Homework 2
Computational Chemistry
(CBE 60553)

Prof. William F. Schneider

Due: *<2015-01-27 Tue>*

1 Lectures 1-2: Review of quantum mechanics

An electron is trapped in a one-dimensional box described by the potential (recall 1 bohr = 0.529177 Å, is the atomic unit of length):

$$V(x) = \begin{cases} 0, & -1 < x < 1 \text{ bohr} \\ \infty, & x \leq -1 \text{ or } x \geq 1 \text{ bohr} \end{cases} \tag{1}$$

(a) Using the energy expression given in class, calculate the ground state (n=1) energy of the electron, in Hartree (the atomic unit of energy), in eV, and in kJ mol^{-1}.

(b) We found in class that the ground-state wavefunction for this electron is $\psi_1(x) = cos(\pi x/2)$. Sketch this wavefunction and show that it obeys the proper boundary conditions, has zero nodes, and is normalized.

(c) Suppose you approximate the true ground-state wavefunction by $\phi_1(x) = 1 - x^2$. Calculate the expectation value of the energy (in Hartree) for this approximation. How does your answer compare to the energy you calculated in (a)?

(d) Can you guess an even better approximate wavefunction? Guess a candidate, evaluate the expectation value of its energy, and compare to question (c). Did you do any better than my guess?

1.1 Solution

1.1.1 a)

We have,

$E_n = n^2\pi^2\hbar^2/2m_eL^2$

```
1 import numpy as np
2
```

```
# In atomic units
m = 1
hbar =1
L = 2

E_atomic = np.pi ** 2 / (2 * 2 ** 2) # hartree
E_eV = E_atomic * 27.2114
E_kJ_mol = E_atomic * 2625.4996

print 'E = {0:1.3f} hartree = {1:1.3f} eV = {2:1.3f} kJ/mol'.format(E_atomic, E_eV, E_kJ_mol)
```

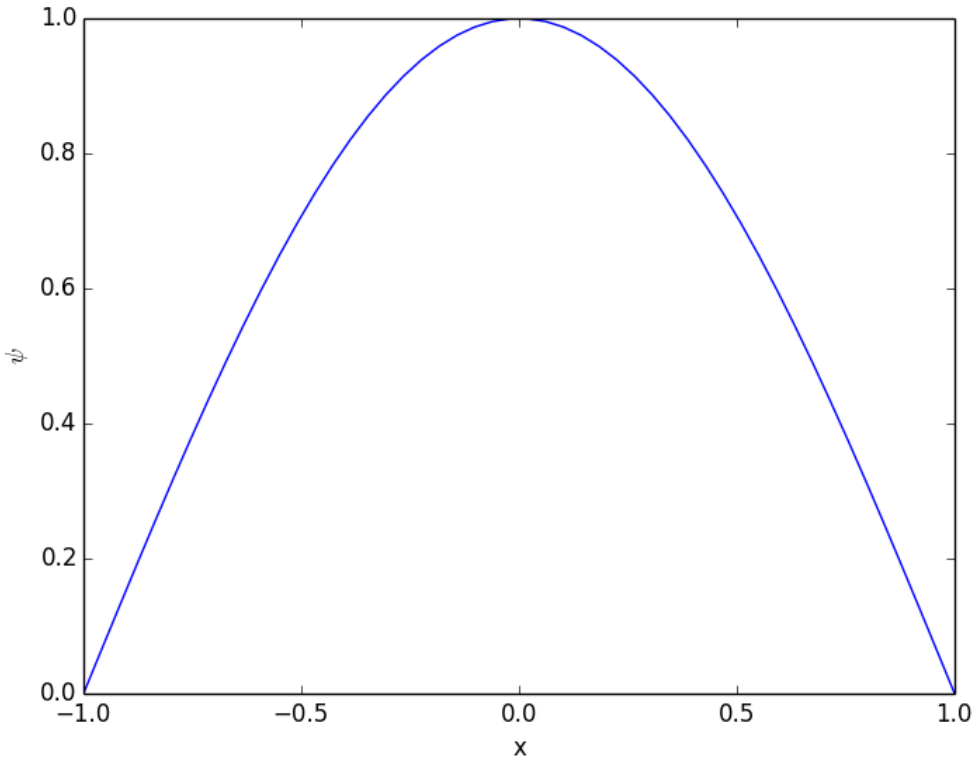
```
E = 1.234 hartree = 33.571 eV = 3239.080 kJ/mol
```

1.1.2 b)

