compiler is installed and loaded. On Linux systems using modules, type module avail to see the list of modules available. Look for something called gnu, gfortran, or gcc. Then load this module by typing module load gnu, module load gcc, or a similar command to load the module. Once GFortran is installed and loaded, change to the

/chargemol_FORTRAN_09_26_2017/sourcecode/ directory and set the permissions on the compile_serial.txt and compile_parallel.txt files to type 'executable'. When compiling the program for Linux, make sure to set `run_interactive=.FALSE.` in the `module_global_parameters.f08 file`. Navigate to the /chargemol_FORTRAN_09_26_2017/sourcecode/ directory and type ./compile_serial.txt at the Linux command prompt to compile the serial version. Navigate to the /chargemol_FORTRAN_09_26_2017/sourcecode/ directory and type ./compile_parallel.txt at the Linux command prompt to compile the parallel version. After compiling, we recommend moving the binary executable files (e.g., `Chargemol_09_26_2017_linux_serial` or `Chargemol_09_26_2017_linux_parallel`) to your ~/bin/ directory.

**Running the program:** To run the program on Linux clusters, we recommend submitting jobs to a batch queuing system. (It is also possible to run the program directly from the command prompt.) Please see the /example_job_submission_scripts/Fortran_linux/ directory for example job submission scripts. The details of the script will depend on the particulars of the batch queuing system for the computational cluster you're using. Please consult with an experienced colleague using the same computational cluster if you have questions about the format of the script file.

**FAQS:** 1) *Can the program be compiled using Intel, PGI, or other Fortran compilers?* **Answer:** We have not tested compilation of the program on other Linux Fortran compilers, such as Intel, PGI, etc. In the future, we may test compilation on these, but at the present time we cannot recommend these for compiling the program. We recommend using a recent version of the GFortran compiler to compile the program.

2) *What level of optimization should be used when compiling the program?* **Answer:** We do not recommend using any compiler optimizations when compiling the program.

3) *What kinds of flags/options should be set when compiling the program?* **Answer:** As shown in the compile_serial.txt and compile_parallel.txt files, libraries should be statically linked using `-static-libgfortran`. As shown in the compile_parallel.txt file, OpenMP should be enabled using `-fopenmp`. We do not recommend any other compiler flags at this time.

**C4) Installing and running the program on Unix operating systems:**

We have not compiled the program on Unix operating systems. However, compiling on a Unix system is similar to compiling on a Linux system. See instructions for Linux systems, above.

**C5) Installing and running the program on Apple/Macintosh operating systems:**

We have not compiled or run the program on Apple/Mac Operating Systems, but you can find information about how to install GFortran on the Mac OS at the following link: http://web.mit.edu/mfloyd/www/computing/mac/gfortran/. Once GFortran is installed, you will need to copy the sourcecode files from /chargemol_FORTRAN_09_26_2017/sourcecode/ to your computer and then compile the program.

**C6) Operating instructions for VASP users:**

Place the following files into one directory: AECCAR0, AECCAR2, CHGCAR (or CHG), POTCAR, job_control.txt, and your submission script file (if submitting the job to a batch queue). Please see section **B1** for instructions on generating the AECCAR0, AECCAR2, CHGCAR (or CHG), and POTCAR files. Please see section **C1** for instructions on how to set up the job_control.txt file.

Linux users: At the Linux command prompt, cd to the directory your job is in. Then submit the script file using the command qsub scriptfile.sh (for PBS queuing system), msub scriptfile.sh (for Moab queuing system), or sbatch scriptfile.sh (for SLURM queuing system), etc. The job will run and produce the output files. If you want to run the job from the command prompt instead of submitting it to a batch queue, then use the following commands:

```
OMP_NUM_THREADS=12          (specify the number of parallel threads)
export OMP_NUM_THREADS
~/bin/Chargemol_09_26_2017_linux_parallel (or whatever the path to the executable is)
```

Note: Normally the number of parallel threads should be set equal to the number of processors on a single compute node. You can use fewer threads if you want to use only some of the available processors, but specifying more threads than available processors makes no sense. For example, if you have a node with 4 processing cores, you would set OMP_NUM_THREADs = 4. If you have a node with 8 processing cores, you would set OMP_NUM_THREADs = 8.

For serial execution from the command prompt, simply type the following:

```
~/bin/Chargemol_09_26_2017_linux_serial (or whatever the path to the executable is)
```

Windows users: You do not need a submission script file. Double click on the executable or its shortcut (see section **C2**) to start the Chargemol program. Once the program starts, it will open a window asking you to select the directory containing the job_control.txt file. If using the GUI, browse to the directory your job files are in, select the job_control.txt file and hit the "open file" button. If using the command line interface, enter the directory name as a string (either by typing manually or by passing via a scripting program). The program will run and produce the output files. When the program has finished running, it will automatically close the command window.

**C7) Operating instructions for Gaussian 09 users:**

Place the following files into one directory: wfx file, job_control.txt, and your submission script file (if submitting the job to a batch queue). Please see section **B2** for instructions on generating the wfx file. Please see section **C1** for instructions on how to set up the job_control.txt file.

Linux users: At the Linux command prompt, cd to the directory your job is in. Then submit the script file using the command qsub scriptfile.sh (for PBS queuing system), msub scriptfile.sh (for Moab queuing system), or sbatch scriptfile.sh (for SLURM queuing system), etc. The job will run and produce the output files. If you want to run the job from the command prompt instead of submitting it to a batch queue, then use the following commands:

```
OMP_NUM_THREADS=12          (specify the number of parallel threads)
export OMP_NUM_THREADS
~/bin/Chargemol_09_26_2017_linux_parallel (or whatever the path to the executable is)
```

Note: Normally the number of parallel threads should be set equal to the number of processors on a single compute node. You can use fewer threads if you want to use only some of the available processors, but specifying more threads than available processors makes no sense. For example, if you have a node with 4 processing cores, you would set OMP_NUM_THREADs = 4. If you have a node with 8 processing cores, you would set OMP_NUM_THREADs = 8.

