

Molecular Simulation HOMEWORK 3: Properties of nitromethane

Zhongnan Xu

9/18/12 Thursday

Contents

1	Reaction energy of CO oxidation	1
1.1	Compute the reaction energy for $\text{CO} + 1/2 \text{O}_2 \rightarrow \text{CO}_2$	1
1.2	Convergence test	3
2	Zero-point energy corrections	5
2.1	Compute vibrational modes for CO, CO ₂ and O ₂	5
2.2	Compute the CO oxidation reaction energy with zero-point energy corrections.	7
2.3	Compare your computed energy to a value from the literature.	8
3	Plot the electron density of the CO₂ molecule.	8

1 Reaction energy of CO oxidation

1.1 Compute the reaction energy for $\text{CO} + 1/2 \text{O}_2 \rightarrow \text{CO}_2$

Use a cutoff energy of 250 eV. The molecules should all be relaxed to their lowest energy geometry (perform a geometry optimization). Demonstrate that all the forces on the molecule are less than 0.05 eV/Å.

```
1 from ase import Atoms, Atom
2 from jasp import *
3 import numpy as np
4 np.set_printoptions(precision=3, suppress=True)
5
6 CO = Atoms([Atom('C', (0, 0, 0)),
7             Atom('O', (1.2, 0, 0))],
8            cell=(10, 10, 10))
9 O2 = Atoms([Atom('O', (0, 0, 0)),
10            Atom('O', (1.23, 0, 0))],
11            cell=(9.8, 9.9, 10))
12 CO2 = Atoms([Atom('O', (0, 0, 0)),
13             Atom('C', (1.16, 0, 0)),
14             Atom('O', (2.32, 0, 0))],
15             cell=(10, 10, 10))
16 with jasp('probia/CO',
```

```

17         xc='PBE', lreal=False,
18         encut=250, prec='Accurate',
19         kpts=(1, 1, 1), ismear=1, sigma=0.05,
20         ibrion=1, nsw=50, ediffg=-0.05, isif=2,
21         atoms=C0) as C0calc:
22     try:
23         eC0 = C0.get_potential_energy()
24         fC0 = C0.get_forces()
25     except:
26         pass
27 with jasp('probia/O2',
28         xc='PBE', lreal=False,
29         encut=250, prec='Accurate',
30         kpts=(1, 1, 1), ismear=1, sigma=0.05,
31         ibrion=1, nsw=50, ediffg=-0.05, isif=2,
32         atoms=O2) as O2calc:
33     try:
34         eO2 = O2.get_potential_energy()
35         fO2 = O2.get_forces()
36     except:
37         pass
38 with jasp('probia/CO2',
39         xc='PBE', lreal=False,
40         encut=250, prec='Accurate',
41         kpts=(1, 1, 1), ismear=1, sigma=0.05,
42         ibrion=1, nsw=50, ediffg=-0.05, isif=2,
43         atoms=CO2) as CO2calc:
44     try:
45         eCO2 = CO2.get_potential_energy()
46         fCO2 = CO2.get_forces()
47     except:
48         pass
49
50 re = eCO2 - eC0 - 0.5*eO2
51 print 'The total energy of C0 is {0:1.3f}'.format(eC0)
52 print 'The forces (eV/angstrom) on the atoms in C0 are'
53 print 'C: {0}'.format(fC0[0])
54 print 'O: {0}\n'.format(fC0[1])
55 print 'The total energy of O2 is {0:1.3f}'.format(eO2)
56 print 'The forces (eV/angstrom) on the atoms in O2 are'
57 print 'O: {0}'.format(fO2[0])
58 print 'O: {0}\n'.format(fO2[1])
59 print 'The total energy of CO2 is {0:1.3f}'.format(eCO2)
60 print 'The forces (eV/angstrom) on the atoms in CO2 are'
61 print 'O: {0}'.format(fCO2[0])
62 print 'C: {0}'.format(fCO2[1])
63 print 'O: {0}\n'.format(fCO2[2])
64 print 'The reaction energy is {0:1.3f}'.format(re)

```

The total energy of C0 is -15.168
 The forces (eV/angstrom) on the atoms in C0 are
 C: [-0.033 0. 0.]
 O: [0.033 0. 0.]