The wavefunction, $\psi_1(x) = cos(\pi x/2)$, is plotted below. We can see that it obeys the boundary conditions. In the range, $-1 < x < 1$, ψ never equals zero, so there are zero nodes. Also, $\langle\psi|\psi\rangle = 1$, so the wavefunction is normalized.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import quad

x = np.linspace(-1,1)
psi = np.cos(np.pi * x / 2)

plt.plot(x,psi)
plt.xlabel('x')
plt.ylabel('$\psi$')
plt.savefig('images/1b.png')
plt.show()

# Normalizing
def integrand(x):
    return np.cos(np.pi * x / 2) ** 2

ans, err = quad(integrand, -1, 1)
print ans
```

```
1.0
```



1.1.3 c)

First we need to normalize the wavefunction, $\phi_1(x) = 1 - x^2$. The expectation value of the energy can then be calculated using the Hamiltonian operator as $\langle E \rangle = \langle \psi | H | \psi \rangle$.

```
import numpy as np
from scipy.integrate import quad


# Normalizing
def integrand(x):
    return (1 - x ** 2) ** 2

c, err = quad(integrand, -1, 1)

# The wavefunction can also be written as a polynomial
# (-1*x^2 + 0*x + 1) / sqrt(c)
p = 1 / np.sqrt(c) * np.array([-1, 0, 1])

def func(x):
    return -0.5 * 1 / np.sqrt(c) * (1 - x ** 2) * np.polyder(np.polyder(p))

E, err = quad(func, -1, 1)

