國立清華大學
化學工程學系 National Tsing Hua University
Department of Chemical Engineering

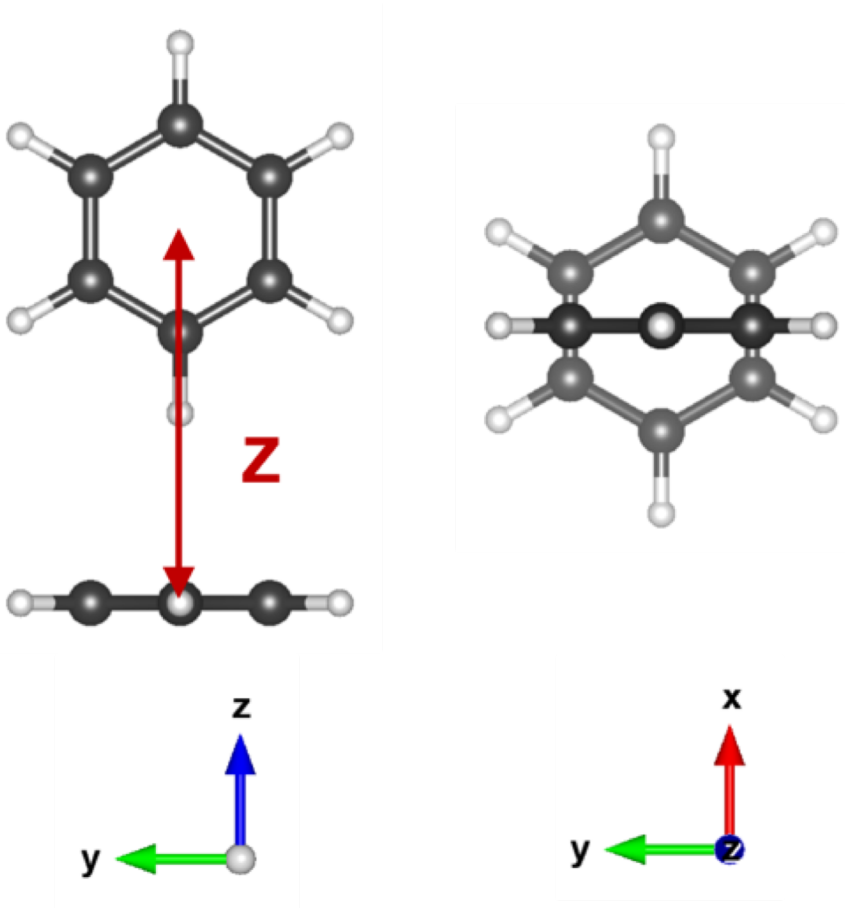
2022 Summer School

Electronic Structure Calculations Using GFN2-xTB

Prof. Kun-Han Lin (林昆翰助理教授)
Department of Chemical Engineering,
National Tsing Hua University, Taiwan

Homework 3

Q1: T-shaped benzene dimer



Tasks:

- Create a series of configurations of T-shaped dimers with different Z.
- Create the E_b-Z plot and determine the minimum.
- Evaluate the dimer interactions using multipole-expansion approximation.

Rotate the benzene – rotate.py

```
1  import numpy as np
2  from ase.io import read, write
3  from ase.build import rotate
4  from mpl_toolkits.mplot3d import Axes3D
5  import matplotlib.pyplot as plt
```

Import necessary library

- Numpy (numerical analysis)
- Ase (atomic simulation environment)
- Matplotlib (MATLAB-like plotting)

```
7  # Read the xyz as Atom object in ASE
8  xyz = read('xtbopt.xyz')
```

Read xtbopt.xyz (optimized benzene geometry using xTB) and create an Atoms object of ASE

What is an Atoms object? https://wiki.fysik.dtu.dk/ase/ase/atoms.html

The Atoms object

The Atoms object

The `Atoms` object is a collection of atoms. Here is how to define a CO molecule:

```
from ase import Atoms
d = 1.1
co = Atoms('CO', positions=[(0, 0, 0), (0, 0, d)])
```

Here, the first argument specifies the type of the atoms and we used the `positions` keywords to specify their positions. Other possible keywords are: `numbers`, `tags`, `momenta`, `masses`, `magmoms` and `charges`.

Extract information from the Atoms object

The get method

`get_atomic_numbers()`	`set_atomic_numbers()`
`get_initial_charges()`	`set_initial_charges()`
`get_charges()`	
`get_chemical_symbols()`	`set_chemical_symbols()`
`get_initial_magnetic_moments()`	`set_initial_magnetic_moments()`
`get_magnetic_moments()`	
`get_masses()`	`set_masses()`
`get_momenta()`	`set_momenta()`
`get_forces()`	
`get_positions()`	`set_positions()`
`get_potential_energies()`	
`get_scaled_positions()`	`set_scaled_positions()`
`get_stresses()`	
`get_tags()`	`set_tags()`
`get_velocities()`	`set_velocities()`

Align the plane norm to z-axis

```
10  # Get the moment of inertia of the compound
11  I = xyz.get_moments_of_inertia(vectors=True)
12
13  # Get the center of mass before translation
14  COM = xyz.get_center_of_mass()
15  print('Original center of mass: ',COM)
16
17  # Get the center of mass after translation COM to (0,0,0)
18  xyz.translate(-COM)
19  COM = xyz.get_center_of_mass()
20  print('Center of mass after translation: ', COM)
21
22  # Align the shortest molecular axis to z-axis
23  xyz.rotate(I[1][2],'z','COM', rotate_cell=False)
```

Obtain the moments of inertia of the benzene.

Obtain the COM of the benzene.

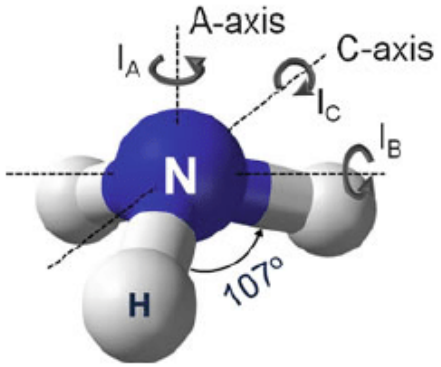
Move the COM to the origin (0,0,0) using translate. And print the new COM to check the new COM.

Rotate the benzene so that its plane norm I[1][2] align with the z-axis.

Print the moments of inertia

Shortest molecular axis with the largest eigen value. Why?

```
(array([ 87.45954357, 87.4597337 , 174.91927726]),
 array([[ 0.19390466, -0.06025610,  0.97916811],
        [-0.98017909,  0.02942734,  0.19591576],
        [-0.04061943, -0.99774908, -0.05335567]]))
```



It's a tuple object with the length of 2 (like a 2x1 vector).

