Manipulation & 3D Plotting of Cube Files

Alistair Sterling

1st June 2021

1 Tutorial content

In this tutorial, we will cover the basics of how to generate cube files in Gaussian, and how to plot these cubes using Pymol and Chimera. We'll then go over some other things you can do with cube files, including generating NCI and ESP plots. For those interested in using ORCA instead of Gaussian, please refer to the appendix for details of how to do this.

To follow this tutorial, you will need to download (or have access to) the following software:

1. Gaussian (easiest to use on aleph)
2. Multiwfn (also easiest to use on aleph)
3. Pymol (easiest to be used locally, and can be downloaded for free at https://pymol.org/2/#download. When prompted, use the educational licence file sent with this tutorial to activate the software.)

2 Creating 3D structures in Pymol

To start this tutorial, we will need a 3D structure to give the surfaces some context. Open the xyz file (water_dimer.xyz) in Pymol.

Next, open up the text file "pymol_style", and copy the first block of text into the Pymol command line (press enter to run). This set of commands do the following:

1. ```
   bg_color white
   set depth_cue=1
   set ray_trace_fog=1
   ```

   Change the background colour to white, and give the image some perspective.
   ```

2.
```
show spheres
set sphere_scale, 0.18
```

   Show each atom as a sphere with a sensible size.

3.
```
set_color mred = [1.00, 0.40, 0.40]
color white, elem H
color mred, elem O
```

   Define a CMYK-safe red colour, then change H atoms to white and O atoms to red.

4.
```
show sticks
set stick_radius, 0.07
set stick_h_scale, 0.8
set stick_color, black
```

   Show bonds between atoms as black sticks, with a sensible stick radius for heavy atom bonds and a reduced radius for bonds to hydrogen.

Next, we can show a dashed line to represent the hydrogen bond. To do this:

1. Select one of the atoms by clicking on it (e.g. the H atom)
2. On the right hand panel, click **A** on the *sele* option, then go to **rename selection**
3. Enter a new name for the selection (e.g. **h1**)
4. Repeat this process for the other atom, naming the oxygen atom **o1**

Next, copy the text block from "pymol_style" that creates a dashed line between O and H. The commands do the following:

1.
```
dist d1, o1, h1
hide labels
```

   Define a distance between atoms **o1** and **h1** called **d1**, then hide the distance label that is automatically generated.

2.
```
set dash_radius, 0.05
set dash_gap, 0.2
set dash_color, black
```

   Change the radius of the dashed line, the space between the dashes, and the colour of the dashed line.

3 Generating orbital cube files with Gaussian

Chemists often like to use orbitals to rationalise molecular properties and reactivity.[1] The most commonly-encountered type of orbital in quantum chemistry is from an SCF calculation (either with HF theory or DFT), and it's sometimes useful to look at 3D representations of these orbitals.[2]

One way to visualise these orbitals is to take a checkpoint file from a Gaussian calculation, which contains information on the wavefunction of the system – i.e. the molecular orbital (MO) coefficients and basis set. In order to do anything with this information, the file first needs to be formatted.

To do this, we first need to request 4 cores on a compute node (it's always best to avoid running calculations on the head node):

```
qrsh -pe smp 4
```

Next, load Gaussian 16 in the terminal:

```
module load gaussian/g16
```

The *formchk* program can then be used to convert the checkpoint file into a formatted checkpoint file:

```
/usr/local/gaussian/g16/formchk water_dimer.chk water_dimer.fchk
```

You should now have a formatted checkpoint file called water_dimer.fchk in your current directory. This file can be used with many programs including Multiwfn, but we will use Gaussian's built-in cube generation program, *cubegen*, to plot some orbitals and surfaces.

To plot the highest occupied MO (HOMO) with *cubegen*, the following command can be run:

```
/usr/local/gaussian/g16/cubegen 4 MO=Homo water_dimer.fchk
    water_dimer_homo.cube
```

Follow the same procedure for the LUMO, replacing *MO=Homo* with *MO=Lumo*, and changing the name of the output cube file to *water_dimer_lumo.cube*.

More information on the syntax of cubegen can be found here:
https://gaussian.com/cubegen/

[1] Disclaimer: Opinions may vary on whether this is a good idea.

