

pm-project-1

Prateek Mehta

November 5, 2012

Contents

1	Introduction	1
2	CO Example	2
3	Unit Cell Effects	4
4	H₂O Example	6
5	Tweaking the Opacity Transfer Function	8
6	Creating a Python Module	10
7	Illustration of the module usage with Chloroform and Carbon Dioxide molecules	10

1 Introduction

This project aims to plot electron density as a “fog” rather than with iso-surfaces using the visualization tool, Mayavi. DFT calculations have been performed using Jasp to generate the data.

A step by step approach has been used to illustrate how the final method of plotting the charge density was formulated. Examples of different molecules have been provided with every step. A python module has been written which plots the electron density using volume rendering. DFT calculations have been done using jasp. Details and examples on usage of

the module have been documented. The source code has uploaded along with this file on github.

The code has been tested for cubic and non-cubic unit cells and an issue regarding the low opacity of the electron fog has also been resolved by using the opacity transfer function in Mayavi.

As per the Mayavi documentation, the limitation of using volume rendering is that it can be difficult to analyze the details of the plotted field, even though it might look pretty. Other than that, no significant errors can be seen with the plots and the plotting works well with simple molecules like H₂O and CO to more complex molecules like CHCl₃ and CF₃Br. One frustrating thing about making such plots is that one has to manually adjust parameters to get the desired opacity of the fog using the standard process. An effort to resolve this using the opacity transfer function has been made, and once the afore mentioned python module has been formulated, its usage is much less tedious.

The following sections better illustrate the method.

2 CO Example

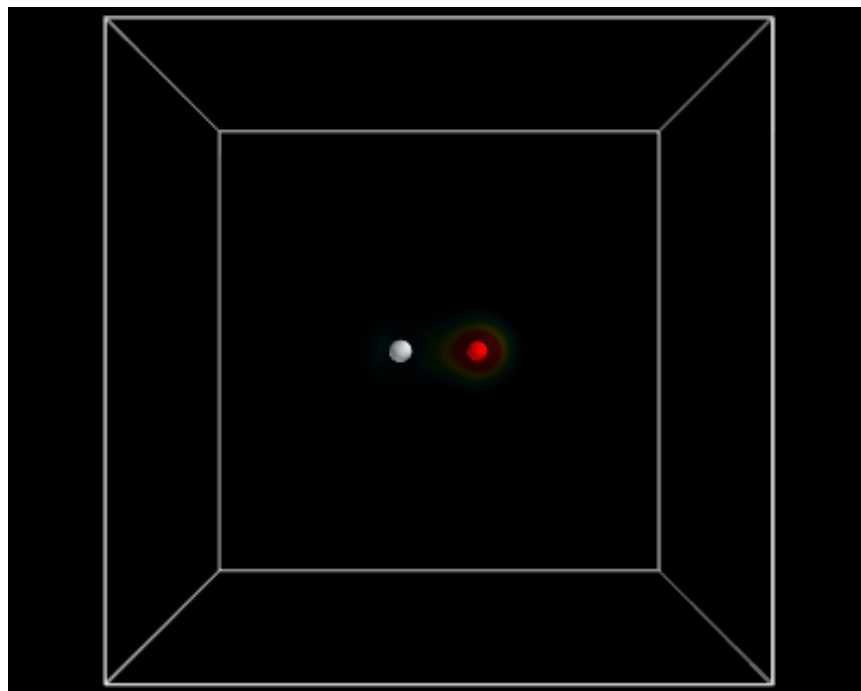
We start off with plotting the electron density fog for a simple CO molecule.

```
1 from jasp import *
2 from enthought.mayavi import mlab
3 from ase.data import vdw_radii
4 from ase.data.colors import cpk_colors