print 'The expectation value of the energy is {0} hartree.'.format(E)
```

```
The expectation value of the energy is 1.25 hartree.
```

We see that the energy is within 1.3 % of part (a).

1.1.4 d)

You can guess any wavefunction here, as long as it is normalized. Ideally you would make linear combinations of two or more functions, e.g. $\psi_1(x) = c_1(1-x^2) + c_2(1-x^4)$.

2 Lectures 3: Many-electron atoms

Hartree's father performed the first calculations on multi-electron atoms by hand. Today those same calculations (much better ones, in fact) can be done in the blink of an eye on a computer. In this problem you will use a code first developed by Herman and Skillman in the 1960's to calculate the wavefunctions and energy of an atom using the Hartree-Fock-Slater (HFS) model, an early predecessor to DFT. The necessary software, rewritten in `C++`, is available at `/afs/crc.nd.edu/users/w/wschnei1/CBE547/fda.tar.gz`.

Copy the software to your home directory, unpack (`tar -xzf fda.tar.gz`), change into the directory (`cd fda`) and compile the code(`make fda`) to create the fda executable. Look at the `00README` file for information about the computer program and the format of the input. If you are brave, glance through the various source files (`*.cxx`) to get a sense of what the code is doing. Note that the code uses atomic units, Hartree for energy and bohr for distance.

(a) Run the `Ar.inp` example included in the directory (`fda Ar`). If all goes well, you should get an output file (`Ar.out`) and a dump file (`Ar.dmp`). Look at the `Ar.out` file to answer these questions:

- How many self-consistent field (SCF) iterations does the calculation take to converge?
- What is the final calculated HFS energy of the atom?
- What are the identities (1s, 2p, etc.) and energies of the occupied atomic orbitals?

(b) The fda code solves the HFS equations on a radial grid. The `Ar.dmp` file contains the radial grid values and the total charge density in two columns of length 300, followed by an output of each orbital on the same grid. Plot out the charge density and each of the orbitals.

(c) Choose one of the d block atoms. From the periodic table, figure out its electronic configuration and create an fda input file for it (follow the instructions in `00README` for how to specify the atomic number and the orbital occupancies of your atom). Run the fda calculation on your atom.

- What is the final calculated HFS energy of the atom? How does it compare to Ar?
- What are the identities (1s, 2p, etc.) and energies of the occupied atomic orbitals?

(d) The orbital energies are a rough approximation of the energy to remove an electron from that orbital. Use your result to estimate the first ionization energy of your atom. How does it compare with the experimental first ionization energy?

(e) You can also do calculations on anions or cations. Modify the input file for your atom by removing one of the valence electrons, to make it a cation. Rerun fda on the cation.

- How does the HFS energy of the cation compare to the neutral metal atom?
- Do the energies of the orbitals go up or down from the neutral to the cation?

- Do the electrons get closer to or further from the nucleus in the cation compared to the neutral? Use the expectation values of the distances from the nucleus (<r>) to answer the question.

(f) The difference in total energy between your neutral and cation calculations is another estimate of the first ionization energy of your atom. How does this estimate compare with experiment?

2.1 Solution

2.1.1 a)

It takes 29 iterations to converge. The final HFS energy is -526.8275 hartree. The orbital energies are tabulated below.

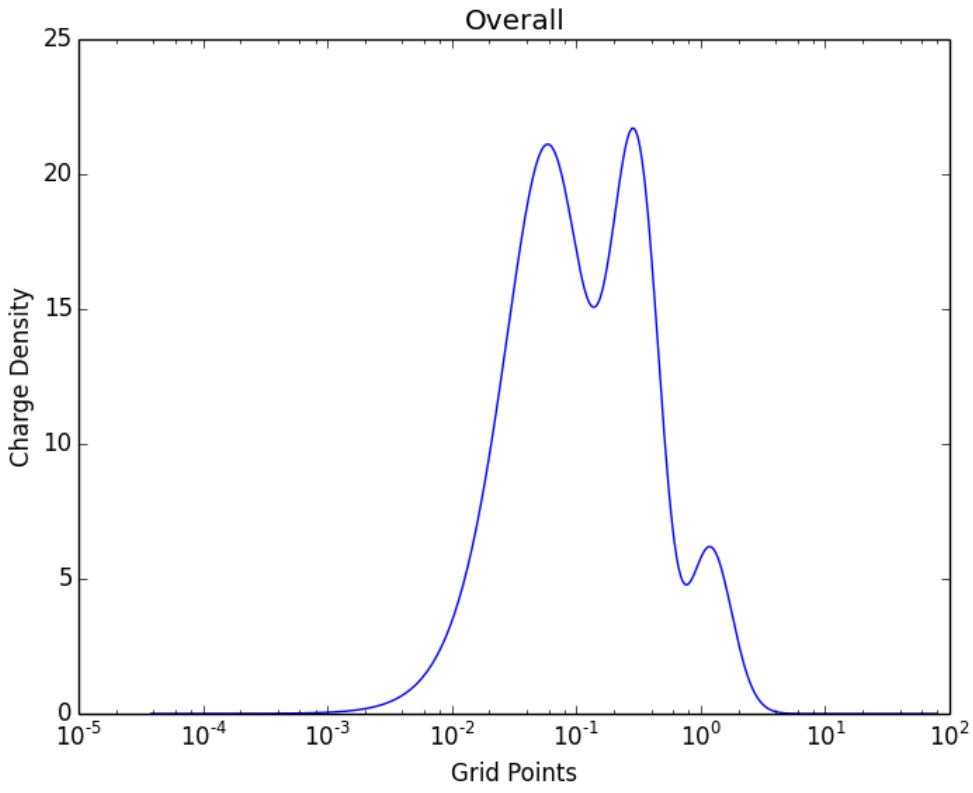
nl	E
1s	-116.9366
2s	-11.6037
2p	-9.2721
3s	-1.1022
3p	-0.5735

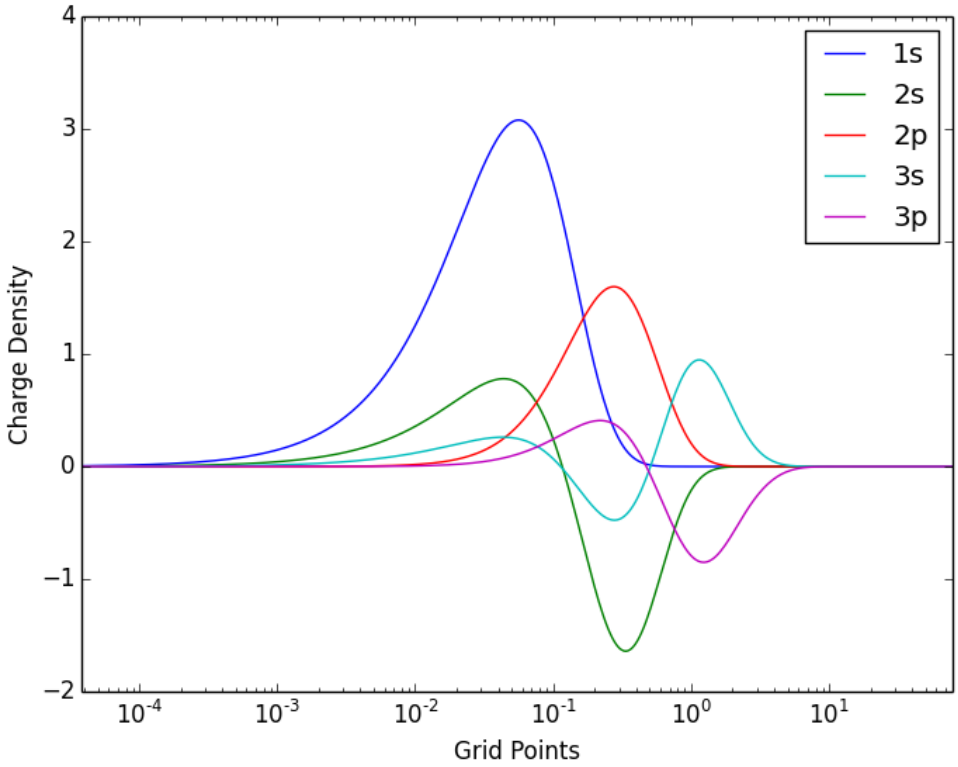
2.1.2 b)

The Ar charge densities should be easily plottable from the code block provided in the lab.