The first entry is an array containing I_A, I_B and I_C.

The second entry is an array containing three arrays (eigen vectors) corresponding to the three eigen values (I_A, I_B and I_C).

Align the plane norm to z-axis

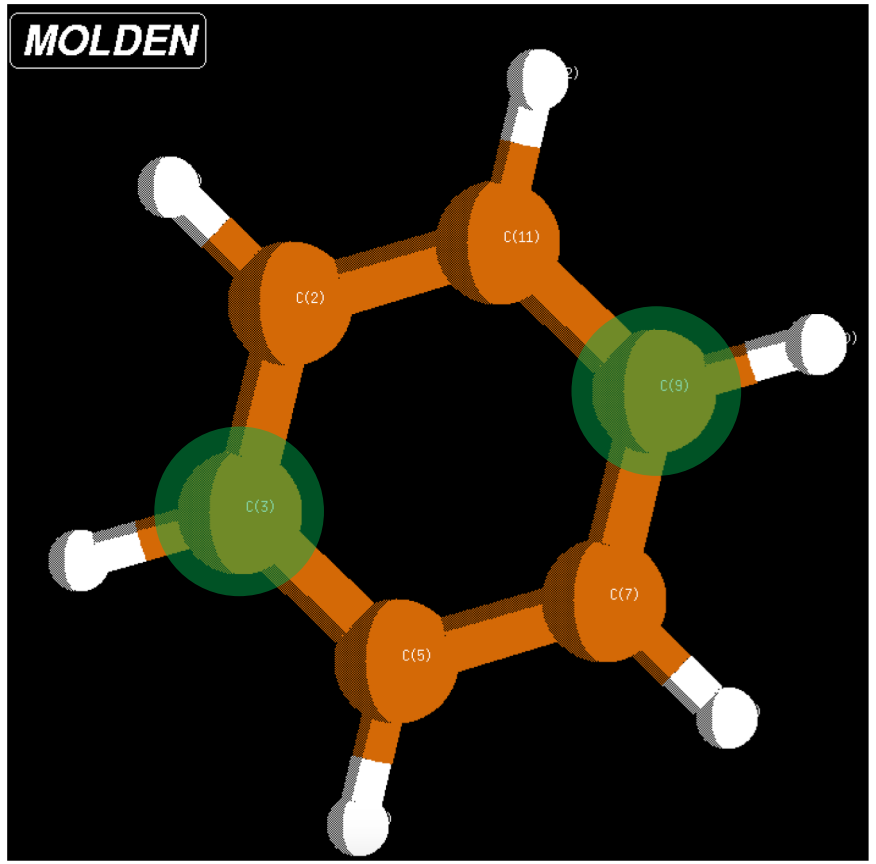
```
25  C3 = xyz.get_positions()[2]
26  C9 = xyz.get_positions()[8]
27
28  v = C3 - C9
29
30  # Align the C3-C9 vector to x-axis
31  xyz.rotate(v,'x','COM', rotate_cell=False)
32
33  # Write out the final structure as new.xyz
34  xyz.write('aligned_xz.xyz')
```

Write out the aligned benzene as aligned_xz.xyz

Obtain Cartesian coordinates of two carbons at the *para* position.

The vector C3 – C9.

Align v with the x-axis.



Generate the dimer stuctures – gen_dimer.py

```
1  import sys
2  import numpy as np
3  from ase.io import read, write
4  from ase.build import rotate
```

Import necessary library

- Numpy (numerical analysis)
- Ase (atomic simulation environment)
- sys (the sys.argv can be used to pass the variable to python script)

For example:

```
1  import sys
2
3  print (sys.argv[0])
4  print (sys.argv[1])
```

The name of your python code
First string you enter (separated by a space)

python test.py hello

Outeput:

```
test.py
hello
```

Therefore, sys.argv[1], sys.argv[2],…sys.argv[n] could be used to store the 1st, 2nd,…nth variables in your command line.

Generate the dimer stuctures – gen_dimer.py

```
6  # Read the xyz as Atom object in ASE
7  xyz = read('aligned_xz.xyz')
8  xyz_original = xyz.copy()
```

Read aligned.xyz and create an Atoms object of ASE.
Copy a xyz Atoms object and name it xyz_original.

```
10 # rotate the molecule along y axis by 90 degree
11 xyz.rotate(90.0,'y','COM', rotate_cell=False)
```

Rotate the aligned benzene along y-axis by 90 degree.

```
13 # Shift the molecules in z direction with P% of 5.0 Angstrom
14 v = np.array([0,0,5.0])*float(sys.argv[1])/100.0
15 xyz.translate(v)
```

Translate the rotated and aligned benzene by **v**.

Define a translation vector **v** = [0, 0, z], where the length could be controlled externally by sys.argv[1] when we run the code!

Generate the dimer stuctures – gen_dimer.py

```
17  xyz.extend(xyz_original)
18  # Write out the final structure
19  xyz.write(sys.argv[2])
```

Combine the original and the rotated benzene Atoms objects together using extend().

Write out the final dimer Atoms object into xyz file. The file name is controlled externally using sys.argv[2].

Create folders and inputs, and run! – create.sh

```
for a in 70 71 72 73 74 75 80 85 90 95 100 105 110 115 120 130 140
    150 160 170 180 190 200
do
    ## for each variable a ($a), create a folder
    mkdir $a

    ## copy the xtb running script to folder $a
    cp xtb.sh $a

    ## generate a dimer input xyz file in the folder. the distance is
        $a% of 5.0 Angstrom
    python gen_dimer.py $a ${a}/input.xyz

    ## enter the folder and run the script
    cd $a
    echo "Running $a now..."
    nohup bash xtb.sh &
    ## leave the $a folder and do the next loop
    cd ..
done
```

- The **for** loop in bash. It will repeatedly run the commands sandwiched between **do** and **done**.
- The repetition will go over all numbers defined for a.
- When you need the value of a, please use $a.

Create folders and inputs, and run! – create.sh

```
for a in 70 71 72 73 74 75 80 85 90 95 100 105 110 115 120 130 140
    150 160 170 180 190 200
do
    ## for each variable a ($a), create a folder
    mkdir $a

    ## copy the xtb running script to folder $a
    cp xtb.sh $a

    ## generate a dimer input xyz file in the folder. the distance is
        $a% of 5.0 Angstrom
    python gen_dimer.py $a ${a}/input.xyz

    ## enter the folder and run the script
    cd $a
    echo "Running $a now..."
    nohup bash xtb.sh &
    ## leave the $a folder and do the next loop
    cd ..
done
```

Create a folder with the name a.

Copy the xtb.sh script into folder a.

Run the gen_dimer.py code, given the 1st variable the distance; and the 2nd variable the file name.

Enter the folder a.