For serial execution from the command prompt, simply type the following:

```
~/bin/Chargemol_09_26_2017_linux_serial (or whatever the path to the executable is)
```

Windows users: You do not need a submission script file. Double click on the executable or its shortcut (see section **C2**) to start the Chargemol program. Once the program starts, it will open a window asking you to select the directory containing the job_control.txt file. If using the GUI, browse to the directory your job files are in, select the job_control.txt file and hit the "open file" button. If using the command line interface, enter the directory name as a string (either by typing manually or by passing via a scripting program). The program will run and produce the output files. When the program has finished running, it will automatically close the command window.

**C8) Operating instructions for GPAW users:**

Place the following files into one directory: total_density.cube, job_control.txt, spin_density.cube (for spin-polarized systems), and your submission script file (if submitting the job to a batch queue). Please see section **B3** for instructions on generating the cube files. Please see section **C1** for instructions on how to set up the job_control.txt file.

Linux users: At the Linux command prompt, cd to the directory your job is in. Then submit the script file using the command qsub scriptfile.sh (for PBS queuing system), msub scriptfile.sh (for Moab queuing system), or sbatch scriptfile.sh (for SLURM queuing system), etc. The job will run and produce the output files. If you want to run the job from the command prompt instead of submitting it to a batch queue, then use the following commands:

```
OMP_NUM_THREADS=12          (specify the number of parallel threads)
export OMP_NUM_THREADS
~/bin/Chargemol_09_26_2017_linux_parallel (or whatever the path to the executable is)
```

Note: Normally the number of parallel threads should be set equal to the number of processors on a single compute node. You can use fewer threads if you want to use only some of the available processors, but specifying more threads than available processors makes no sense. For example, if you have a node with 4 processing cores, you would set OMP_NUM_THREADs = 4. If you have a node with 8 processing cores, you would set OMP_NUM_THREADs = 8.

For serial execution from the command prompt, simply type the following:

```
~/bin/Chargemol_09_26_2017_linux_serial (or whatever the path to the executable is)
```

Windows users: You do not need a submission script file. Double click on the executable or its shortcut (see section **C2**) to start the Chargemol program. Once the program starts, it will open a window asking you to select the directory containing the job_control.txt file. If using the GUI, browse to the directory your job files are in, select the job_control.txt file and hit the "open file" button. If using the command line interface, enter the directory name as a string (either by typing manually or by passing via a scripting program). The program will run and produce the output files. When the program has finished running, it will automatically close the command window.

**C9) Operating instructions for Siesta users:**

Place the following files into one directory: xfs file, job_control.txt, and your submission script file (if submitting the job to a batch queue). Please see section **B4** for instructions on generating the xsf file. Please see section **C1** for instructions on how to set up the job_control.txt file.

To run properly the Chargemol program must know the number of core electrons for each element. By default, these values are set equal to largest noble gas core. For example, Ru (atomic number 44) would have num_core(44) = 36 by default since Kr (atomic number 36) is the largest noble gas with atomic number less than 44. As a second example, Kr corresponds to a default value num_core(36) = 18, since Ar (atomic number 18) is

the noble gas appearing just before Kr. For example, a pseudo-potential for Ag (atomic number 47) might use a core of 28 electrons ([Ar] + the 3d electrons), 36 electrons (the default), or 30 electrons ([Ar] + the 3d electrons + 4s electrons), etc. The number of core electrons for each element can be changed by placing an appropriate entry in the job_control.txt file using the following format:

```
<number of core electrons>
36 28    <-- First number is the element number and second number is num_core.
47 30    <-- First number is the element number and second number is num_core.
</number of core electrons>

The above example uses 28 core electrons for Kr and 30 core electrons for Ag atoms.
```

Use the following steps to determine the number of core electrons from a Siesta output file:

(a) Open the Siesta output file and locate the following line for each element used in the calculation:

Vna: chval, zval: 4.000 4.000

The atomic number minus zval is the number of core electrons. The element for this example was Si (atomic number 14) with zval = 4, so num_core(14) = 14 – 4 = 10. Since this is the default value used in the Chargemol program (for Si the largest noble gas core is [Ar] with ten electrons), nothing needs to be done. If instead zval was 10.00, then num_core(14) = 4 would need to be specified in the job_control.txt file.

(b) For a given pseudo-potential, you only need to do step (a) once for each element. Moreover, it doesn't hurt to have num_core specified for extra elements in the job_control.txt file. If you repetitively use the same pseudo-potentials, you can create a job_control.txt file with num_core values for all elements you commonly study and re-use this same job_control.txt file to compute the net charges for your different Siesta jobs.

(c) If you mess up step (a), the Chargemol program will probably catch your mistake and terminate with a warning that the number of valence electrons does not sum correctly. If the program doesn't complain *and* the computed net atomic charges are reasonable, then you can be sure the number of core electrons was correctly specified.

Linux users: At the Linux command prompt, cd to the directory your job is in. Then submit the script file using the command qsub scriptfile.sh (for PBS queuing system), msub scriptfile.sh (for Moab queuing system), or sbatch scriptfile.sh (for SLURM queuing system), etc. The job will run and produce the output files. If you want to run the job from the command prompt instead of submitting it to a batch queue, then use the following commands:

```
OMP_NUM_THREADS=12          (specify the number of parallel threads)
export OMP_NUM_THREADS
~/bin/Chargemol_09_26_2017_linux_parallel (or whatever the path to the executable is)
```

Note: Normally the number of parallel threads should be set equal to the number of processors on a single compute node. You can use fewer threads if you want to use only some of the available processors, but specifying more threads than available processors makes no sense. For example, if you have a node with 4 processing cores, you would set OMP_NUM_THREADs = 4. If you have a node with 8 processing cores, you would set OMP_NUM_THREADs = 8.

For serial execution from the command prompt, simply type the following:

```
~/bin/Chargemol_09_26_2017_linux_serial (or whatever the path to the executable is)
```

Windows users: You do not need a submission script file. Double click on the executable or its shortcut (see section **C2**) to start the Chargemol program. Once the program starts, it will open a window asking you to select the directory containing the job_control.txt file. If using the GUI, browse to the directory your job files are in, select the job_control.txt file and hit the "open file" button. If using the command line interface, enter the directory name as a string (either by typing manually or by passing via a scripting program). The program will run and produce the output files. When the program has finished running, it will automatically close the command window.