5 from ase import Atom, Atoms
6
7
8 # creating atoms object
9 atoms = Atoms([Atom('C', [2.422, 0.0, 0.0]),
10               Atom('O', [3.578, 0.0, 0.0])],
11              cell=(8,8,8))
12 atoms.center()
13
14 # running dft calculation
15 with jasp('molecules/fog-co',
16          encut=350,
17          xc='PBE',
18          atoms=atoms) as calc:
19     calc.calculate()
20     atoms = calc.get_atoms()
21     x, y, z, cd = calc.get_charge_density()
22
23 # making a black background
24 mlab.figure(bgcolor=(0, 0, 0))
25
26 # plotting the atoms as spheres,
```

```

27
28 for atom in atoms:
29     mlab.points3d(atom.x,
30                   atom.y,
31                   atom.z,
32                   scale_factor=vdw_radii[atom.number]/5.,
33                   resolution=20,
34                   # a tuple is required for the color
35                   color=tuple(cpk_colors[atom.number]),
36                   scale_mode='none')
37
38 # draw the unit cell - there are 8 corners, and 12 connections
39 a1, a2, a3 = atoms.get_cell()
40 origin = [0, 0, 0]
41 cell_matrix = [[origin, a1],
42                [origin, a2],
43                [origin, a3],
44                [a1, a1 + a2],
45                [a1, a1 + a3],
46                [a2, a2 + a1],
47                [a2, a2 + a3],
48                [a3, a1 + a3],
49                [a3, a2 + a3],
50                [a1 + a2, a1 + a2 + a3],
51                [a2 + a3, a1 + a2 + a3],
52                [a1 + a3, a1 + a3 + a2]]
53
54 for p1, p2 in cell_matrix:
55     mlab.plot3d([p1[0], p2[0]], # x-positions
56                [p1[1], p2[1]], # y-positions
57                [p1[2], p2[2]], # z-positions
58                tube_radius=0.02)
59
60 # plotting the charge density
61 source = mlab.pipeline.scalar_field(x,y,z,cd)
62 min=cd.min()
63 max=cd.max()
64
65 # vmin and vmax have to be adjusted for best view by iteration
66 vol = mlab.pipeline.volume(source, vmin=min+ 0.1*(max-min),
67                             vmax= min+0.6*(max-min))
68
69 # view adjusted by iteration
70 mlab.view(azimuth=-90, elevation=90, distance='auto')
71
72 mlab.savefig('molecules/co-fog.png')
73 mlab.show()