The total energy of O2 is -8.719
 The forces (eV/angstrom) on the atoms in O2 are
 O: [0.019 0. 0.]
 O: [-0.019 0. 0.]

The total energy of CO2 is -23.508
 The forces (eV/angstrom) on the atoms in CO2 are

```

O: [-0.015  0.      0.   ]
C: [ 0.   0.   0.]
O: [ 0.015  0.      0.   ]

```

The reaction energy is -3.980

1.2 Convergence test

Repeat the previous problem at 350, 450, and 500 eV. Reoptimize the geometry at each ENCUT value. Compare (in a graph) the convergence of the total energy of each species with the convergence of the reaction energy. Which converges faster?

```

1  from ase import Atoms, Atom
2  from jasp import *
3  import numpy as np
4  np.set_printoptions(precision=3, suppress=True)
5
6  CO = Atoms([Atom('C', (0, 0, 0)),
7              Atom('O', (1.2, 0, 0))],
8             cell=(10, 10, 10))
9  O2 = Atoms([Atom('O', (0, 0, 0)),
10             Atom('O', (1.23, 0, 0))],
11            cell=(9.8, 9.9, 10))
12  CO2 = Atoms([Atom('O', (0, 0, 0)),
13              Atom('C', (1.16, 0, 0)),
14              Atom('O', (2.32, 0, 0))],
15             cell=(10, 10, 10))
16  dirs = ('/e350', '/e450', '/e500')
17  cuts = (350, 450, 500)
18  eCOs = []
19  eO2s = []
20  eCO2s = []
21
22  for d, cut in zip(dirs, cuts):
23      with jasp('prob1b' + d + '/CO',
24              xc='PBE', lreal=False,
25              encut=cut, prec='Accurate',
26              kpts=(1, 1, 1), ismear=1, sigma=0.05,
27              ibrion=1, nsw=50, ediffg=-0.05, isif=2,
28              atoms=CO) as COcalc:
29          try:
30              eCO = CO.get_potential_energy()
31              eCOs.append(eCO)
32          except:
33              pass
34      with jasp('prob1b' + d + '/O2',
35              xc='PBE', lreal=False,
36              encut=cut, prec='Accurate',
37              kpts=(1, 1, 1), ismear=1, sigma=0.05,
38              ibrion=1, nsw=50, ediffg=-0.05, isif=2,
39              atoms=O2) as O2calc:
40          try:
41              eO2 = O2.get_potential_energy()
42              eO2s.append(eO2)
43          except:
44              pass
45      with jasp('prob1b' + d + '/CO2',
46              xc='PBE', lreal=False,
47              encut=cut, prec='Accurate',
48              kpts=(1, 1, 1), ismear=1, sigma=0.05,
49              ibrion=1, nsw=50, ediffg=-0.05, isif=2,

```

```

50         atoms=C02) as C02calc:
51     try:
52         eC02 = C02.get_potential_energy()
53         eC02s.append(eC02)
54     except:
55         pass
56
57 import matplotlib.pyplot as plt
58 from matplotlib.ticker import ScalarFormatter
59 import numpy as np
60
61 eC0s = np.array(eC0s)
62 eO2s = np.array(eO2s)
63 eC02s = np.array(eC02s)
64
65 fig = plt.figure(1)
66 axC0 = fig.add_subplot(221)
67 axC0.plot(cuts, eC0s, marker='o')
68 axC0.set_title('$\mathdefault{CO}$')
69 axC0.set_xlim((300, 550))
70 axC0.set_ylim((-14.81, -14.75))
71 axC0.set_xticklabels(())
72 axC0.yaxis.set_major_formatter(ScalarFormatter(useOffset=False))
73 axC0.set_ylabel('Total Energy (eV/atom)')
74
75 axO2 = fig.add_subplot(222)
76 axO2.plot(cuts, eO2s, marker='o')
77 axO2.set_title('$\mathdefault{O_2}$')
78 axO2.set_xlim((300, 550))
79 axO2.set_ylim((-8.76, -8.70))
80 axO2.set_xticklabels(())
81 axO2.yaxis.set_major_formatter(ScalarFormatter(useOffset=False))
82
83 axC02 = fig.add_subplot(223)
84 axC02.plot(cuts, eC02s, marker='o')
85 axC02.set_title('$\mathdefault{CO_2}$')
86 axC02.set_xlim((300, 550))
87 axC02.set_xlabel('Kinetic Energy Cutoff (eV)')
88 axC02.set_ylabel('Total Energy (eV/atom)')
89 axC02.xaxis.set_major_formatter(ScalarFormatter(useOffset=False))
90 axC02.yaxis.set_major_formatter(ScalarFormatter(useOffset=False))
91
92 axrxn = fig.add_subplot(224)
93 axrxn.plot(cuts, eC02s - eC0s - 0.5*eO2s, marker='o')
94 axrxn.set_title(r'$\Delta H\mathdefault{(CO + \frac{1}{2} O_2} \rightarrow \mathdefault{CO_2})}$')
95 axrxn.set_xlim((300, 550))
96 axrxn.set_ylim((-3.84, -3.78))
97 axrxn.set_xlabel('Kinetic Energy Cutoff (eV)')
98 axrxn.xaxis.set_major_formatter(ScalarFormatter(useOffset=False))
99 axrxn.yaxis.set_major_formatter(ScalarFormatter(useOffset=False))
100 fig.tight_layout()
101 plt.savefig('1b.png')
102 plt.show()