**C10) Operating instructions for CP2K and ONETEP users:**

Place the following files into one directory: valence_density.cube, job_control.txt, spin_density.cube (for spin-polarized systems), and your submission script file (if submitting the job to a batch queue). Please see section **B5** for instructions on generating the cube files. Please see section **C1** for instructions on how to set up the job_control.txt file.

To run properly the Chargemol program must know the number of core electrons for each element. By default, these values are set equal to largest noble gas core. For example, Ru (atomic number 44) would have num_core(44) = 36 by default since Kr (atomic number 36) is the largest noble gas with atomic number less than 44. As a second example, Kr corresponds to a default value num_core(36) = 18, since Ar (atomic number 18) is the noble gas appearing just before Kr. For example, a pseudo-potential for Ag (atomic number 47) might use a core of 28 electrons ([Ar] + the 3d electrons), 36 electrons (the default), or 30 electrons ([Ar] + the 3d electrons + 4s electrons), etc. The number of core electrons for each element can be changed by placing an appropriate entry in the job_control.txt file using the following format:

```
<number of core electrons>
36 28    <-- First number is the element number and second number is num_core.
47 30    <-- First number is the element number and second number is num_core.
</number of core electrons>

The above example uses 28 core electrons for Kr and 30 core electrons for Ag atoms.
```

Linux users: At the Linux command prompt, cd to the directory your job is in. Then submit the script file using the command qsub scriptfile.sh (for PBS queuing system), msub scriptfile.sh (for Moab queuing system), or sbatch scriptfile.sh (for SLURM queuing system), etc. The job will run and produce the output files. If you want to run the job from the command prompt instead of submitting it to a batch queue, then use the following commands:

```
OMP_NUM_THREADS=12          (specify the number of parallel threads)
export OMP_NUM_THREADS
~/bin/Chargemol_09_26_2017_linux_parallel (or whatever the path to the executable is)
```

Note: Normally the number of parallel threads should be set equal to the number of processors on a single compute node. You can use fewer threads if you want to use only some of the available processors, but specifying more threads than available processors makes no sense. For example, if you have a node with 4 processing cores, you would set OMP_NUM_THREADs = 4. If you have a node with 8 processing cores, you would set OMP_NUM_THREADs = 8.

For serial execution from the command prompt, simply type the following:

```
~/bin/Chargemol_09_26_2017_linux_serial (or whatever the path to the executable is)
```

Windows users: You do not need a submission script file. Double click on the executable or its shortcut (see section **C2**) to start the Chargemol program. Once the program starts, it will open a window asking you to select the directory containing the job_control.txt file. If using the GUI, browse to the directory your job files are in, select the job_control.txt file and hit the "open file" button. If using the command line interface, enter the directory name as a string (either by typing manually or by passing via a scripting program). The program will run and produce the output files. When the program has finished running, it will automatically close the command window.

**C11) What to do if you encounter a problem:**

First, make sure you carefully read and follow the above instructions. If this does not resolve your problem, try the following: If the problem is about installing compilers or the use of compilers ***except*** GFortran or SimplyFortran2, since we do not have expertise in this area, please send the question to the CCL.net mailing list. See http://www.ccl.net/ for instructions on how to join the Computational Chemistry List (CCL). If the question is about compiling the program using GFortran or SimplyFortran2 or about running the program or interpreting the results, please send the question to thomasamanz{(at)}gmail^,com. (Email disguised to prevent spambots from harvesting it.)

## PART D: Understanding the Program Output

### D1) DDEC_even_tempered_net_atomic_charges.xyz file:

The DDEC_even_tempered_net_atomic_charges.xyz file contains the atomic coordinates and partial atomic charges in a format that can be read and displayed by the Jmol program. An example is shown below:

```
    3
Nonperiodic system <- Indicates the system is nonperiodic (otherwise the unit cell information is displayed on this line).
O      0.000000    -0.000000     0.582650     0.215330
O      0.000000     1.029479    -0.291325    -0.107665
O      0.000000    -1.029479    -0.291325    -0.107665

 Chargemol Version 3.4.4 June 26, 2016. <- Displays the version information.
See ddec.sourceforge.net for latest version.

Computational parameters:
Number of radial integration shells =    100
Cutoff radius (pm) =    500
Error in the integrated total number of electrons before renormalization (e) =    6.7953E-11
Charge convergence tolerance =    1.0000E-05
Minimum radius for electron cloud penetration fitting (pm) =    200
Minimum decay exponent for electron density of buried atom tails =   1.7500
Maximum decay exponent for electron density of buried atom tails =  2.5000
Number of iterations to convergence =      7

The following XYZ coordinates are in angstroms. The atomic dipoles and quadrupoles are in
atomic units.
```

```
atom number, atomic symbol, x, y, z, net_charge, dipole_x, dipole_y, dipole_z, dipole_mag,
Qxy, Qxz, Qyz,Q(x^2-y^2), Q(3z^2 - R^2), three eigenvalues of traceless quadrupole moment
tensor
    1 O      0.000000    -0.000000     0.582650     0.215330     0.000000    -0.000000    -
0.214687     0.214687    -0.000000    -0.000000    -0.000000    -0.253359    -0.163831    -
0.093946    -0.054786     0.148732
    2 O      0.000000     1.029479    -0.291325    -0.107665    -0.000000    -0.045530
0.076323     0.088872    -0.000000    -0.000000    -0.105973    -0.344922     0.033472    -
0.172599    -0.040587     0.213186
    3 O      0.000000    -1.029479    -0.291325    -0.107665    -0.000000     0.045530
0.076323     0.088872     0.000000    -0.000000     0.105973    -0.344922     0.033472    -
0.172599    -0.040587     0.213186

The sperically averaged electron density of each atom fit to a function of the form exp(a -
br) for r >=rmin_cloud_penetration
atom number,atomic symbol, x, y, z, a, b, Rsquared where a and b are in atomic units and
Rsquared is the squared correlation coefficient
    1 O      0.000000    -0.000000     0.582650     0.406456     2.399834     0.997523
    2 O      0.000000     1.029479    -0.291325    -0.274317     2.164011     0.998514
    3 O      0.000000    -1.029479    -0.291325    -0.274346     2.164005     0.998514


 Since the system is non-periodic, the total dipole and quadrupole moments were computed
relative to the system center of mass.
 Multipole moments (in atomic units) using the net atomic charges:
 Partial_charge_dipole_x =    0.0000   Partial_charge_dipole_y =    0.0000
Partial_charge_dipole_z =    0.3556     Partial_charge_dipole_magnitude =    0.3556
 Partial_charge_quadrupole_xy =    0.0000   Partial_charge_quadrupole_xz =    0.0000
Partial_charge_quadrupole_yz =   -0.0000   Partial_charge_quadrupole_x2minusy2 =    0.8150
Partial_charge_quadrupole_3z2minusr2 =    1.2065
 Eigenvalues of traceless quadrupole moment tensor =   -0.6086   0.2064   0.4022
 Multipole moments (in atomic units) using the net atomic charges and atomic dipoles:
 Dipole_x =   -0.0000   Dipole_y =    0.0000   Dipole_z =    0.2936   Dipole_magnitude =
0.2936
 Quadrupole_xy =   -0.0000   Quadrupole_xz =    0.0000   Quadrupole_yz =   -0.0000
Quadrupole_x2minusy2 =    1.1693   Quadrupole_3z2minusr2 =    0.2792
 Eigenvalues of traceless quadrupole moment tensor =   -0.6312   0.0931   0.5381
 Multipole moments (in atomic units) using the net atomic charges, atomic dipoles, and atomic
quadrupoles:
 Dipole_x =   -0.0000   Dipole_y =    0.0000   Dipole_z =    0.2936   Dipole_magnitude =
0.2936
 Full_quadrupole_xy =   -0.0000   Full_quadrupole_xz =    0.0000   Full_quadrupole_yz =   -
0.0000   Full_quadrupole_x2minusy2 =    0.2261   Full_quadrupole_3z2minusr2 =    0.1823
 Eigenvalues of traceless quadrupole moment tensor =   -0.1434   0.0608   0.0827
```

