pm-project-1

Prateek Mehta

November 3, 2012

Contents

1 Introduction

This project aims to plot electron density as a “fog” rather than with isosurfaces 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 H_2O and CO to more complex molecules like $CHCl_3$ and CF_3Br. 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.

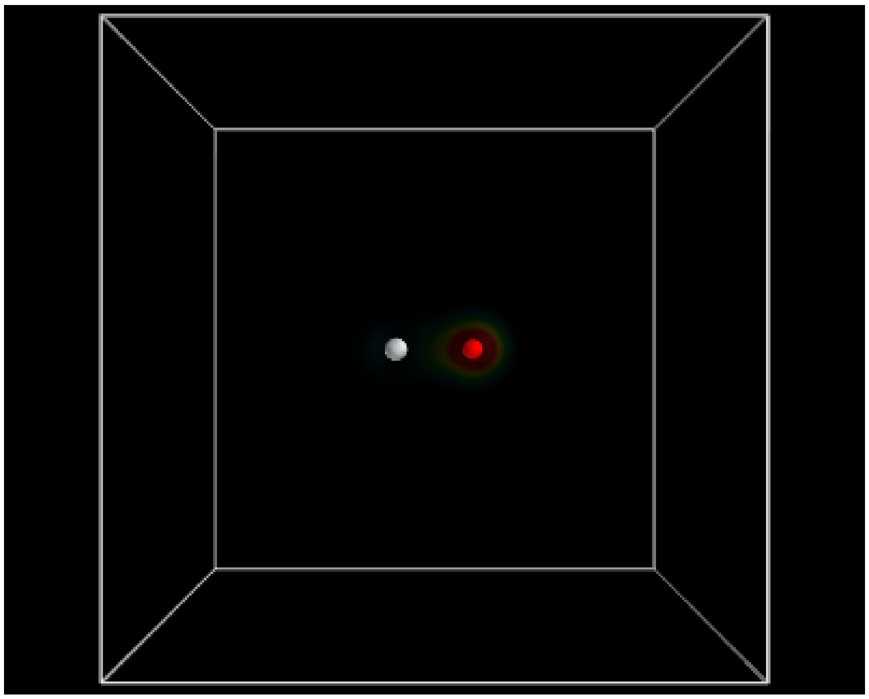
The following sections better illustrate the method.

2 CO Example

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

```
from jasp import *
from enthought.mayavi import mlab
from ase.data import vdw_radii
from ase.data.colors import cpk_colors
from ase import Atom, Atoms


# creating atoms object
atoms = Atoms([Atom('C', [2.422, 0.0, 0.0]),
               Atom('O', [3.578, 0.0, 0.0])],
               cell=(8,8,8))
atoms.center()

# running dft calculation
with jasp('molecules/fog-co',
          encut=350,
          xc='PBE',
          atoms=atoms) as calc:
    calc.calculate()
    atoms = calc.get_atoms()
    x, y, z, cd = calc.get_charge_density()

# making a black background
mlab.figure(bgcolor=(0, 0, 0))

# plotting the atoms as spheres,
```

```python

for atom in atoms:
    mlab.points3d(atom.x,
                  atom.y,
                  atom.z,
                  scale_factor=vdw_radii[atom.number]/5.,
                  resolution=20,
                  # a tuple is required for the color
                  color=tuple(cpk_colors[atom.number]),
                  scale_mode='none')

# draw the unit cell - there are 8 corners, and 12 connections
a1, a2, a3 = atoms.get_cell()
origin = [0, 0, 0]
cell_matrix = [[origin,  a1],
               [origin,  a2],
               [origin,  a3],
               [a1,      a1 + a2],
               [a1,      a1 + a3],
               [a2,      a2 + a1],
               [a2,      a2 + a3],
               [a3,      a1 + a3],
               [a3,      a2 + a3],
               [a1 + a2, a1 + a2 + a3],
               [a2 + a3, a1 + a2 + a3],
               [a1 + a3, a1 + a3 + a2]]

for p1, p2 in cell_matrix:
    mlab.plot3d([p1[0], p2[0]], # x-positions
                [p1[1], p2[1]], # y-positions
                [p1[2], p2[2]], # z-positions
                tube_radius=0.02)

# plotting the charge density
source = mlab.pipeline.scalar_field(x,y,z,cd)
min=cd.min()
max=cd.max()

# vmin and vmax have to be adjusted for best view by iteration
vol = mlab.pipeline.volume(source, vmin=min+ 0.1*(max-min),
                                   vmax= min+0.6*(max-min))

# view adjusted by iteration
mlab.view(azimuth=-90, elevation=90, distance='auto')

mlab.savefig('molecules/co-fog.png')
mlab.show()
```