Run the xtb.sh script in background (&) without being interrupted (nohup).

Go back to the folder in the previous layer.

```
# Shift the molecules in z direction with P% of 5.0 Angstrom
v = np.array([0,0,5.0])*float(sys.argv[1])/100.0
xyz.translate(v)
```

```
xyz.write(sys.argv[2])
```

Extract and plot the results – plot_Eb.py

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

Import necessary library

- Numpy (numerical analysis)
- Matplotlib (MATLAB-like plotting)

```
4  # Figure Setting
5  # Edit the font, font size, and axes width
6  plt.rcParams['font.size'] = 14
7  plt.rcParams['axes.linewidth'] = 2
8  fig,ax = plt.subplots()
```

Set up the figure setting, font size and axes linewidth.

Create a figure object and axes objects.

matplotlib.pyplot.subplots

```
matplotlib.pyplot.subplots(nrows=1, ncols=1, *, sharex=False,
sharey=False, squeeze=True, subplot_kw=None, gridspec_kw=None, **fig_kw)
```

Extract the energy from monomer

```
10  # Read energy from monomer
11  name = 'monomer/xtb.out'
12  f = open(name,'r')
13  lines = f.readlines()
14  for l in lines:
15      if 'TOTAL ENERGY' in l:
16          tmp = l.split()
17          Em = float(tmp[3])
```

Specify the path to the xtb.out of the monomer computation.

Open the file use open().

Generate a list. In this list, each element is a line from the file f.

A for loop in python:
For each line in the list lines,
if the string 'TOTAL ENERGY' appears in the line
split the line by space and read the value.

For example, a file f like below:

```
I have a pen.
I have an apple.
Umm ugh!
Apple pen!
```

/ref_func_open.asp

The list lines by f.readlines() will be:

["I have a pen.", "I have an apple.", "Umm ugh!", "Apple pen!"]

Split a string using split()

For example, a line l like below:

```
| TOTAL ENERGY            -15.879640643370 Eh   |
```

l.split() will split the line by space, giving you a list like:

```
["|", "TOTAL", "ENERGY", "-15.879640643370", "Eh", "|"]
```

```
10  # Read energy from monomer
11  name = 'monomer/xtb.out'
12  f = open(name,'r')
13  lines = f.readlines()
14  for l in lines:
15      if 'TOTAL ENERGY' in l:
16          tmp = l.split()
17          Em = float(tmp[3])
```

That's why we take tmp[3], which is the 4th entry of the list tmp. And the float() function will make the string "-15.879640643370" into a float -15.879640643370.

Create lists and open files

```
19  # Read energy from dimer
20  D  = []
21  Eb = []
22  folder = [ 70, 71, 72, 73, 74, 75, 80, 85, 90, 95, 100, 105, 110, 115
        , 120, 130, 140, 150, 160, 170, 180, 190, 200]
23
24  # Write the Eb and D value into E-D.txt file
25  fout = open('E-D.txt','w')
```

Create empty lists for distance (D) and binding energy (Eb).

Create a list (folder). containing the names of folders.

Open the file E-D.txt, which will be used to write the E-D information into it.

Create lists and open files

```
27 for a in folder:
28     # Read xtb.out in 'a' folder
29     name = str(a)+'/xtb.out'
30     f = open(name,'r')
31     lines = f.readlines()
32     for l in lines:
33         if 'TOTAL ENERGY' in l:
34             tmp = l.split()
35             Ed = float(tmp[3])
36             D.append(float(a)*5.0/100.0)
37             Eb.append( (Ed - 2.0*Em)*27.2114 )
38             fout.write(str(float(a)*5.0/100.0)+'  '+str((Ed - 2.0*Em
                   )*27.2114) +'\n')
```

A for loop in python:
For each name in folder list,

1. Open the xtb.out file and read the lines
2. A for loop in python:
   For each line (l) in lines list,
   if the string 'TOTAL ENERGY' appears in the line
   1. split the line by space and read the value.
   2. store the distance into D by append()
   3. calculate binding energy and store into Eb by append()
   4. write out the distance and binding energy into fout using write().

Create lists and open files

```
42  # Plot the figure
43  ax.plot([0,np.max(folder)*5.0/100.0],[0,0],ls=':',c='black')
44  ax.scatter(D,Eb)
45
46  # set label for X and Y axis
47  ax.set_xlabel('Z (Å)',fontsize=18)
48  ax.set_ylabel(r'E$_b$ (eV)',fontsize=18)
49
50  # set x range
51  ax.set_xlim(2.0,np.max(folder)*5.0/100.0)
52
53  fig.tight_layout()
54  plt.show()
```

Plot the y=0 horizontal dotted line in black.

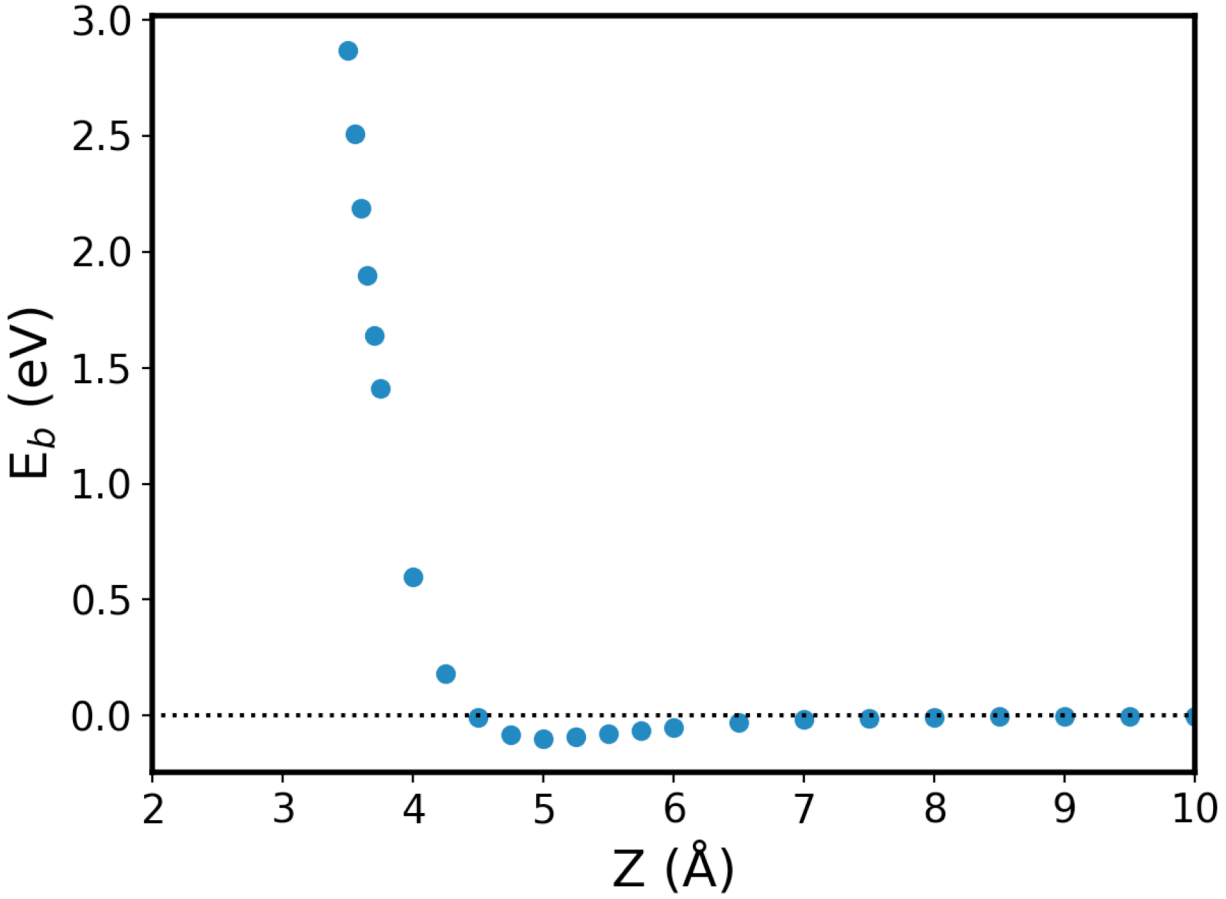
Plot the scatter plot using D as x-axis and Eb as y-axis.