```



The approach above is pretty much the same as the one in dft-book with volume rendering being used to plot the electron density instead of isosurfaces. The parameters ‘vmin’ and ‘vmax’ need to be adjusted empirically to achieve the best visualization.

3 Unit Cell Effects

The example code(http://docs.enthought.com/mayavi/mayavi/auto/example_chemistry.html) had been written for a water molecule having a cubic unit cell, with scaling done for all the atom positions individually. It can be easy to wonder that the code might fail for cells which are not perfect cubes (as I did). In the following example we extend our analysis to check whether our code is valid for other unit cells.

```
1 from jasp import *
2 from enthought.mayavi import mlab
3 from ase.data import vdw_radii
4 from ase.data.colors import cpk_colors
5 from ase import Atom, Atoms
6
7
8 # creating atoms object
```

```

9  atoms = Atoms([Atom('C', [2.422, 0.0, 0.0]),
10                 Atom('O', [3.578, 0.0, 0.0])],
11               cell=(8,5,4))
12  atoms.center()
13
14  # running dft calculation
15  with jasp('molecules/fog-co-cell',
16            encut=350,
17            xc='PBE',
18            atoms=atoms) as calc:
19      calc.calculate()
20      atoms = calc.get_atoms()
21      x, y, z, cd = calc.get_charge_density()
22
23  # making a black background
24  mlab.figure(bgcolor=(0, 0, 0))
25
26  # plotting the atoms as spheres,
27
28  for atom in atoms:
29      mlab.points3d(atom.x,
30                    atom.y,
31                    atom.z,
32                    scale_factor=vdw_radii[atom.number]/5.,
33                    resolution=20,
34                    # a tuple is required for the color
35                    color=tuple(cpk_colors[atom.number]),
36                    scale_mode='none')
37
38  # draw the unit cell - there are 8 corners, and 12 connections
39  a1, a2, a3 = atoms.get_cell()
40  origin = [0, 0, 0]
41  cell_matrix = [[origin, a1],
42                 [origin, a2],
43                 [origin, a3],
44                 [a1, a1 + a2],
45                 [a1, a1 + a3],
46                 [a2, a2 + a1],
47                 [a2, a2 + a3],
48                 [a3, a1 + a3],
49                 [a3, a2 + a3],
50                 [a1 + a2, a1 + a2 + a3],
51                 [a2 + a3, a1 + a2 + a3],
52                 [a1 + a3, a1 + a3 + a2]]
53
54  for p1, p2 in cell_matrix:
55      mlab.plot3d([p1[0], p2[0]], # x-positions
56                 [p1[1], p2[1]], # y-positions
57                 [p1[2], p2[2]], # z-positions
58                 tube_radius=0.02)
59
60  # plotting the charge density
61  source = mlab.pipeline.scalar_field(x,y,z,cd)
62  min=cd.min()
63  max=cd.max()
64

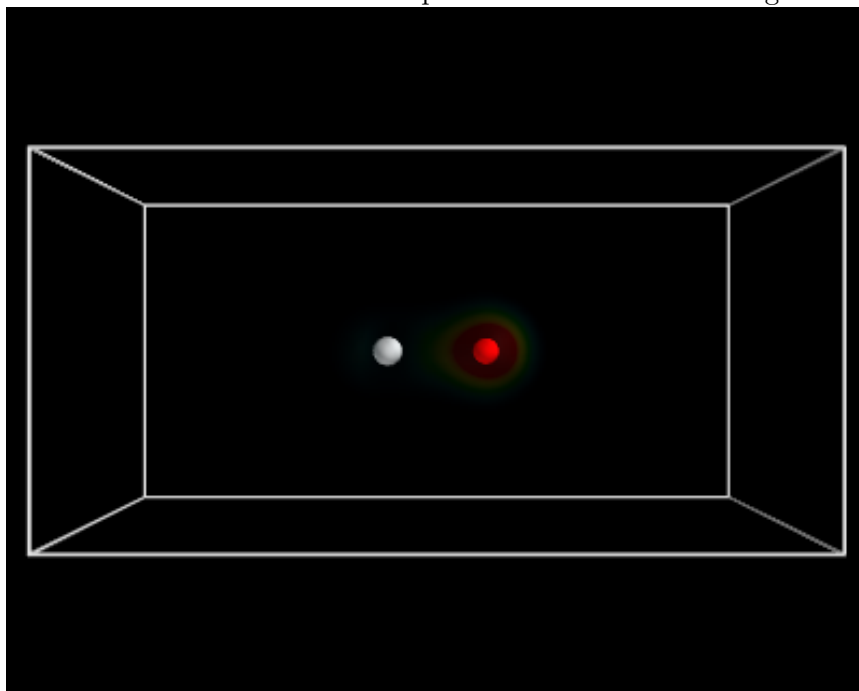
```

```

65 # vmin and vmax have to be adjusted for best view by iteration
66 vol = mlab.pipeline.volume(source, vmin=min+ 0.1*(max-min),
67                             vmax= min+0.6*(max-min))
68
69 # view adjusted by iteration
70 mlab.view(azimuth=-90, elevation=90, distance='auto')
71
72 mlab.savefig('molecules/co-cell-fog.png')
73 mlab.show()

```

We see that the unit cell shape does not affect the charge density plot.