[2] There are many other types of orbitals that are often more 'interpretable' than the SCF orbitals, but for simplicity we won't discuss things like NBOs, NTOs, valence bond orbitals, etc. However, the same procedures mentioned below apply to plot these other types of orbitals.

4 Visualising orbitals with Pymol

Now we're ready to plot some orbitals! Judicious choice of the orbital names means you can just copy and paste the next block of text into the Pymol command line.

The commands work as follows:

1. `isosurface pos_homo, water_dimer_homo, 0.1`

   This command generates a positive isosurface of the HOMO and calls it "pos_homo", and sets its isovalue to 0.1.

2. `set surface_color, red, pos_homo`

   This command sets the colour of the orbital lobe.

3. `set transparency, 0.3`

   This command sets the isosurface transparency to 0.3.

You can play around with the isovalue and colour settings – there's no hard and fast rule for the isovalue but 0.1 is usually a good place to start.

Now we can save a png of the figure. To do this, copy the next block of text from "pymol_style" into the command line:

```
set ray_shadows,on
set ray_opaque_background, off
ray 1000,1000
png water_dimer_HOMO_LUMO, dpi=1600
```

This list of commands adds shadows and a transparent background to the final image, makes the plot 1000×1000 pixels in size, names the png "water_dimer_HOMO_LUMO.png", and uses a resolution of 1600 dpi.

It's likely that this will save the image to your home directory (you can also give the full path in the *png* line), wherever that might be. Your final figure should look something like Fig. 1.

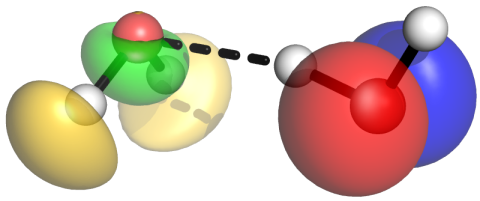


Figure 1: HOMO (red/blue lobes) and LUMO (green/gold lobes) of the water dimer, calculated at the RHF/3-21G level of theory and shown at an isovalue of 0.1.

5 Generating NCI plots with Pymol

Non-covalent interaction (NCI) plots provide a visual representation of electrostatic interactions, steric repulsion and van der Waal's interactions.[3] The basic principle behind the NCI method is that different kinds of local fluctuations in electron density can be assigned to different kinds of interaction.

To begin with, these local fluctuations must be characterised, which in this case is done using two quantities:

1. **The reduced density gradient**[4], s, a quantity which depends on the electron density ρ and its first derivative $\nabla\rho$, and indicates deviations from a uniform electron gas. NCIs are found in regions of space where both ρ and s are small, whereas covalency is indicated by small s but large ρ.

2. **The sign of the second eigenvalue of electron density Laplacian**[5] $(sign(\lambda_2)\rho)$. Bonding interactions (e.g. H–bonding) are characterised by an accumulation of density, so $\lambda_2 < 0$, whereas for non-bonding interactions (e.g. steric repulsion), $\lambda_2 > 0$. For van der Waal's interactions, $\lambda_2 \approx 0$. We can therefore use $sign(\lambda_2)\rho$ to classify the type of non-covalent interaction.

It's easiest to see how these plots work with an example. Using our water dimer, we can identify the hydrogen bond between the monomers. To do this, we first need to generate two files:

1. The reduced density gradient file (which we'll refer to as *rdg*)

[3] J. Am. Chem. Soc. **2010**, *132*, 6498
[4] $s \propto |\nabla\rho|/\rho^{4/3}$
[5] $\nabla^2\rho = \lambda_1 + \lambda_2 + \lambda_3$, where $(\lambda_1 \leq \lambda_2 \leq \lambda_3)$

2. The $sign(\lambda_2)\rho$ file (which we'll refer to as *lamb*)

One way to generate these files is using Multiwfn,[6] which is available on aleph.