Set the label for x and y axis.

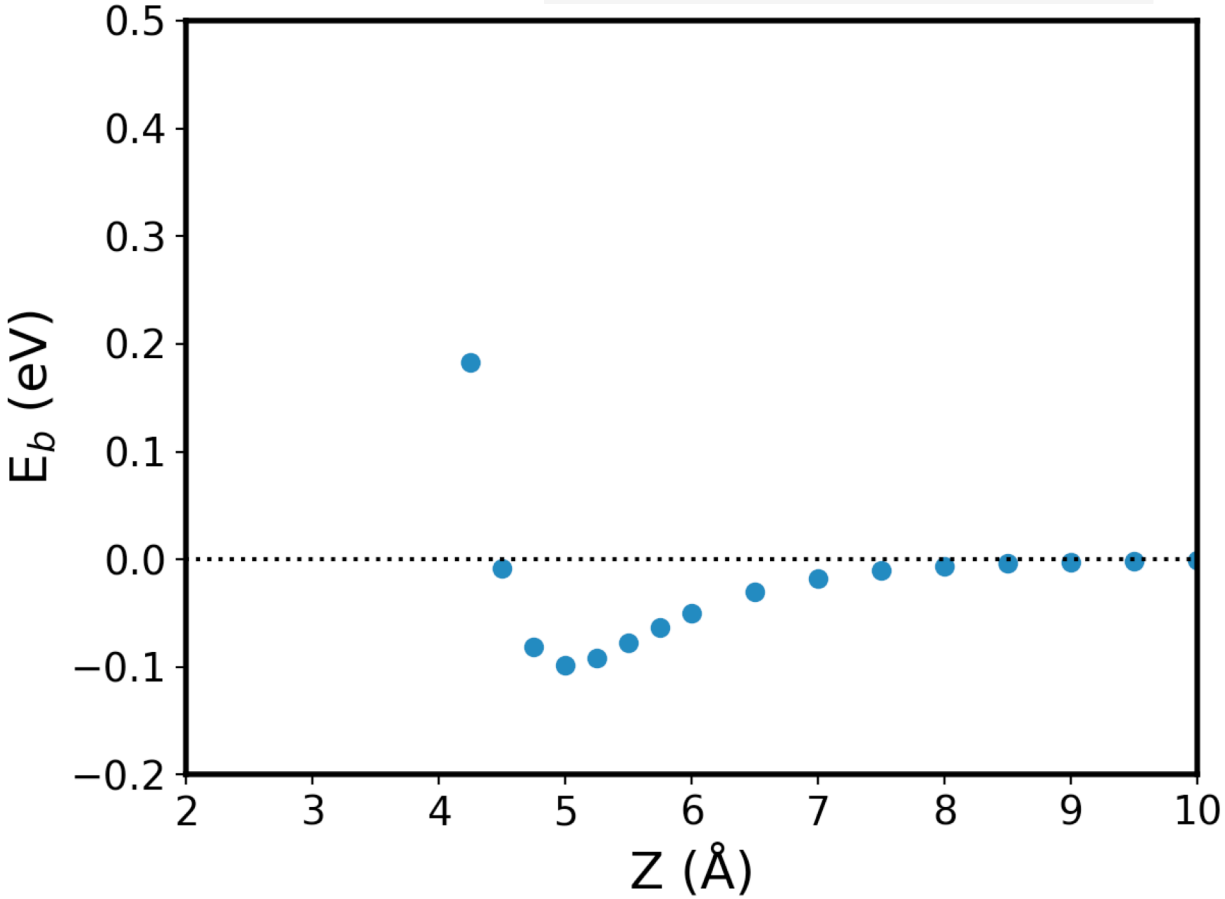
Set the range of display for x-axis.

Set the layout of figure and display the plot using show().