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

3 Unit Cell Effects

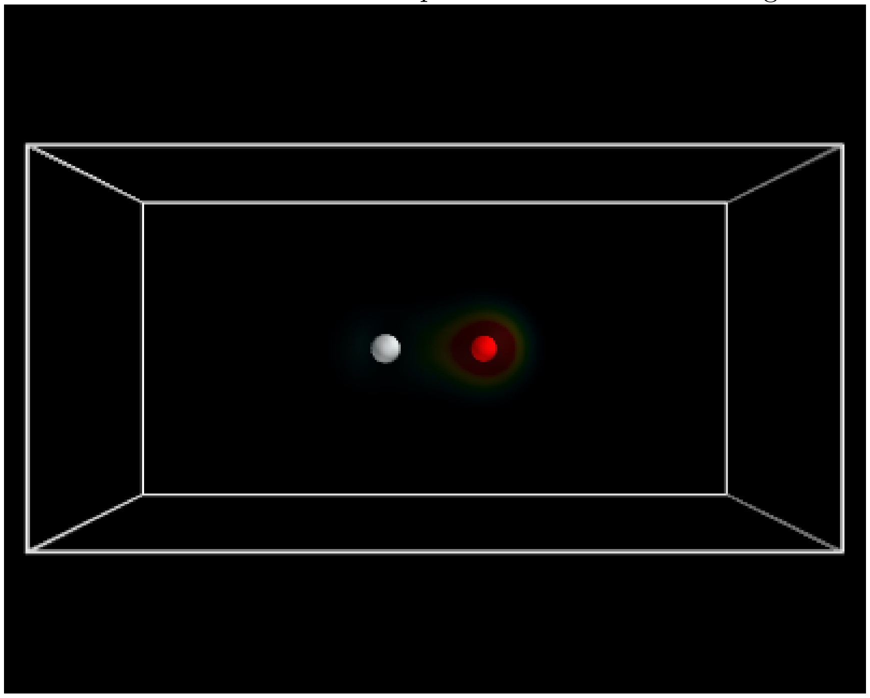
The example code(http://docs.enthought.com/mayavi/mayavi/auto/example_chemistry.html) had been written for a water molecule having a cubic unit cell, with scaling done for all the atom positions individually. It can be easy to wonder that the code might fail for cells which are not perfect cubes (as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

```
from jasp import *
from enthought.mayavi import mlab
from ase.data import vdw_radii
from ase.data.colors import cpk_colors
from ase import Atom, Atoms


# creating atoms object
```

```python
atoms = Atoms([Atom('C', [2.422, 0.0, 0.0]),
               Atom('O', [3.578, 0.0, 0.0])],
               cell=(8,5,4))
atoms.center()

# running dft calculation
with jasp('molecules/fog-co-cell',
          encut=350,
          xc='PBE',
          atoms=atoms) as calc:
    calc.calculate()
    atoms = calc.get_atoms()
    x, y, z, cd = calc.get_charge_density()

# making a black background
mlab.figure(bgcolor=(0, 0, 0))

# plotting the atoms as spheres,

for atom in atoms:
    mlab.points3d(atom.x,
                  atom.y,
                  atom.z,
                  scale_factor=vdw_radii[atom.number]/5.,
                  resolution=20,
                  # a tuple is required for the color
                  color=tuple(cpk_colors[atom.number]),
                  scale_mode='none')

# draw the unit cell - there are 8 corners, and 12 connections
a1, a2, a3 = atoms.get_cell()
origin = [0, 0, 0]
cell_matrix = [[origin,  a1],
               [origin,  a2],
               [origin,  a3],
               [a1,      a1 + a2],
               [a1,      a1 + a3],
               [a2,      a2 + a1],
               [a2,      a2 + a3],
               [a3,      a1 + a3],
               [a3,      a2 + a3],
               [a1 + a2, a1 + a2 + a3],
               [a2 + a3, a1 + a2 + a3],
               [a1 + a3, a1 + a3 + a2]]

for p1, p2 in cell_matrix:
    mlab.plot3d([p1[0], p2[0]], # x-positions
                [p1[1], p2[1]], # y-positions
                [p1[2], p2[2]], # z-positions
                tube_radius=0.02)

# plotting the charge density
source = mlab.pipeline.scalar_field(x,y,z,cd)
min=cd.min()
max=cd.max()