To generate the *rdg* and *lamb* files, from the command line run (again on the compute node that you requested earlier):

```
/usr/local/bin/Multiwfn water_dimer.fchk
```

You should now be presented with an interface. To run the NCI calculation, select option **20** (*Visual study of weak interactions*), then option **1** (*NCI analysis (Also known as RDG analysis. JACS, 132, 6498)*). Since this is such a small system, select the "high quality grid" (option **3**)

The program should take a few seconds to run. Once it's finished, select option **3** (*Output cube files to func1.cub and func2.cub in current folder*). You can then exit Multiwfn with *ctrl+c* (or equivalent), and logout of the compute node by entering *logout*.

Rename *func1.cub* as *lamb.cube* and *func2.cub* as *rdg.cube* (otherwise it's easy to forget which is which).

Open *lamb.cube* and *rdg.cube* in the Pymol window containing the water_dimer orbital plots from earlier.

Next, paste the text in "Making NCI plots" from the "pymol_style" text file into the Pymol command line. The script does the following:

1. ```
   isosurface nci, rdg, 0.6
   ```

   This command generates an isosurface called *nci* from the reduced density gradient file (*rdg.cube*) with an (arbitrary) isovalue of 0.6.

2. ```
   ramp_new spectrum, lamb, [-0.01, 0, 0.01], [blue, green, red]
   ```

   This command generates a colour key called spectrum, which maps the value of $sign(\lambda_2)\rho$ to a colour (blue: $sign(\lambda_2)\rho < 0$, green: $sign(\lambda_2)\rho = 0$, red: $sign(\lambda_2)\rho > 0$).

3. ```
   set surface_color, spectrum, nci
   ```

   This command colours the *nci* isosurface by the colours mapped by *spectrum*.

4. ```
   set transparency, 0.4
   ```

   This command sets the isosurface transparency to 0.4.

You should now see something like Figure 2 on your screen. Unfortunately some versions of Pymol appear to have a bug where saving a png of this image turns the isosurface grey.

[6] J. Comput. Chem. **2012**, *33*, 580

If anyone has a solution to this, please let me know. As a rather inelegant work-around, you can screenshot the image in the Pymol window.

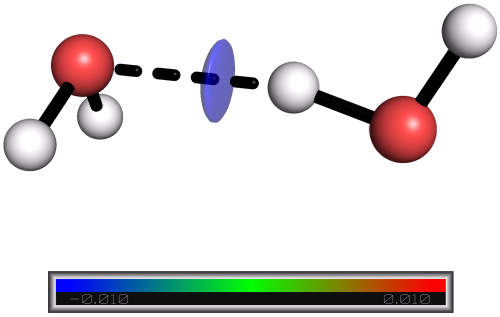


Figure 2: NCI isosurface for a water dimer, shown at an isovalue of 0.6. The blue lobe between H and O indicates an H-bonding interaction

6 Generating ESP plots with Pymol

Electrostatic potential (ESP) surfaces indicate how favourable it is to bring a negative point charge to a particular position in space. This analysis is commonly used to show sites of charge accumulation and depletion on the van der Waal's surface of a molecule, for example in the identification of σ-holes in halogen bond donors,[7] or binding mode preferences in supramolecular cage design.[8] The procedure for generating ESPs is taken from sourceforge.net/p/pymol/mailman/message/28230726/.

To generate an ESP plot, we need to return to the *cubegen* function in Gaussian.[9] We need to create two cube files:

1. An electron density file
2. An electrostatic potential file

The electron density is used to define the van der Waal's isosurface, and the ESP is projected onto this surface.

Next, request 4 cores on a compute node (see above), then load Gaussian 16 (also see above). To generate the electron density cube file, run:

```
cubegen 4 density water_dimer.fchk water_dimer_density.cube
```

[7] *Chem. Sci.* **2020**, *11*, 4722.

[8] *J. Chem. Inf. Model.* **2020**, *60*, 3546 & http://cgbind.chem.ox.ac.uk

[9] You can also do this in Multiwfn, but in my experience it generates quite a bit of noise which causes problems with the ESP surfaces.