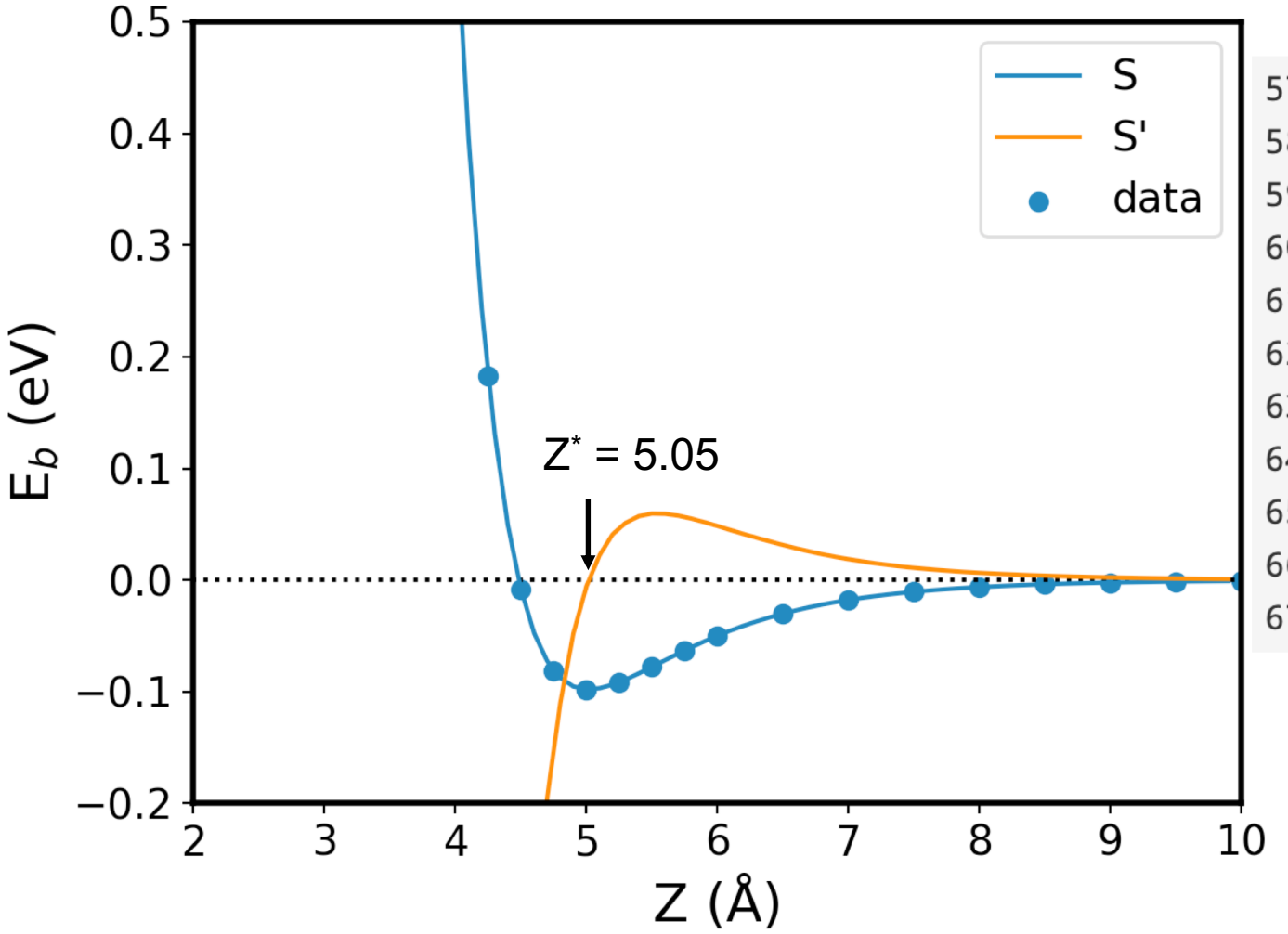
Eb-D curve




```
52  ax.set_ylim(-0.2,0.5)
```



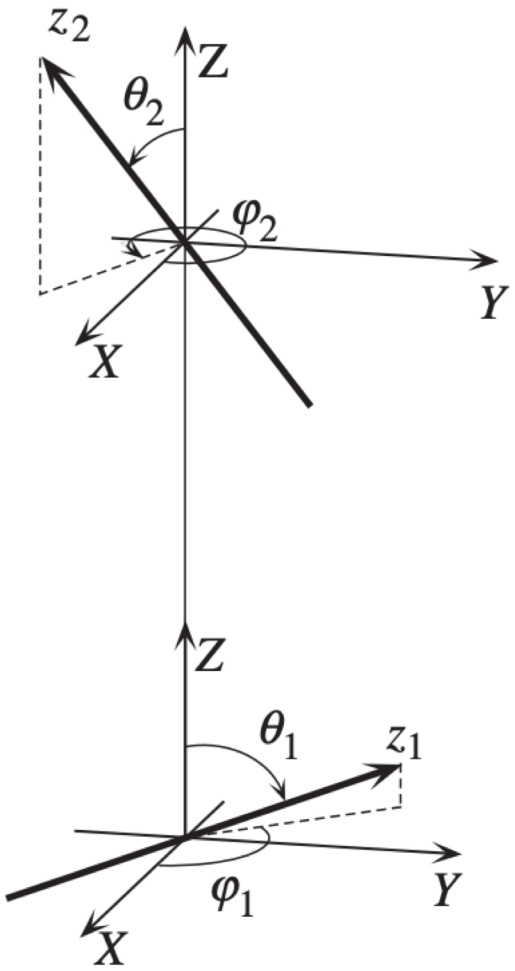
Cubic spline




```
57  from scipy.interpolate import CubicSpline
58  cs = CubicSpline(D, Eb)
59  xs = np.arange(0, np.max(folder)*5.0/100.0, 0.1)
60
61  ax.plot(xs, cs(xs), label="S")
62  ax.plot(xs, cs(xs,1), label="S'")
63  pold = 0
64  for n,p in enumerate(cs(xs,1)):
65      if n > 0 and pold*p < 0:
66          print (xs[n-1],xs[n], 0.5*(xs[n-1]+xs[n]))
67      pold = p
```

Run a dimer with 5.05 Å separation again!

Quadrupole-quadrupole interaction energy



$$U_{QQ} = \frac{Q^A Q^B}{4\pi\epsilon_0 R^5}\frac{3}{4}[1 - 5cos^2\theta_A - 5cos^2\theta_B - 15cos^2\theta_A cos^2\theta_B$$

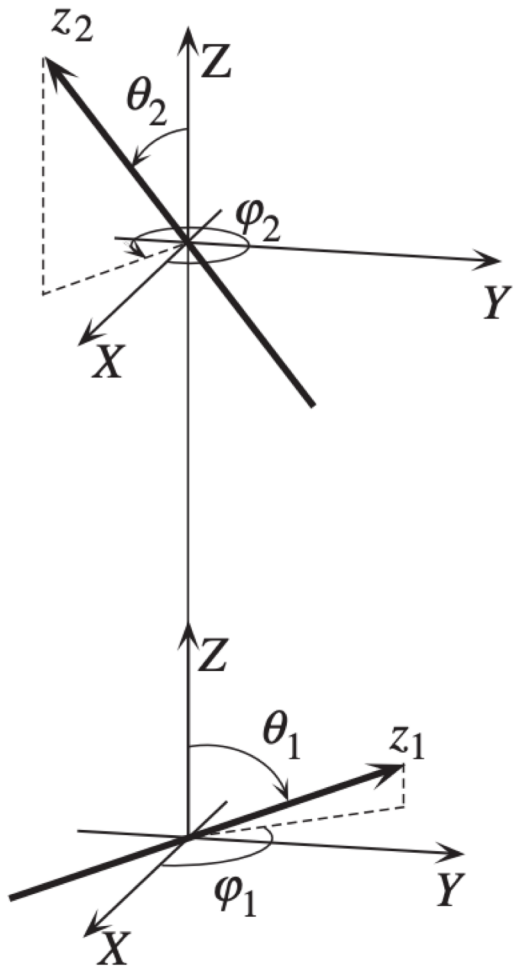
$$+2(4cos\theta_A cos\theta_B - sin\theta_A sin\theta_B cos\varphi)^2]$$