```

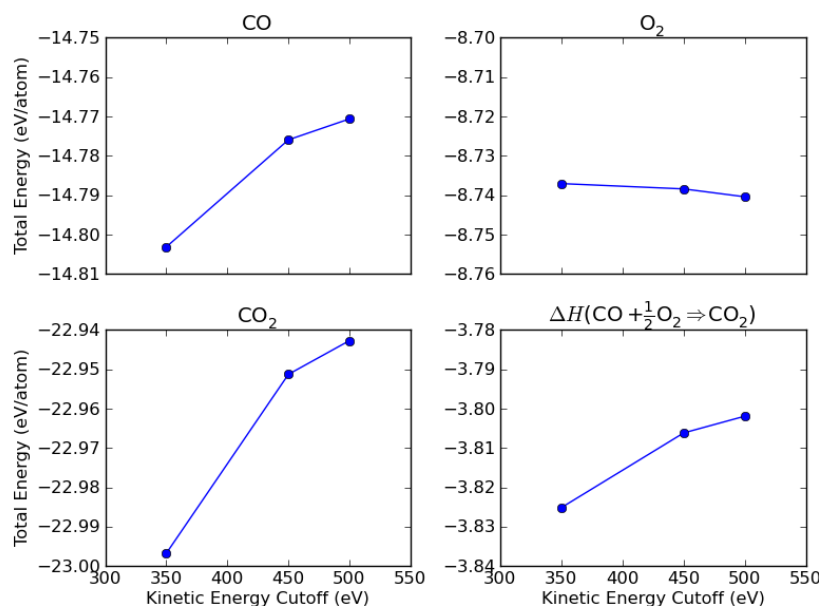


Figure 1: Convergence of CO, O₂, CO₂, and the reaction enthalpy of $\text{CO} + \frac{1}{2}\text{O}_2 \rightarrow \text{CO}_2$ with respect to plane wave cutoff energy

The total energy of the oxygen molecule converges the fastest

2 Zero-point energy corrections

2.1 Compute vibrational modes for CO, CO₂ and O₂

Compute the vibrational modes of each molecule in the CO oxidation reaction. Do this at 350 eV cutoff energy only. Prepare a table of the vibrational modes for molecule.

```

1 import os
2 import sys
3 from ase.calculators.vasp import Vasp
4 import ase.units
5 from jasp import *
6
7 # Since we wanted relaxed molecules for these calculations, we can take
8 # these geometries from the previous problem. Note, I could not seem to use
9 # the jasp.get_atoms(), so I had to resort to using the Vasp calculator.
10 # I wonder how you typically do this with jasp.
11
12 CWD = os.getcwd()
13 os.chdir(CWD + '/prob1b/e350/CO')
14 CO = Vasp(restart=True)
15 CO = CO.get_atoms()
16 CO.center()
17 os.chdir(CWD + '/prob1b/e350/O2')
18 O2 = Vasp(restart=True)
19 O2 = O2.get_atoms()
20 O2.center()
21 os.chdir(CWD + '/prob1b/e350/CO2')
22 CO2 = Vasp(restart=True)
23 CO2 = CO2.get_atoms()