4 H₂O Example

A similar approach has been taken to plot the electron density of the water molecule. The unit cell has not been shown in order to get a better picture of the electron density fog.

```

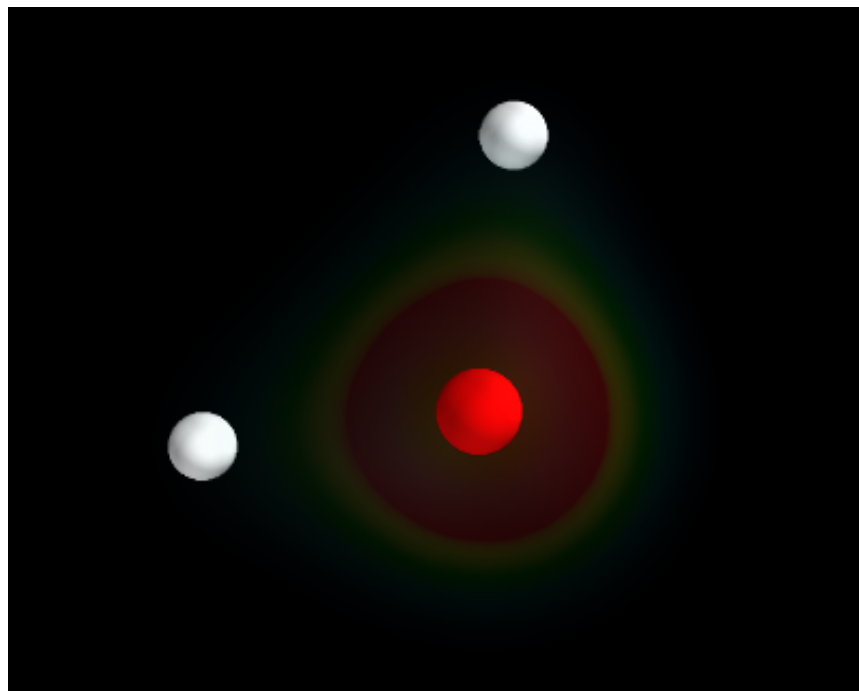
1 from jasp import *
2 from enthought.mayavi import mlab
3 from ase.data import vdw_radii
4 from ase.data.colors import cpk_colors
5 from ase import Atom, Atoms
6

```

```

7
8 #defining atoms object
9 atoms = Atoms([Atom('H', [0.5960812, -0.7677068, 0.0000000]),
10               Atom('O', [0.0000000, 0.0000000, 0.0000000]),
11               Atom('H', [0.5960812, 0.7677068, 0.0000000])),
12               cell=(8, 8, 8))
13 atoms.center()
14
15 # running a dft calculation
16 with jasp('molecules/water-fog',
17           encut=350,
18           xc='PBE',
19           atoms=atoms) as calc:
20     calc.calculate()
21     atoms = calc.get_atoms()
22     x, y, z, cd = calc.get_charge_density()
23
24 # making a black background
25 n1, n2, n3=cd.shape
26 mlab.figure(bgcolor=(0, 0, 0))
27
28 # Plotting the atoms as spheres
29 for atom in atoms:
30     mlab.points3d(atom.x,
31                   atom.y,
32                   atom.z,
33                   scale_factor=vdw_radii[atom.number]/5.,
34                   resolution=20,
35                   color=tuple(cpk_colors[atom.number]),
36                   scale_mode='none')
37
38 #Plotting the charge density
39 source = mlab.pipeline.scalar_field(x,y,z,cd)
40 min = cd.min()
41 max = cd.max()
42 vol = mlab.pipeline.volume(source, vmin=min+.05*(max-min),
43                             vmax=min+0.3*(max-min))
44 mlab.view(azimuth=135, elevation=0, distance='auto')
45 mlab.savefig('molecules/water-fog.png')
46 mlab.show()

```



5 Tweaking the Opacity Transfer Function

Changing ‘vmin’ and ‘vmax’ values manually is quite tedious and doesn’t allow us to properly control the opacity of the electron density fog. Moreover, in some cases, the electron density can hardly be seen. For a better visualization, we make changes in the Opacity Transfer Function(OTF).