The lines after the second line list information for each of the atoms. The first column is the atomic symbol. Columns two, three, and four contain the X, Y, and Z coordinates, respectively. The net atomic charges (shown in red above) are found in the fifth column. Additional output including the atomic dipoles and quadrupoles, and a fit of the valence density of each atom to a spherically symmetric exponentially decaying function, are displayed in a section below this in the DDEC_even_tempered_net_atomic_charges.xyz file. For nonperiodic systems, the total system dipole and quadrupole are displayed at the bottom for three different models: (a) net atomic charges only, (b) net atomic charges plus atomic dipoles, and (c) net atomic charges plus atomic dipoles plus atomic quadrupoles.

**D2) DDEC_even_tempered_atomic_spin_moments.xyz file:**

For magnetic systems, the atomic spin moments of each atom are contained in the file DDEC_even_tempered_atomic_spin_moments.xyz, which has the following format for a collinear magnetic system:

14

```
jmolscript: load "" {1 1 1} spacegroup "x,y,z" unitcell [{ 0.000000 4.200000 4.200000 }, {
4.200000 0.000000 4.200000 }, { 4.200000 4.200000 0.000000 }]
O    2.155335    2.154197    2.147179    0.051449
O    2.146704    4.142620    4.145560    0.051418
O    4.153493    2.157368    4.145509    0.051391
O    4.144774    4.145921    2.147338    0.051457
O    6.244665    6.245803    6.252821    0.051449
O    6.253296    4.257380    4.254440    0.051417
O    4.246507    6.242632    4.254491    0.051391
O    4.255226    4.254079    6.252662    0.051456
Fe   7.349954    7.349958    7.351852   -3.789384
Fe   1.050046    1.050042    1.048148   -3.789383
Fe   4.200000    4.200000    4.200000    3.815113
Fe   4.200000    2.100000    2.100000    3.815532
Fe   2.100000    4.200000    2.100000    3.815571
Fe   2.100000    2.100000    4.200000    3.815087

Collinear spin population analysis was performed
The total spin magnetic moment of the unit cell is    8.093964

See ddec.sourceforge.net for latest version.

Computational parameters:
spin_ref_fraction = 0.50
Number of radial integration shells = 100
Cutoff radius (pm) = 500
spin convergence tolerance = 0.000050
Number of iterations to convergence = 7

1    O    2.155335    2.154197    2.147179    0.051449
2    O    2.146704    4.142620    4.145560    0.051418
3    O    4.153493    2.157368    4.145509    0.051391
4    O    4.144774    4.145921    2.147338    0.051457
5    O    6.244665    6.245803    6.252821    0.051449
6    O    6.253296    4.257380    4.254440    0.051417
7    O    4.246507    6.242632    4.254491    0.051391
8    O    4.255226    4.254079    6.252662    0.051456
9    Fe   7.349954    7.349958    7.351852   -3.789384
10   Fe   1.050046    1.050042    1.048148   -3.789383
11   Fe   4.200000    4.200000    4.200000    3.815113
12   Fe   4.200000    2.100000    2.100000    3.815532
13   Fe   2.100000    4.200000    2.100000    3.815571
14   Fe   2.100000    2.100000    4.200000    3.815087

09-Jun-2014 19:41:35
```

The format of the DDEC_even_tempered_atomic_spin_moments.xyz file is similar to that of the DDEC_even_tempered_net_atomic_charges.xyz, with the atomic spin moments displayed in the fifth column. A positive value indicates spin up magnetism, and a negative value indicates spin down magnetism. The information at the bottom of the file is a repeat of the information at the top of the file, except each line starts with the atom number. This makes it easy for Jmol to read the top section and a human to read the bottom section.