```
import matplotlib.pyplot as plt
import numpy as np

# Lets open the file in read mode
with open('FDA/Ar.dmp', 'r') as f:

    # Reading all the lines in the file
    # Each line is stored as an element of a list
    lines = f.readlines()

    # First we read the grid points and the total charge densities
    grid_points = []
    total_charge_densities = []

    for line in lines[3:303]:

        # Each is a string with two columns
        grid_point, tot_charge_density = line.split()

        # We need to convert each line to a float add it to our lists
        grid_points.append(float(grid_point))
        total_charge_densities.append(float(tot_charge_density))

    # Now the individual orbitals
    one_s_charge_density = [float(x) for x in lines[304:604]]
    two_s_charge_density = [float(x) for x in lines[605:905]]
    two_p_charge_density = [float(x) for x in lines[906:1206]]
    three_s_charge_density = [float(x) for x in lines[1207:1507]]
    three_p_charge_density = [float(x) for x in lines[1508:1808]]

plt.figure()
plt.semilogx(grid_points, total_charge_densities)
```

```
33  plt.xlabel('Grid Points')
34  plt.ylabel('Charge Density')
35  plt.title('Overall')
36  plt.savefig('images/Ar-overall-charge-density.png')
37
38  plt.figure()
39  plt.semilogx(grid_points, one_s_charge_density, label='1s')
40  plt.semilogx(grid_points, two_s_charge_density, label='2s')
41  plt.semilogx(grid_points, two_p_charge_density, label='2p')
42  plt.semilogx(grid_points, three_s_charge_density, label='3s')
43  plt.semilogx(grid_points, three_p_charge_density, label='3p')
44  plt.xlabel('Grid Points')
45  plt.ylabel('Charge Density')
46  plt.xlim(min(grid_points), max(grid_points))
47  plt.legend()
48  plt.savefig('images/Ar-orbital-charge-density.png')
49  plt.show()
```





2.1.3 c)

- Here is an example calculation for Zn. The total energy is -1779.5562 hartree, which is about 3.5 times the energy for Ar.
- The identities and the energies of the orbitals are below.

nl	occ	E	<r>
1s	2.00	-350.4349	0.0509
2s	2.00	-42.8635	0.2276
2p	6.00	-38.0715	0.1972
3s	2.00	-4.9959	0.6811
3p	6.00	-3.4181	0.7023
3d	10.00	-0.6922	0.8306
4s	2.00	-0.3265	2.6030

2.1.4 d)

The experimental ionization energy of Zn is 906 kJ/mol. From the table we can see that our first ionization energy is 0.3265 hartree = 857.23 kJ/mol, which is about 5% off from the experimental value.

2.1.5 e)

- The total energy after removing one of the cations is -1779.2279 hartree, about 0.3283 hartree more than the neutral Zn atom.
- The orbital energies are tabulated below. All the orbital energies seem to have decreased. The 3d and 4d orbitals move closer to the nucleus.

nl	occ	E	<r>
1s	2.00	-350.8245	0.0509
2s	2.00	-43.2466	0.2276
2p	6.00	-38.4550	0.1972
3s	2.00	-5.3839	0.6811
3p	6.00	-3.8058	0.7022
3d	10.00	-1.0763	0.8227
4s	1.00	-0.6508	2.3916

2.1.6 f)

The energy difference between the neutral atom and the cation is 0.3283 Hartree = 861.95 kJ/mol. This is almost the same as what we had earlier.