```

```
# vmin and vmax have to be adjusted for best view by iteration
vol = mlab.pipeline.volume(source, vmin=min+ 0.1*(max-min),
                                   vmax= min+0.6*(max-min))

# view adjusted by iteration
mlab.view(azimuth=-90, elevation=90, distance='auto')

mlab.savefig('molecules/co-cell-fog.png')
mlab.show()
```

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

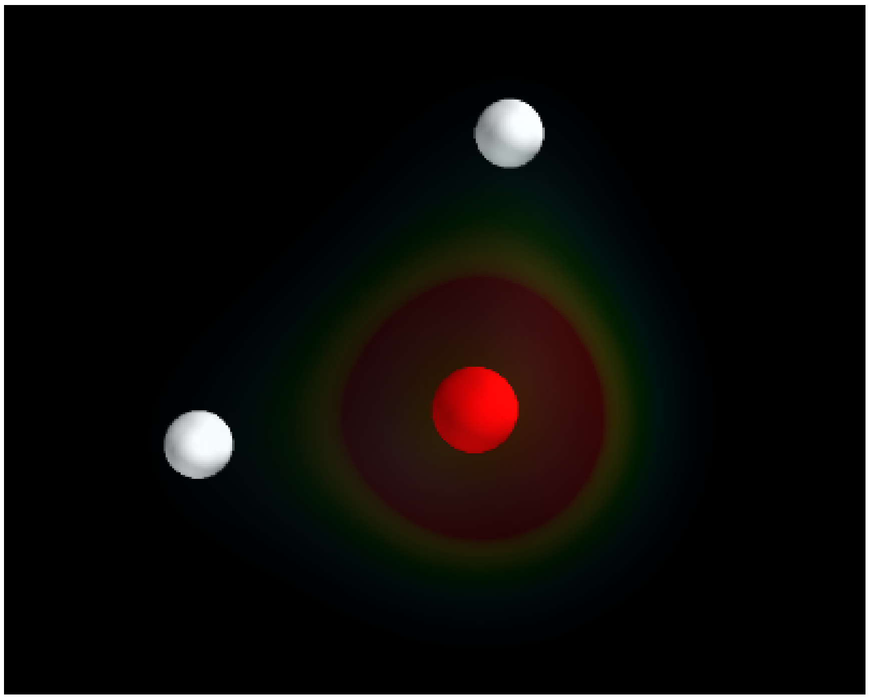


4 H_2O Example

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.

```
from jasp import *
from enthought.mayavi import mlab
from ase.data import vdw_radii
from ase.data.colors import cpk_colors
from ase import Atom, Atoms

```

```python

#defining atoms object
atoms = Atoms([Atom('H', [0.5960812,  -0.7677068,   0.0000000]),
               Atom('O', [0.0000000,   0.0000000,   0.0000000]),
               Atom('H', [0.5960812,   0.7677068,   0.0000000])],
               cell=(8, 8, 8))
atoms.center()

# running a dft calculation
with jasp('molecules/water-fog',
          encut=350,
          xc='PBE',
          atoms=atoms) as calc:
    calc.calculate()
    atoms = calc.get_atoms()
    x, y, z, cd = calc.get_charge_density()

# making a black background
n1, n2, n3=cd.shape
mlab.figure(bgcolor=(0, 0, 0))

# Plotting the atoms as spheres
for atom in atoms:
    mlab.points3d(atom.x,
                  atom.y,
                  atom.z,
                  scale_factor=vdw_radii[atom.number]/5.,
                  resolution=20,
                  color=tuple(cpk_colors[atom.number]),
                  scale_mode='none')

#Plotting the charge density
source = mlab.pipeline.scalar_field(x,y,z,cd)
min = cd.min()
max = cd.max()
vol = mlab.pipeline.volume(source, vmin=min+.05*(max-min),
                                   vmax=min+0.3*(max-min))
mlab.view(azimuth=135, elevation=0, distance='auto')
mlab.savefig('molecules/water-fog.png')
mlab.show()
```