For a non-collinear magnetic system, the DDEC_even_tempered_atomic_spin_moments.xyz file has the following format:

```
  2
jmolscript: load "" {1 1 1} spacegroup "x,y,z" unitcell [{ 10.000000 0.000000 0.000000 }, {
0.000000 10.000000 0.000000 }, { 0.000000 0.000000 10.000000 }]
Mg   5.000000   5.000000   5.000000   0.904210   0.521873   0.522044   0.522221
I    5.000000   5.000000   7.609670   0.053736   0.031018   0.031023   0.031033

Noncollinear spin population analysis was performed
```

```
The total spin magnetic moment vector of the unit cell is   0.552891   0.553066   0.553254

See ddec.sourceforge.net for latest version.

Computational parameters:
spin_ref_fraction = 0.50
Number of radial integration shells = 50
Cutoff radius (pm) = 300
convergence tolerance = 0.000050
Number of iterations to convergence = 7

       11-Jun-2011 18:59:58
```

This file is also readable by Jmol. In this case, the fifth column (e.g. 0.904210) contains the magnitude of the spin moment vector for each atom, while the last three columns contain the vector's x, y, z components. The spin moments are listed in as number of electrons. For example, a hydrogen atom whose electronic spin moment is aligned along the z-direction would have a spin moment 0.0 0.0 1.0.

**D3) DDEC_even_tempered_bond_orders.xyz file:**

The DDEC_even_tempered_bond_orders.xyz file contains the atomic coordinates and sum of bond orders (SBOs) in a format that can be read and displayed by the Jmol program. A portion of an example DDEC_even_tempered_bond_orders.xyz file is given below:

```
14
jmolscript: load "" {1 1 1} spacegroup "x,y,z" unitcell [{ 0.000000 4.200000 4.200000 }, {
4.200000 0.000000 4.200000 }, { 4.200000 4.200000 0.000000 }]
O    2.155335    2.154197    2.147179    2.224018 <-  Sum of bond orders (SBO) for each atom
O    2.146704    4.142620    4.145560    2.228984
O    4.153493    2.157368    4.145509    2.229258
O    4.144774    4.145921    2.147338    2.224376
O    6.244665    6.245803    6.252821    2.224009
O    6.253296    4.257380    4.254440    2.228974
O    4.246507    6.242632    4.254491    2.229249
O    4.255226    4.254079    6.252662    2.224366
Fe   7.349954    7.349958    7.351852    2.346497
Fe   1.050046    1.050042    1.048148    2.346475
Fe   4.200000    4.200000    4.200000    2.352900
Fe   4.200000    2.100000    2.100000    2.354956
Fe   2.100000    4.200000    2.100000    2.355312
Fe   2.100000    2.100000    4.200000    2.352013

See ddec.sourceforge.net for latest version.


The sum of bond orders (SBO) for each atom in the unit cell are listed above.
All bond orders greater than 0.0010 are printed below.


================================================================================
Printing BOs for ATOM # 1 ( O ) in the reference unit cell.
Bonded to the (-1, 0, 0) translated image of atom number 2 ( O ) with bond order = 0.030
Bonded to the (0, 0, 0) translated image of atom number 2 ( O ) with bond order = 0.054
Bonded to the (0, -1, 0) translated image of atom number 3 ( O ) with bond order = 0.031
Bonded to the (0, 0, 0) translated image of atom number 3 ( O ) with bond order = 0.053
Bonded to the (0, 0, -1) translated image of atom number 4 ( O ) with bond order = 0.030
Bonded to the (0, 0, 0) translated image of atom number 4 ( O ) with bond order = 0.054
Bonded to the (-1, -1, 0) translated image of atom number 5 ( O ) with bond order = 0.002
Bonded to the (-1, 0, -1) translated image of atom number 5 ( O ) with bond order = 0.002
Bonded to the (-1, 0, 0) translated image of atom number 5 ( O ) with bond order = 0.001
Bonded to the (0, -1, -1) translated image of atom number 5 ( O ) with bond order = 0.002
Bonded to the (0, -1, 0) translated image of atom number 5 ( O ) with bond order = 0.001
```

```
Bonded to the (0, 0, -1) translated image of atom number 5 ( O ) with bond order = 0.001
Bonded to the (0, -1, 0) translated image of atom number 6 ( O ) with bond order = 0.048
Bonded to the (0, 0, -1) translated image of atom number 6 ( O ) with bond order = 0.047
Bonded to the (-1, 0, 0) translated image of atom number 7 ( O ) with bond order = 0.049
Bonded to the (0, 0, -1) translated image of atom number 7 ( O ) with bond order = 0.046
Bonded to the (-1, 0, 0) translated image of atom number 8 ( O ) with bond order = 0.047
Bonded to the (0, -1, 0) translated image of atom number 8 ( O ) with bond order = 0.047
Bonded to the (-1, -1, 0) translated image of atom number 9 ( Fe ) with bond order = 0.009
Bonded to the (-1, 0, -1) translated image of atom number 9 ( Fe ) with bond order = 0.009
Bonded to the (0, -1, -1) translated image of atom number 9 ( Fe ) with bond order = 0.009
Bonded to the (0, 0, 0) translated image of atom number 10 ( Fe ) with bond order = 0.543
Bonded to the (-1, 0, 0) translated image of atom number 11 ( Fe ) with bond order = 0.005
Bonded to the (0, -1, 0) translated image of atom number 11 ( Fe ) with bond order = 0.005
Bonded to the (0, 0, -1) translated image of atom number 11 ( Fe ) with bond order = 0.005
Bonded to the (0, 0, 0) translated image of atom number 11 ( Fe ) with bond order = 0.005
Bonded to the (0, 0, 0) translated image of atom number 12 ( Fe ) with bond order = 0.363
Bonded to the (0, 0, 0) translated image of atom number 13 ( Fe ) with bond order = 0.362
Bonded to the (0, 0, 0) translated image of atom number 14 ( Fe ) with bond order = 0.359
The sum of bond orders for this atom is SBO =   2.224018
```

(Bond orders for the atoms 2 to 12 in the unit are listed here and have been deleted for display purposes.)

By opening this file in a text editor, it is possible to read out the bond order between any two atoms in the material. For example, the bond order between atom number 1 (oxygen) in the unit cell and the (-1, 0, 0) translated image of atom number 7 (oxygen) is bond order = 0.049.

After listing the bond order, each line contains a phrase of the following sort "`The average spin polarization of this bonding =     0.0000`" If all electrons comprising this bond are paired (i.e., spin unpolarized), its average spin polarization will be 0.0. If all electrons comprising the bond have only one spin direction (i.e. all spin up or all spin down), its average spin polarization will be 1.0. Numbers between 0.0 and 1.0 indicate a partial spin polarization in the bond. For conciseness, these phrases are not shown in the above output example.