The procedure using a CF_3Br molecule is described in the following section.

```

1 from jasp import *
2 from enthought.mayavi import mlab
3 from ase.data import vdw_radii
4 from ase.data.colors import cpk_colors
5 from ase import Atom, Atoms
6
7
8 # creating atoms object
9 atoms = Atoms([Atom('C', [ 0.0000,      0.0000,     -0.8088]),
10               Atom('Br', [ 0.0000,      0.0000,      1.1146]),
11               Atom('F', [ 0.0000,      1.2455,     -1.2651]),
12               Atom('F', [ 1.0787,     -0.6228,     -1.2651]),
13               Atom('F', [-1.0787,     -0.6228,     -1.2651])],
14               cell=(10, 10, 10))

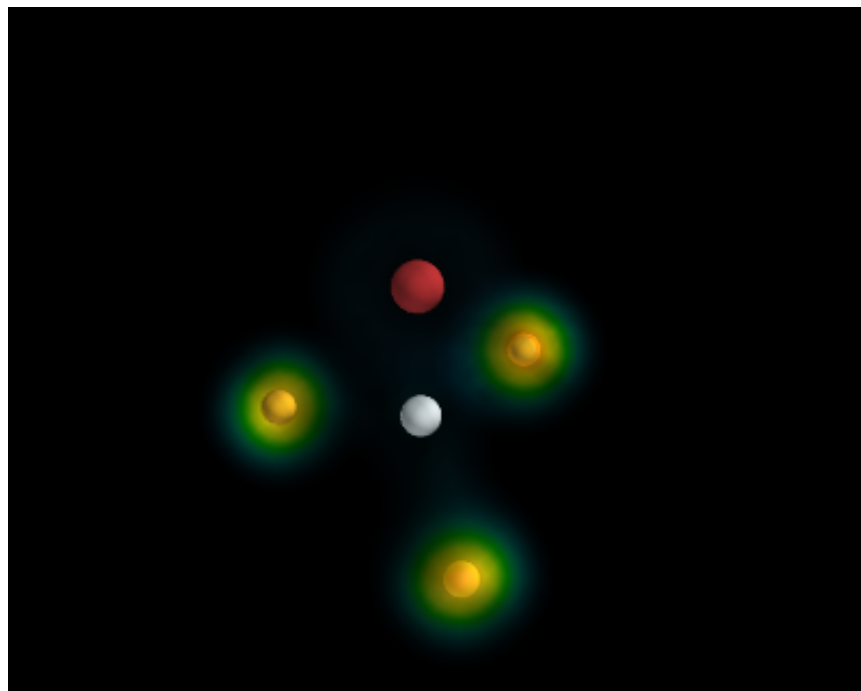
```



```

15 atoms.center()
16
17 # running dft calculation
18 with jasp('molecules/fog-CF3Br',
19         encut=350,
20         xc='PBE',
21         ibrion=1,
22         nsw=50,
23         atoms=atoms) as calc:
24     calc.calculate()
25     atoms = calc.get_atoms()
26     x, y, z, cd = calc.get_charge_density()
27
28 # making a black background
29 mlab.figure(bgcolor=(0, 0, 0))
30
31 # plotting the atoms as spheres,
32
33 for atom in atoms:
34     mlab.points3d(atom.x,
35                 atom.y,
36                 atom.z,
37                 scale_factor=vdw_radii[atom.number]/5.,
38                 resolution=20,
39                 # a tuple is required for the color
40                 color=tuple(cpk_colors[atom.number]),
41                 scale_mode='none')
42
43 # plotting the charge density
44 source = mlab.pipeline.scalar_field(x,y,z,cd)
45 min=cd.min()
46 max=cd.max()
47
48 # vmin and vmax have to be adjusted for best view by iteration
49 vol = mlab.pipeline.volume(source)
50
51 # Changing the otf
52 from enthought.tvtk.util.ctf import PiecewiseFunction
53 otf = PiecewiseFunction()
54 otf.add_point(min+0.1*(max-min),0)
55 otf.add_point(min+0.9*(max-min),0.8)
56 vol._otf = otf
57 vol._volume_property.set_scalar_opacity(otf)
58
59 #view adjusted by iteration
60 mlab.view(azimuth=-45, elevation=30, distance='auto')
61
62 mlab.savefig('molecules/CF3Br-fog.png')
63 mlab.show()

```



As can be seen from the figure, a much better visualization has been achieved than the ones shown previously.

6 Creating a Python Module

Now that a good visualization has been achieved and we are pretty confident there are no significant errors with our method, we go on to create a python module which plots the volume rendered charge density thus saving us repetitive effort.