```

```

24 CO2.center()
25 os.chdir(CWD)
26
27 # Now we're ready to perform the vibrational calculations
28 with jasp('prob2a/CO',
29         xc='PBE', lreal=False,
30         encut=350, prec='Accurate', ediff=1e-8,
31         kpts=(1, 1, 1), ismear=0, sigma=0.05,
32         ibrion=6, nsw=1, potim=0.015, nfree=2,
33         atoms=CO) as calcC0:
34     try:
35         C0.get_potential_energy()
36         energies, modes = calcC0.get_vibrational_modes()
37         print 'Energies of CO\n====='
38         for i, e in enumerate(energies):
39             print '{0:02d}: {1} eV'.format(i, e)
40     except:
41         pass
42 with jasp('prob2a/O2',
43         xc='PBE', lreal=False,
44         encut=350, prec='Accurate', ediff=1e-8,
45         kpts=(1, 1, 1), ismear=0, sigma=0.05,
46         ibrion=6, nsw=1, potim=0.015, nfree=2,
47         atoms=O2) as calcO2:
48     try:
49         O2.get_potential_energy()
50         energies, modes = calcO2.get_vibrational_modes()
51         print '\nEnergies of O2\n====='
52         for i, e in enumerate(energies):
53             print '{0:02d}: {1} eV'.format(i, e)
54     except:
55         pass
56 with jasp('prob2a/CO2',
57         xc='PBE', lreal=False,
58         encut=350, prec='Accurate', ediff=1e-8,
59         kpts=(1, 1, 1), ismear=0, sigma=0.05,
60         ibrion=6, nsw=1, potim=0.015, nfree=2,
61         atoms=CO2) as calcC02:
62     try:
63         C02.get_potential_energy()
64         energies, modes = calcC02.get_vibrational_modes()
65         print '\nEnergies of CO2\n====='
66         for i, e in enumerate(energies):
67             print '{0:02d}: {1} eV'.format(i, e)
68     except:
69         pass

```

Energies of CO

=====

```

00: 0.261840727 eV
01: 0.003767323 eV
02: 0.003767323 eV
03: (3.0739e-05+0j) eV
04: (0.000943898+0j) eV
05: (0.000943898+0j) eV

```

Energies of O2

=====

```

00: 0.189490603 eV
01: 0.004093929 eV

```

```
02: 1e-09 eV
03: 0.0 eV
04: (1e-09+0j) eV
05: (0.006638148+0j) eV
```

Energies of CO2

=====

```
00: 0.291924562 eV
01: 0.16318552 eV
02: 0.078492458 eV
03: 0.078492458 eV
04: 0.004836504 eV
05: 0.004836504 eV
06: (4.1677e-05+0j) eV
07: (5.9833e-05+0j) eV
08: (5.9833e-05+0j) eV
```

2.2 Compute the CO oxidation reaction energy with zero-point energy corrections.

Compare the reaction energy with and without the zero-point energy correction.

```
1  from jasp import *
2  import numpy as np
3  c = 3e10 # speed of light cm/s
4  h = 4.135667516e-15 # eV/s
5
6  # Get the vibrational energies from problem 2a. Get the total energies from
7  # problem 1b at 350 eV.
8
9  with jasp('prob2a/CO') as calc:
10     COfreq = calc.get_vibrational_frequencies()
11  with jasp('prob1b/e350/CO') as calc:
12     atoms = calc.get_atoms()
13     COe = atoms.get_potential_energy()
14  for f in COfreq:
15     if not isinstance(f, float):
16         continue
17     nu = f*c
18     COe += 0.5*h*nu
19  with jasp('prob2a/O2') as calc:
20     O2freq = calc.get_vibrational_frequencies()
21  with jasp('prob1b/e350/O2') as calc:
22     atoms = calc.get_atoms()
23     O2e = atoms.get_potential_energy()
24  for f in O2freq:
25     if not isinstance(f, float):
26         continue
27     nu = f*c
28     O2e += 0.5*h*nu
29  with jasp('prob2a/CO2') as calc:
30     CO2freq = calc.get_vibrational_frequencies()
31  with jasp('prob1b/e350/CO2') as calc:
32     atoms = calc.get_atoms()
33     CO2e = atoms.get_potential_energy()
34  for f in CO2freq:
35     if not isinstance(f, float):
```

```

36         continue
37     nu = f*c
38     CO2e += 0.5*h*nu
39 s = 'The reaction energy for CO oxidation with zero point contributions is {0}'
40 print s.format(CO2e - COe - 0.5*O2e)

```

The reaction energy for CO oxidation with zero point contributions is -3.69719889405

2.3 Compare your computed energy to a value from the literature.

Provide a reference for your literature value.