Q^A and Q^B are the Q_{zz} of molecule A and B (note: this is only true for linear or highly-symmetrical molecules and when the molecular axis is aligned with z-axis -> $Q_{xx} = Q_{yy} = -2Q_{zz}$; off-diagonal terms = 0)

$$U_{QQ} = \frac{Q^A Q^B}{4\pi\epsilon_0 R^5} 6$$ for parallel-stacked

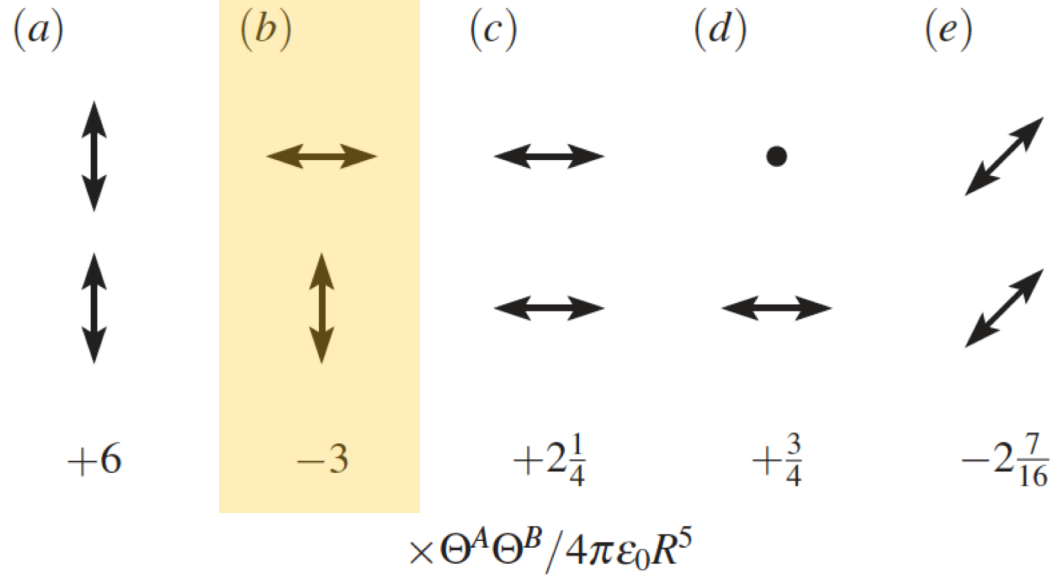
+6

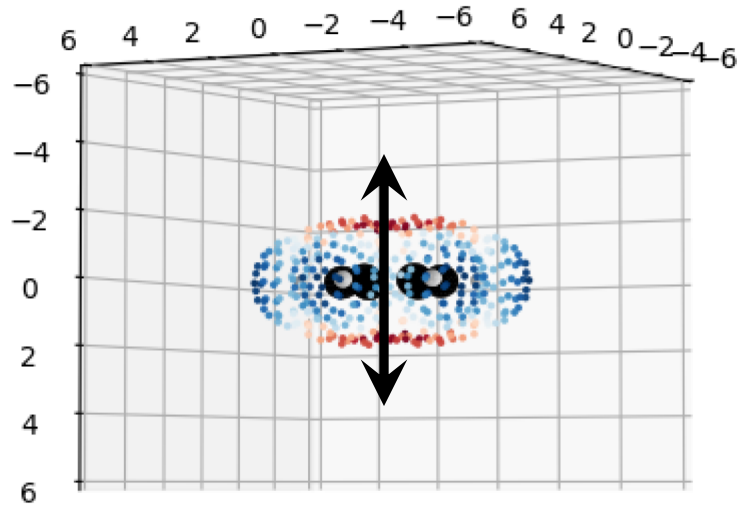
T-shaped benzene dimer



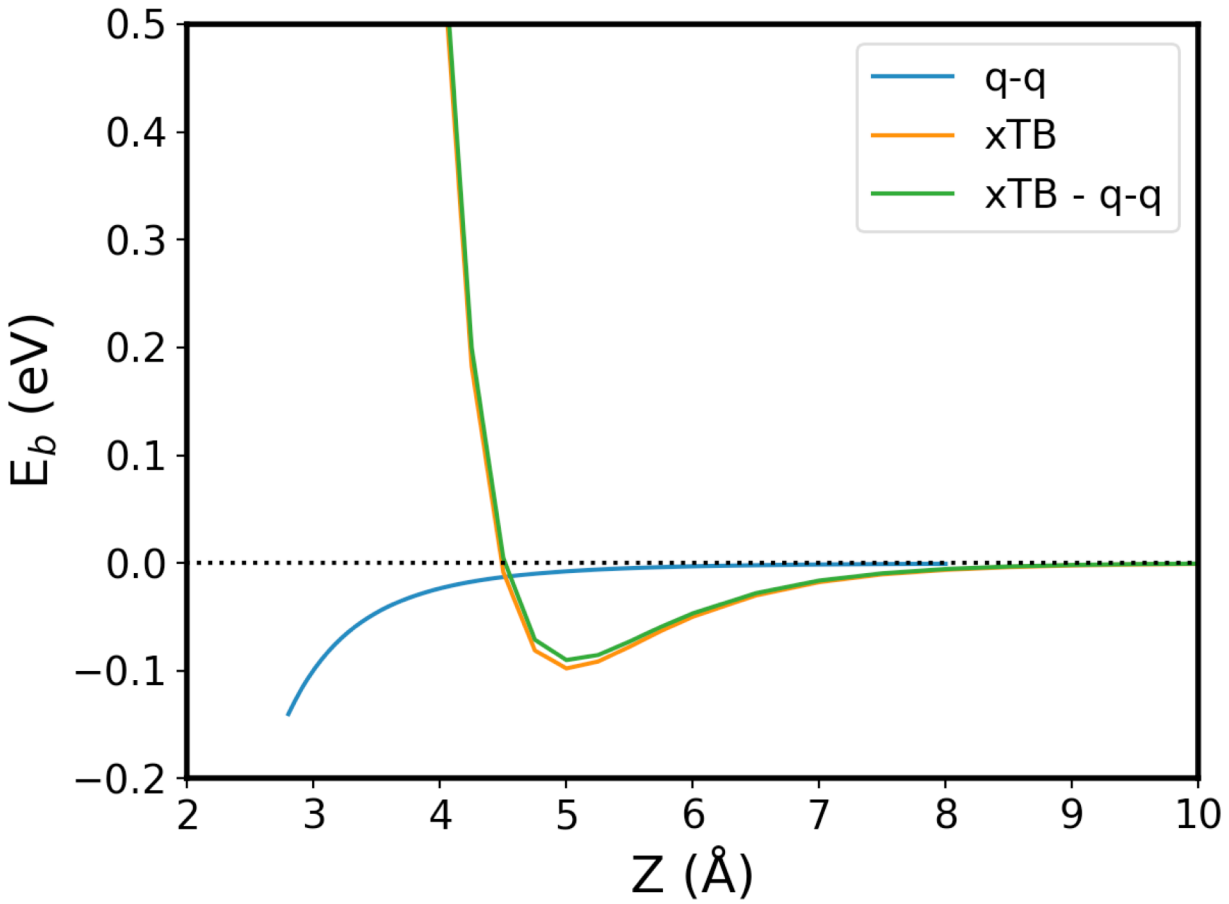
$\theta_A = 0, \theta_B = 90°$

φ doesn't matter in this case