**D4) Radial moments files:**

Version 3.5 produces DDEC_atomic_Rsquared_moments.xyz, DDEC_atomic_Rcubed_moments.xyz, and DDEC_atomic_Rfourth_moments.xyz files that are readable by Jmol or a text editor. These contain the $\left\langle (r_A)^2 \right\rangle$, $\left\langle (r_A)^3 \right\rangle$, and $\left\langle (r_A)^4 \right\rangle$ radial moments of each atom, respectively. An example DDEC_atomic_Rcubed_moments.xyz file is given below:

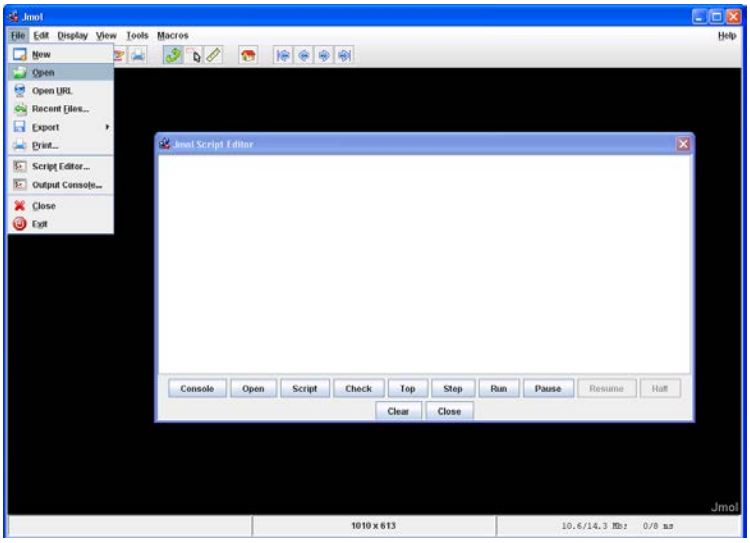
```
   3 <-- number of atoms
jmolscript: load "" {1 1 1} spacegroup "x,y,z" unitcell [{    15.000016    0.000000    0.000000 }, {    0.000000    15.000016
0.000000 }, {    0.000000    0.000000    15.000016 }] <-- unit cell information
O     0.000000    1.031228    1.125381   28.840451
H     0.000000    1.785070    0.512177    2.643166
H     0.000000    0.253722    0.542498    2.649899

   1 O     0.000000    1.031228    1.125381   28.840451
   2 H     0.000000    1.785070    0.512177    2.643166
   3 H     0.000000    0.253722    0.542498    2.649899
```

Each line contains the atomic symbol, X, Y, and Z Cartesian coordinates (in Å) of this atom, and the radial moment (e.g., $\left\langle (r_A)^3 \right\rangle$) of this atom in atomic units (e.g., in cubic bohrs). The second set of lines is identical to the first, except the second set lists the atom number to make it easier to locate a particular atom in a large system.

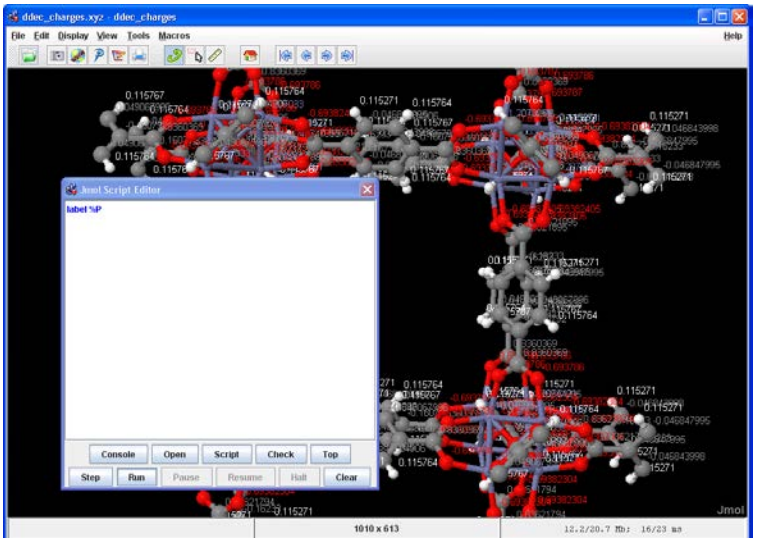
**D5) Using Jmol to visualize the output files:**

a) Download the latest version of Jmol from jmol.sourceforge.net
(Jmol version 12 or higher is required to load the periodic unit cell information.)

b) After starting Jmol, use the File/Open menu to browse to the directory containing the DDEC_even_tempered_net_atomic_charges.xyz file and open it. If the Jmol Script Editor is not open, you can open it using the File/Script Editor menu command.



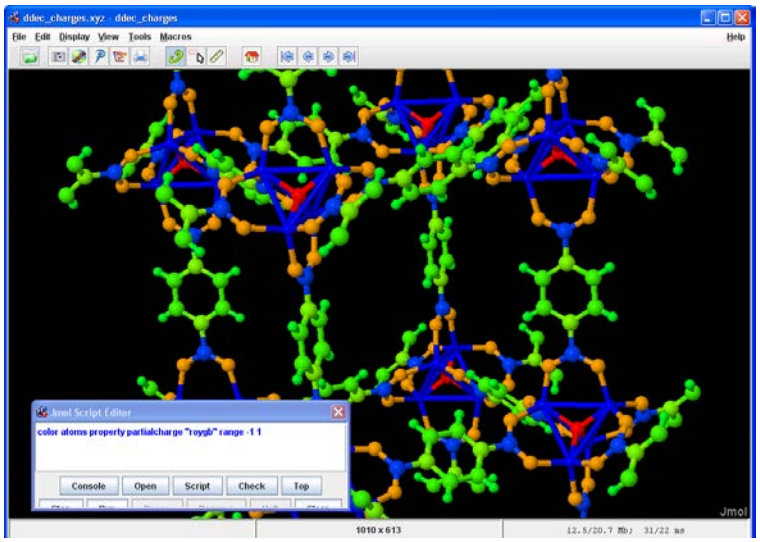

c) To view the net atomic charges next to each atom, type label %P in the Jmol Script Editor window and then hit Run:



d) Atoms can be colored according to charge by typing any one of the following commands in the Jmol Script Editor:

color partialcharge
color atoms property partialcharge
color atoms property partialcharge "rwb" range -1 1
color atoms property partialcharge "roygb" range -1 1