Then generate the ESP cube file with the similar command:

```
cubegen 4 potential water_dimer.fchk water_dimer_esp.cube
```

Open both of these files in the Pymol window that you used for the orbital and NCI plotting, then paste the text in the "Making ESP plots" section of the "pymol_style" text file into the Pymol command line. These commands do the following:

1. ```
   isosurface dens, water_dimer_density, 0.002
   ```

   This command creates an isosurface called *dens* from the density cube file, with an isovalue of 0.002 (which approximates the van der Waal's surface, or 99% of the molecular electron density).

2. ```
   ramp_new esp_spectrum, water_dimer_esp, [-0.05, 0, 0.05], [blue,
   green, red]
   ```

   This command creates a colour map called esp_spectrum based on the ESP cube file.

3. ```
   set surface_color, esp_spectrum, dens
   ```

   This command projects the colour map defined in esp_spectrum onto the density isosurface (*dens*).

4. ```
   set transparency, 0.4
   ```

   This command sets the isosurface transparency to 0.4.

   You can then export a png of the ESP isosurface as you did for the orbitals (for some reason this works normally, unlike the NCI plot). You should end up with a plot that looks something like Figure 3.

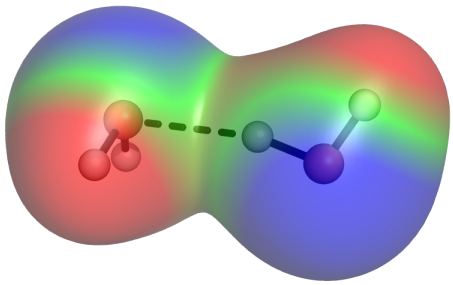


Figure 3: ESP isosurface for a water dimer, showing blue regions (ESP < 0, i.e. repulsive) and red regions (ESP > 0, i.e. attractive) to indicate the electron-rich and electron-poor parts of the molecule

7 Electron density difference plots

Cube files can be manipulated using the *cubman* program in Gaussian.[10] In this example, we'll look at how to calculate the difference between two electron densities.

In the EDDP directory, you'll find Gaussian input, output and formatted checkpoint files for the reduced and oxidised forms of the water dimer. Generate electron densities for these files using cubegen (see above), or use the cube files provided.

If we wanted to know (roughly) where an electron would be removed from the system, we could simply calculate the difference in electron density between the oxidised and neutral forms. We can do this with *cubman*:

1. First, run the cubman program:

   ```
   cubman
   ```

2. Enter "SU" to perform a subtraction.

3. Enter the name of the final state (i.e. the oxidised species, water_dimer_oxidised_density.cube), then enter "y" to indicate that the file has been formatted.

4. Enter the name of the initial state (i.e. the neutral species, water_dimer_density.cube), then enter "y" to indicate that the file has been formatted.

5. Enter the name of the desired output file (e.g. water_dimer_oxidised-neutral.cube, and again enter "y" to say that you'd like a formatted output file.

Load the resultant cube file into your Pymol session, then paste in the text from the first block of the "Electron density difference plots" section in the "pymol_style" text file. The script does the following:

1. ```
   isosurface oxidised_pos, water_dimer_oxidised-neutral, 0.01
   ```

   This command creates an isosurface called *oxidised_pos* from the density difference cube file, with an isovalue of 0.01.

2. ```
   set surface_color, blue, oxidised_pos
   ```