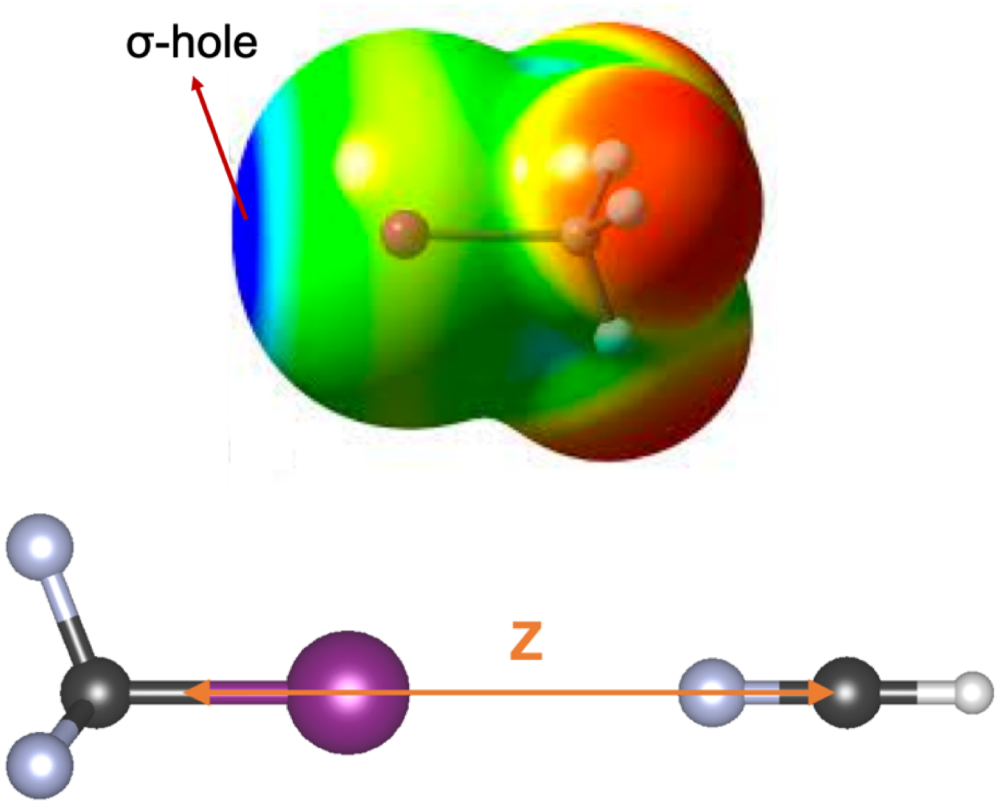
Comparison between xTB and q-q



The script is similar to plot_QQ.py

- It is attraction instead of repulsion (parallel stacked) now!
- The magnitude is still low.

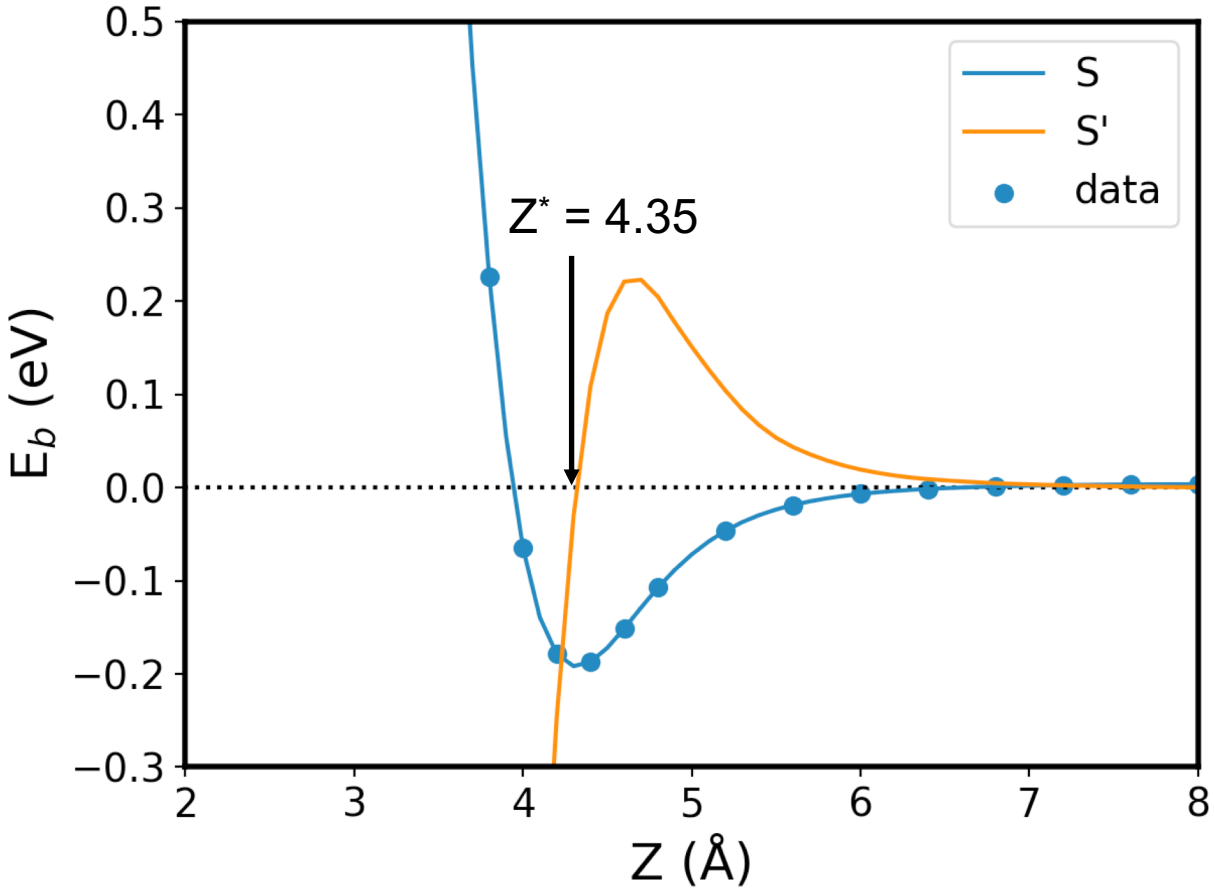
Q2: Halogen bond pair



Tasks:

- Create a series of configurations of ICF_3-NCH dimer with different Z.
- Create the E_b-Z plot and determine the minimum.
- Evaluate the dimer interactions using multipole-expansion approximation.
- Evaluate the dimer interactions using atomic charge model.