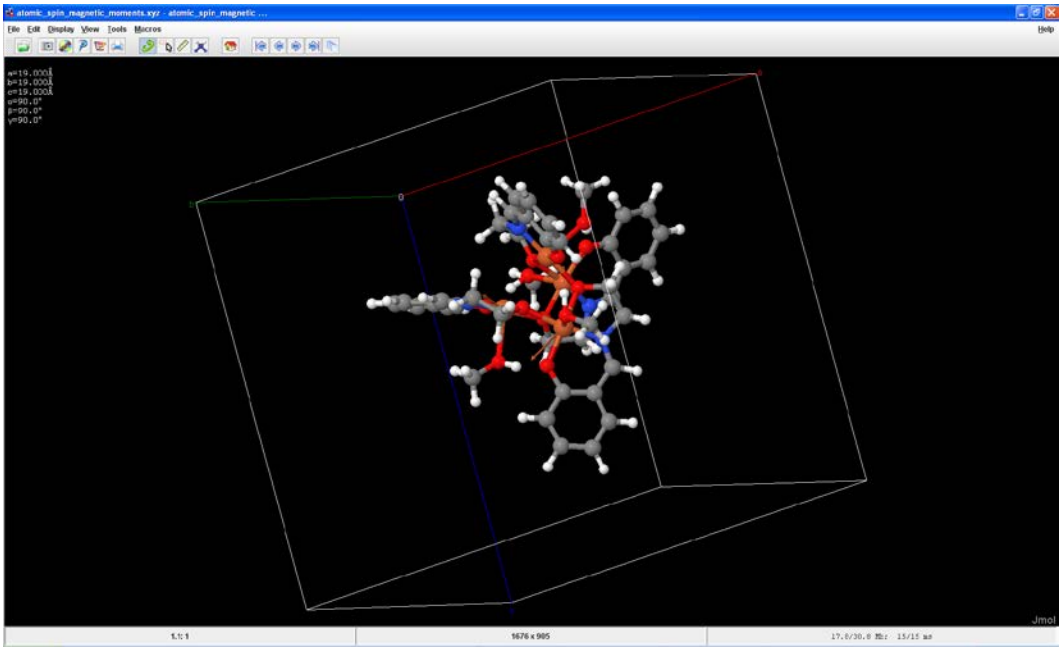
(You can change the range values to get different colors displayed.)




e) The sum of bond orders (SBO) for each atom can be visualized in the same manner as the net atomic charges, except the DDEC_even_tempered_bond_orders.xyz file must be opened. The Jmol commands label %P and color partialcharge display the numeric and color values of the SBOs, respectively.

f) For magnetic systems, the spin magnetic moments of each atom are visualized in the same way, except the DDEC_even_tempered_atomic_spin_moments.xyz file must be opened. The Jmol commands label %P and color partialcharge display the numeric and color values of the spin moments, respectively. For collinear systems, the value will be either positive (spin up) or negative (spin down).

For non-collinear systems, the scalar value is positive and indicates the magnitude of the spin vector. By default, Jmol animates the vectors as vibrations. Go to Tools > Vibrate > Stop vibration to make the atoms stop moving. Once this is done, the spin moment vectors will be automatically displayed for each atom. Below is an example with spin moment vectors displayed. (They are negligible on all atoms except the orange ones.)



**D6) Overlap populations:**

DDEC overlap populations quantify the extent to which the atomic electron distributions of two different atoms overlap. The overlap populations can be constructed with the first atom (i.e., atom A) in the reference unit cell and the second atom (i.e., atom j) anywhere in the material (either inside or outside the reference unit cell). We say atoms j and A are not equal if their nuclear positions are different. That is $j \neq A$ if and only if $\vec{R}_j \neq \vec{R}_A$ . Note that atom j can be a translated image of atom A or an untranslated or translated image of any other atom in the material.

$$OP_{A,j}\Big|_{j\neq A} = 2\oint \frac{\rho_A(\vec{r}_A)\rho_j(\vec{r}_j)}{\rho(\vec{r})} d^3\vec{r} = 2\oint \frac{w_A(r_A)w_j(r_j)}{W(\vec{r})^2}\rho(\vec{r})d^3\vec{r} \qquad (1)$$

For $j \neq A$ each of the non-negligible overlap populations are written to the overlap_populations.xyz file in the following format:

```
================================================================================
1472   <- Total number of overlaps printed
jmolscript: load "" {1 1 1} spacegroup "x,y,z" unitcell [{      8.100270    -0.010842
-0.036431 }, {     -0.010854     8.104335     0.009788 }, {     -0.036446     0.009777
8.104974 }] <- Information about the unit cell for periodic materials. For nonperiodic systems it simply says “Nonperiodic system”.
atom1, atom2, translation A, translation B, translation C, overlap population
     1     10      0     -1      0      0.0019878454
    10      1      0      1      0      0.0019878454
     1     11     -1      0      0      0.0019897240
    11      1      1      0      0      0.0019897240
     1     12     -1      0      0      0.0019152342
    12      1      1      0      0      0.0019152342
     1     13      0      0     -1      0.0019200320
    13      1      0      0      1      0.0019200320
     1     14      0      0     -1      0.0020746918
    14      1      0      0      1      0.0020746918
     1     15      0     -1      0      0.0020501590
    15      1      0      1      0      0.0020501590
     1     16      0      0      0      0.0019114005
```

```
 16      1      0      0      0      0.0019114005
  1     17      0      0      0      0.0020030998
 17      1      0      0      0      0.0020030998
  1     20      0      0      0      0.0020263115
 20      1      0      0      0      0.0020263115
  1     21      0      0      0      0.0019183236
```

(For conciseness, the remaining lines of this file were not reproduced here.)