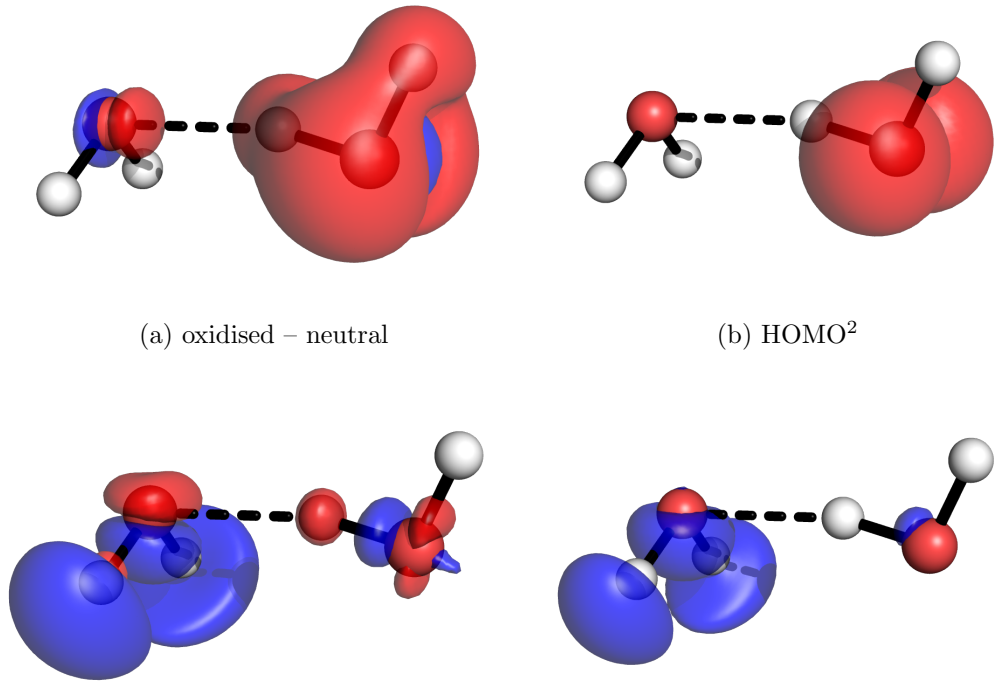
   This command colours the isosurface blue.

The remaining commands have been covered above.

The plot should resemble Figure 4a. Compare the plot with the square of the HOMO (Figure 4b) – do they look the same? Do you expect them to?

[10] Chimera also has this functionality, and also allows you to manipulate pairs of cube files with differing numbers of atoms, but for simplicity we'll stick with Gaussian.

You can repeat the same process for the addition of an electron to the molecule, and the results are shown in Figures 4c and 4d.



(a) oxidised – neutral

(b) HOMO^2

(c) reduced – neutral

(d) LUMO^2

Figure 4: Electron density difference plot for the removal (a) and addition (c) of an electron for the water dimer, and comparison with the squared HOMO (b) and LUMO (d) for the same system. The isovalue is 0.01. For (a) and (c), red lobes indicate electron density depletion, and blue lobes indicate electron density accumulation.

A Orbital and density plotting with ORCA

The *orca_plot* program provides an alternative to cubegen for the creation of cube files with ORCA. In most cases, a .gbw file is required for use with *orca_plot* (exceptions include .nbo, .pmp2re and .pmp2ur files). The .gbw file is equivalent to a chk file in Gaussian, and must be saved back to the submission directory after running an ORCA job.

The usage of *orca_plot* is relatively straightforward. For example, to plot an orbital, in the terminal:

1. `orca_plot <filename>.gbw -i`

   The "-i" flag is essential, and indicates that *orca_plot* will be run in interactive mode.

2. Select option **1** to choose the type of plot (e.g. MOs, electron density, spin density, ...)

3. If MO plotting is chosen, the number of the MO to plot should be chosen from option **2**. For unrestricted calculations, the choice between the alpha and beta set of MOs can be selected with option **3** (0=α, 1=β).

4. The number of grid intervals is selected with option **4** – a value of 120 gives a very fine grid (and is probably overkill in most cases).

5. To plot a cube file, select option **5** and then option **7** within the sub-menu.

6. Generate the plot with option **10**, and exit the program with option **11**.

The electron density can be saved during an ORCA calculation using the **KeepDens** keyword, and gives a .scfp file for single-reference methods. This file can be used in place of the .gbw file in *orca_plot* if the electron density is required. The same is true for perturbative methods (e.g. MP2), where either an unrelaxed (.pmp2ur) or relaxed (.pmp2re) density can be read into *orca_plot*. In principle, *orca_plot* can regenerate these densities from the .gbw file, but in practise this is very expensive and should be done during the main ORCA calculation.

Natural bonding orbitals (NBOs) can also be visualised with *orca_plot*, simply by using the .nbo file in place of the .gbw file, and noting that while ORCA counts from 0, NBO counts from 1.

The electron density of the ground state is the default when using a .gbw file. If you require the density of an excited state, for example from a CASSCF calculation, use the **KeepDens** keyword, and make sure to save the <filename>.cas.mult.<n>.root.<m>.scfp files (where n is the state multiplicity, and m is the root number for that state).