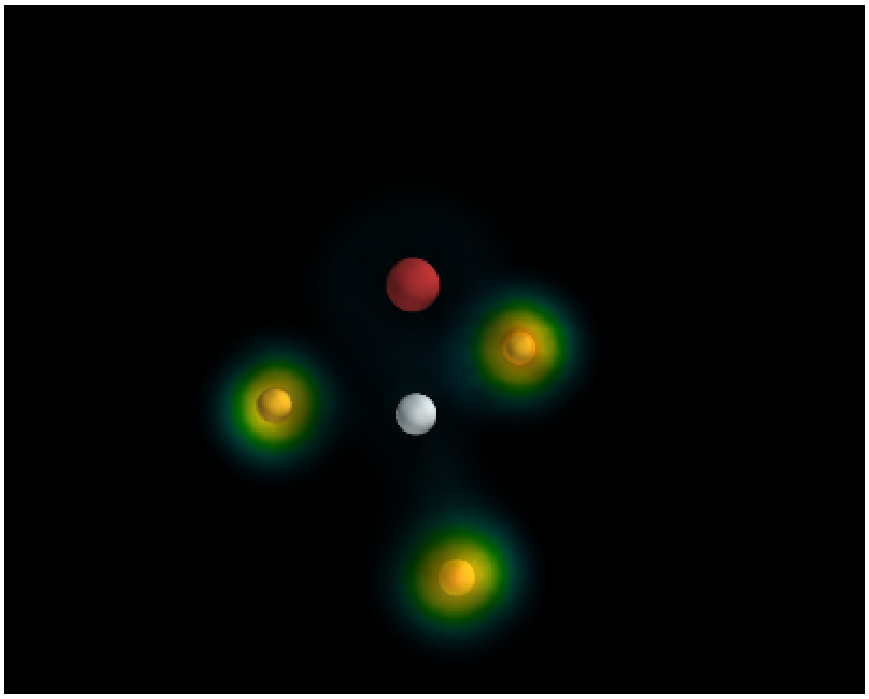
5 Tweaking the Opacity Transfer Function