The translation A, translation B, and translation C refer to the unit cell translations of the second atom along the first, second, and third lattice vectors, respectively, where the first atom is located in the reference unit cell. Note that each line of the file is reproduced with the order of the two atoms reversed. For example, "`1 10 0 -1 0 0.0019878454`" means the same as "`10 1 0 1 0 0.0019878454`". The purpose of this is to make it easy to copy the information into a spreadsheet (such as Microsoft Excel) and sort on the first column to get all overlap populations for a desired atom. If an overlap population for an atom pair is not listed, this means that it is negligible (i.e., approximately zero). Note that overlap populations for a given atom and its periodic images are listed, but the "overlap" population of an atom and its untranslated self is not listed.

**D7) Other program output:**

In addition to producing the above output files, the program produces a running summary of the calculation progress in the 'log' file. The log file is in text format and can be read by programs like Notepad, Wordpad, Notepad++, or other text viewers/editors. Our personal favorite is Notepad++, which can be freely downloaded from http://notepad-plus-plus.org/ . The name given to the 'log' file varies, and may be called something like VASP_DDEC_analysis.output or ozone_singlet_PW91.output, etc.

The top of the 'log' file displays whether the program ran in serial or parallel mode, and the number of parallel threads for the parallel mode. Various information about the progress of the calculation is printed throughout the 'log' file. The total time required for the calculation is printed at the bottom of the 'log' file.

In case the program terminates with an error, the 'log' file may indicate what went wrong and why. When using a batch submission script containing a line like

```
~/bin/Chargemol_09_26_2017_linux_parallel > debug.txt 2>&1
```

any errors detected by the operating system will be placed into a debug.txt file. In this case, you should look at both the 'log' and the debug.txt files to see what error was encountered.

## PART E: Additional Scripts

**E1) Calculating the accuracy of the net atomic charges for reproducing the electrostatic potential:**

The accuracy of a point charge model for reproducing the ab initio quantum mechanical (QM) electrostatic potential energy surface (EPES) can be quantified by the root-mean-square error (RMSE),

$$RMSE = \left\{\sum_i \left[V_i^q + V_{offset} - V_i^{QM}\right]^2 / N_{points}\right\}^{1/2}, \qquad (1)$$

where $V_i^q$ ($V_i^{QM}$) is the potential energy at grid point *i* due to the point charges (QM EPES), and $V_{offset}$ equals the average of $V_i^{QM} - V_i^q$ over the entire set of valid grid points. $V_{offset}$ is re-computed for each point charge model and for the case when all charges are set to zero. The sum in this expression is taken over the set of all valid grid points. An inner and outer van der Waals (vdW) multiplier are used to describe the volume of valid grid points. Points closer to any atom than the inner multiplier times its vdW radius are excluded. Likewise, points that are farther than the outer multiplier times the vdW radius of every atom are excluded. That is, valid grid points lie

between the surfaces described by inner multiplier x vdW radii and outer multipler x vdW radii. The relative root mean squared error (RRMSE) equals RMSE of the point charge model divided by RMSE when all charges are set to zero. Files for computing RMSE and RRMSE are found in the folder RRMSE_and_RMAE_scripts. To compute RMSE and RRMSE in a nonperiodic system, replace the potential.cube (Gaussian file) with that for your system, place the atomic charges and atomic dipoles in the partial_charges_data.m file, and type calculate_nonperiodic_RRMSE at the MATLAB prompt (or use the example script). To compute RMSE and RRMSE in a periodic system, replace the LOCPOT and POTCAR files with those for your system, make sure the VASP version is set correctly in the computational_parameters.m file, place the atomic charges and atomic dipoles in the partial_charges_data.m file, and type calculate_periodic_RRMSE at the MATLAB prompt (or use the example script). The values of the inner and outer multipliers are found in the computational_parameters.m file and can be changed if desired. The vdW_radii.m file contains the vdW radii, which can also be changed if desired.

**E2) Calculating the accuracy of the atomic spin moments for reproducing the magnetic field:**

The accuracy of an atomic spin moment (ASM) model for reproducing the magnetic field $\vec{B}^{spin}(\vec{r})$ can be quantified by the mean absolute error (MAE),

$$MAE = \sum_{\vec{p}} \left| \left( \vec{B}^{spin}(\vec{p})\Big|_{ASM} - \vec{B}^{spin}(\vec{p})\Big|_{spindensity} \right) \right| u(\vec{p}) / \sum_{\vec{p}} u(\vec{p})$$

where $\vec{B}^{spin}(\vec{p})\Big|_{ASM}$ and $\vec{B}^{spin}(\vec{p})\Big|_{spindensity}$ are the values at grid point $\vec{p}$ due to the ASM's and ab initio spin density, respectively. The finite element volume, $u(\vec{p})$, for each grid point makes the summations in the above equation equivalent to corresponding integrals over volume. $\vec{B}^{spin}(\vec{p})\Big|_{spindensity}$ is computed by numerically integrating

$$\vec{B}^{spin}(\vec{p}) = \frac{\mu_0}{4\pi}\left(\frac{g_e \mu_B}{2}\right) \oint \left( \frac{3(\vec{p}-\vec{r}')\left(\vec{m}(\vec{r}') \bullet (\vec{p}-\vec{r}')\right)}{|\vec{p}-\vec{r}'|^5} - \frac{\vec{m}(\vec{r}')}{|\vec{p}-\vec{r}'|^3} \right) d^3\vec{r}'$$

over grid points $\vec{r}'$ inside the surface defined by 2.4× vdW radii. The set of grid points, $\{\vec{p}\}$, used to compute the MAE are distributed in the volume between surfaces defined by 3× and 4× vdW radii. The relative mean absolute error (RMAE) is then defined as the MAE for an ASM model divided by the MAE when all ASMs were set to zero.

Files for computing MAE and RMAE are found in the folder RRMSE_and_RMAE_scripts. At this time, RMAE is only supported for non-periodic systems. To compute RMAE in a nonperiodic system, replace the spin_density.cube (Gaussian file) with that for your system, place the atomic spin moments in the atomic_spin_moments_data.m file, and type calculate_nonperiodic_RMAE at the MATLAB prompt (or use the example script). The space between grid points in the spin_desity.cube file should be no more than 0.1 Å (0.2 bohr) and enough points around the molecule should be included to encompass the surface defined by 2.4× vdW radii. The total magnetic moment will be computed and printed in the output file; if this value is not within ±0.01 of the exact value you should consider using a finer grid in the spin_density.cube file.