The module has been saved in the file `PlotFog.py`. The source code has been uploaded along with this file on github and illustrations using the module are shown in the next section.

7 Illustration of the module usage with Chloroform and Carbon Dioxide molecules

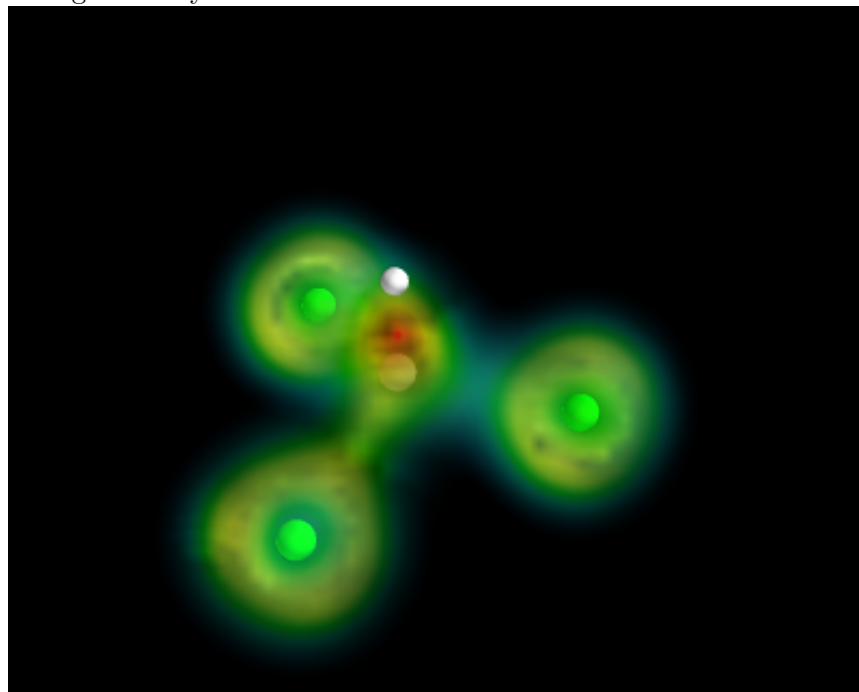
With everything in place we can now use our module to plot charge density for the CHCl_3 and CO_2 molecules.

```

1  from jasp import *
2  from ase.data.molecules import molecule
3  from PlotFog import *
4
5  # creating atoms object
6  atoms = molecule('HCCl3')
7  atoms.set_cell([8,8,8],scale_atoms=False)
8  atoms.center()
9
10 # running dft calculation
11 with jasp('molecules/example/chloroform-fog',
12          encut=350,
13          xc='PBE',
14          ibrion=1,
15          nsw=50,
16          atoms=atoms) as calc:
17     calc.calculate()
18
19 # using the module plot_charge_density_fog to plot the charge density
20
21 plot_charge_density_fog('molecules/example/chloroform-fog',
22                         UnitCell=False,
23                         opacity=[0,0.8],
24                         view=[0,45],
25                         SaveFig=True,
26                         filepath='molecules/chloroform-fog.png')

```

Charge Density for Chloroform.



```

1  from jasp import *
2  from ase.data.molecules import molecule
3  from PlotFog import *
4
5  # creating atoms object
6  atoms = molecule('CO2')
7  atoms.set_cell([8,8,8],scale_atoms=False)
8  atoms.center()
9
10 # running dft calculation
11 with jasp('molecules/example/carbondioxide-fog',
12          encut=350,
13          xc='PBE',
14          ibrion=1,
15          nsw=50,
16          atoms=atoms) as calc:
17     calc.calculate()
18
19 # using the module plot_charge_density_fog to plot the charge density
20 plot_charge_density_fog('molecules/example/carbondioxide-fog',
21                          UnitCell=True,
22                          opacity=[0,0.8],
23                          view=[-90,90],
24                          SaveFig=True,
25                          filepath='molecules/co2-fog.png')

```

Charge density plotted for the CO₂ molecule.