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 follwing section.

```
from jasp import *
from enthought.mayavi import mlab
from ase.data import vdw_radii
from ase.data.colors import cpk_colors
from ase import Atom, Atoms


# creating atoms object
atoms = Atoms([Atom('C',  [ 0.0000,     0.0000,     -0.8088]),
               Atom('Br', [ 0.0000,     0.0000,      1.1146]),
               Atom('F',  [ 0.0000,     1.2455,     -1.2651]),
               Atom('F',  [ 1.0787,    -0.6228,     -1.2651]),
               Atom('F',  [-1.0787,    -0.6228,     -1.2651])],
               cell=(10, 10, 10))
```

```
atoms.center()

# running dft calculation
with jasp('molecules/fog-CF3Br',
          encut=350,
          xc='PBE',
          ibrion=1,
          nsw=50,
          atoms=atoms) as calc:
    calc.calculate()
    atoms = calc.get_atoms()
    x, y, z, cd = calc.get_charge_density()

# making a black background
mlab.figure(bgcolor=(0, 0, 0))

# plotting the atoms as spheres,

for atom in atoms:
    mlab.points3d(atom.x,
                  atom.y,
                  atom.z,
                  scale_factor=vdw_radii[atom.number]/5.,
                  resolution=20,
                  # a tuple is required for the color
                  color=tuple(cpk_colors[atom.number]),
                  scale_mode='none')

# plotting the charge density
source = mlab.pipeline.scalar_field(x,y,z,cd)
min=cd.min()
max=cd.max()

# vmin and vmax have to be adjusted for best view by iteration
vol = mlab.pipeline.volume(source)

# Changing the otf
from enthought.tvtk.util.ctf import PiecewiseFunction
otf = PiecewiseFunction()
otf.add_point(min+0.1*(max-min),0)
otf.add_point(min+0.9*(max-min),0.8)
vol._otf = otf
vol._volume_property.set_scalar_opacity(otf)

#view adjusted by iteration
mlab.view(azimuth=-45, elevation=30, distance='auto')

mlab.savefig('molecules/CF3Br-fog.png')
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.

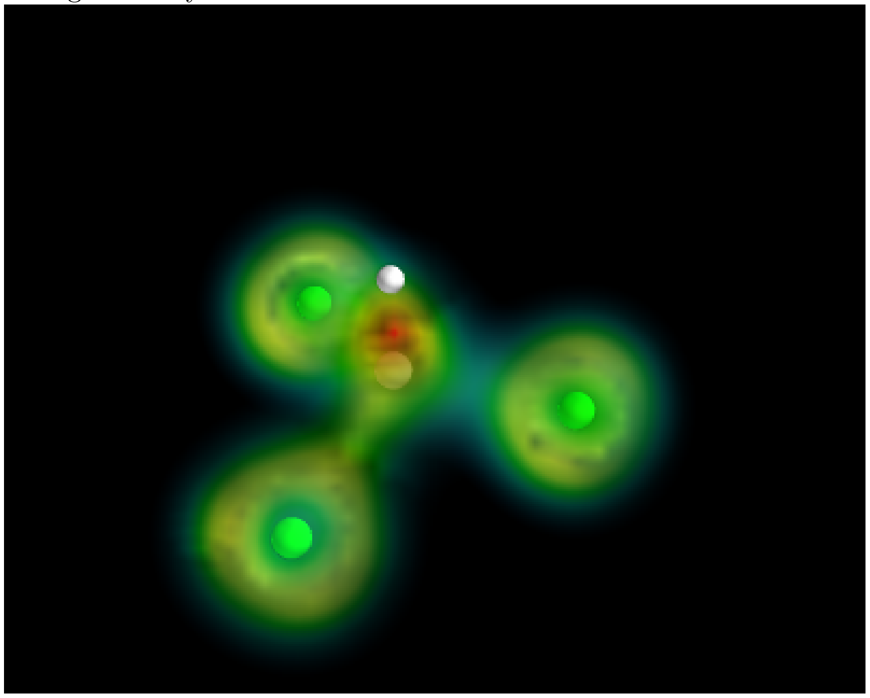
The module has been saved in the file PlotFog.py. The source code has been uploaded along with this file on github and illustrations using the module are shown in the next section.

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.

```python
from jasp import *
from ase.data.molecules import molecule
from PlotFog import *

# creating atoms object
atoms = molecule('HCCl3')
atoms.set_cell([8,8,8],scale_atoms=False)
atoms.center()

# running dft calculation
with jasp('molecules/example/chloroform-fog',
          encut=350,
          xc='PBE',
          ibrion=1,
          nsw=50,
          atoms=atoms) as calc:
      calc.calculate()

# using the module plot_charge_density_fog to plot the charge density

plot_charge_density_fog('molecules/example/chloroform-fog',
                        UnitCell=False,
                        opacity=[0,0.5],
                        view=[0,45],
                        SaveFig=True,
                        filepath='molecules/chloroform-fog.png')
```

Charge Density for Chloroform.



```python
from jasp import *
from ase.data.molecules import molecule
from PlotFog import *

# creating atoms object
atoms = molecule('CO2')
atoms.set_cell([8,8,8],scale_atoms=False)
atoms.center()

# running dft calculation
with jasp('molecules/example/carbondioxide-fog',
          encut=350,
          xc='PBE',
          ibrion=1,
          nsw=50,
          atoms=atoms) as calc:
      calc.calculate()

# using the module plot_charge_density_fog to plot the charge density
plot_charge_density_fog('molecules/example/carbondioxide-fog',
                        UnitCell=True,
                        opacity=[0,0.5],
                        view=[-90,90],
                        SaveFig=True,
                        filepath='molecules/co2-fog.png')
```

Charge density plotted for the CO_2 molecule.