3 Plot the electron density of the CO2 molecule.

Include the figure in your homework.

```

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  # Lets first get the relaxed CO at 500 eV plane wave cutoff, center it,
8  # and recalculate the electron density in the centered cell
9
10 with jasp('prob1b/e500/CO') as calc:
11     CO = calc.get_atoms()
12     CO.center()
13 with jasp('prob3a/CO-centered',
14         xc='PBE', lreal=False,
15         encut=500, prec='Accurate',
16         kpts=(1, 1, 1), ismear=1, sigma=0.05,
17         atoms=CO) as calc:
18     CO.get_potential_energy()
19     x, y, z, cd = calc.get_charge_density()
20
21 mlab.figure(bgcolor=(1, 1, 1))
22 # plot the atoms as spheres
23 for atom in CO:
24     mlab.points3d(atom.x,
25                 atom.y,
26                 atom.z,
27                 scale_factor=vdw_radii[atom.number]/5.,
28                 resolution=20,
29                 # a tuple is required for the color
30                 color=tuple(cpk_colors[atom.number]),
31                 scale_mode='none')
32
33 # draw the unit cell - there are 8 corners, and 12 connections
34 a1, a2, a3 = CO.get_cell()
35 origin = [0, 0, 0]
36 cell_matrix = [[origin, a1],
37                [origin, a2],
38                [origin, a3],
39                [a1, a1 + a2],
40                [a1, a1 + a3],
41                [a2, a2 + a1],
42                [a2, a2 + a3],
43                [a3, a1 + a3],
44                [a3, a2 + a3],
45                [a1 + a2, a1 + a2 + a3],

```



```

46         [a2 + a3, a1 + a2 + a3],
47         [a1 + a3, a1 + a3 + a2]]
48
49 for p1, p2 in cell_matrix:
50     mlab.plot3d([p1[0], p2[0]], # x-positions
51                [p1[1], p2[1]], # y-positions
52                [p1[2], p2[2]], # z-positions
53                tube_radius=0.02)
54
55 # Now plot the charge density
56 mlab.contour3d(x, y, z, cd, transparent=True)
57
58 # this view was empirically found by iteration
59 mlab.view(azimuth=-90, elevation=90, distance='auto')
60
61 mlab.savefig('co-density.png')

```

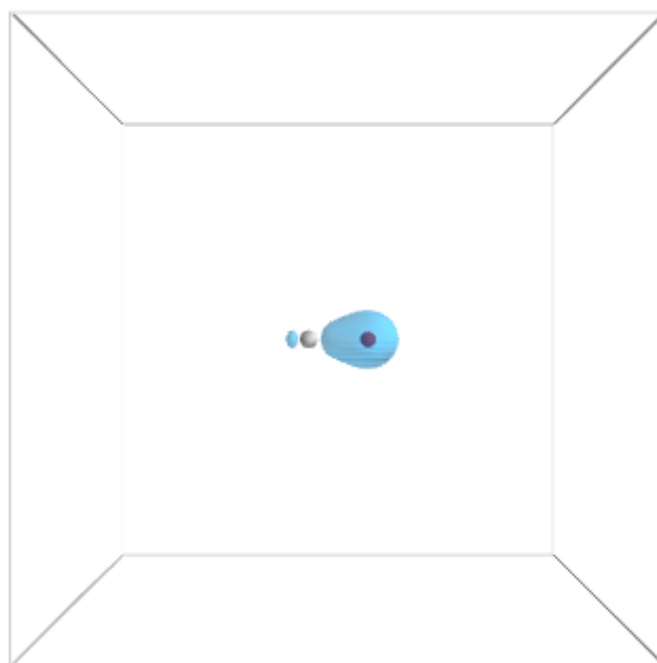


Figure 2: Charge density of CO