Eb-D curve



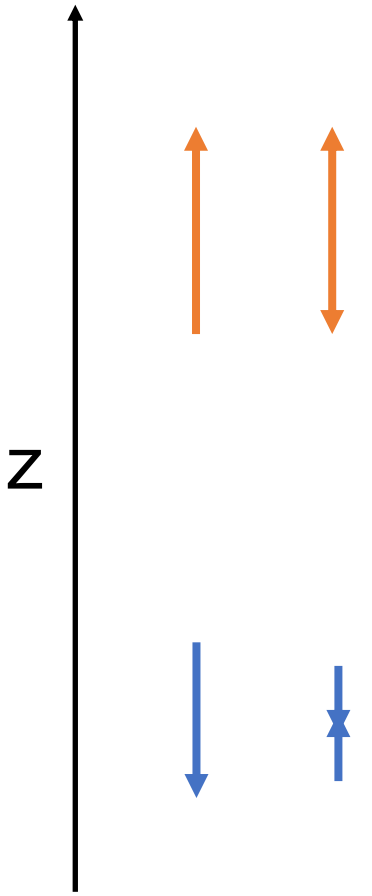
Dipole and quadrupole of ICF_3 and NCH

NCH

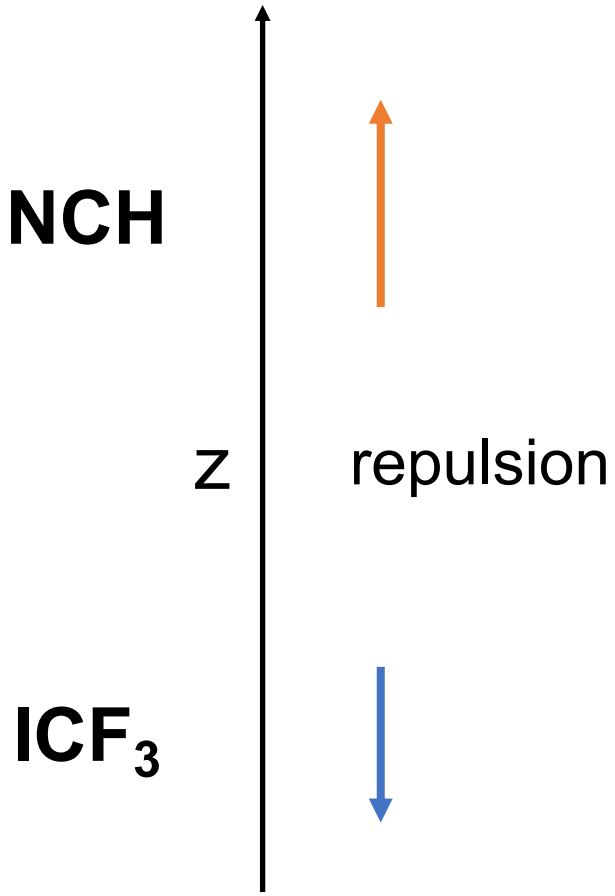
```
molecular dipole:
                x           y           z       tot (Debye)
 q only:       -0.000      -0.000       0.505
   full:       -0.000      -0.000       1.031       2.622
molecular quadrupole (traceless):
                xx          xy          yy          xz          yz          zz
 q only:      -4.149       0.000      -4.149      -0.000      -0.000       8.299
  q+dip:      -9.040       0.000      -9.040      -0.000      -0.000      18.079
   full:      -8.090      -0.000      -8.090      -0.000      -0.000      16.179
```

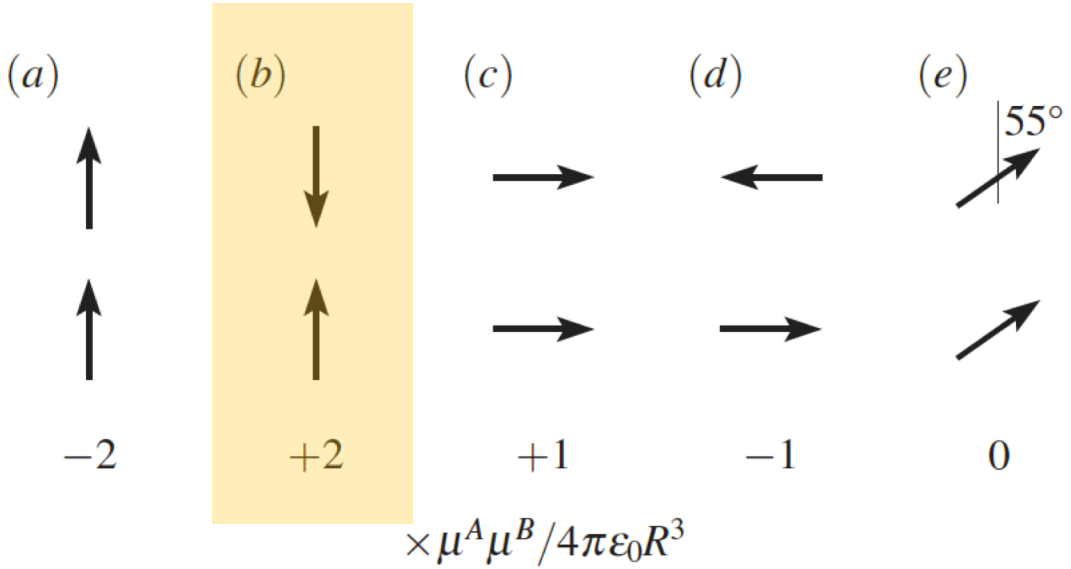
ICF_3

```
molecular dipole:
                x           y           z       tot (Debye)
 q only:       -0.000      -0.000       0.387
   full:       -0.000      -0.000      -0.328       0.835
molecular quadrupole (traceless):
                xx          xy          yy          xz          yz          zz
 q only:       0.957      -0.000       0.957      -0.000       0.000      -1.914
  q+dip:       1.353       0.000       1.353      -0.000       0.000      -2.706
   full:       0.510       0.000       0.510      -0.000       0.000      -1.020
```



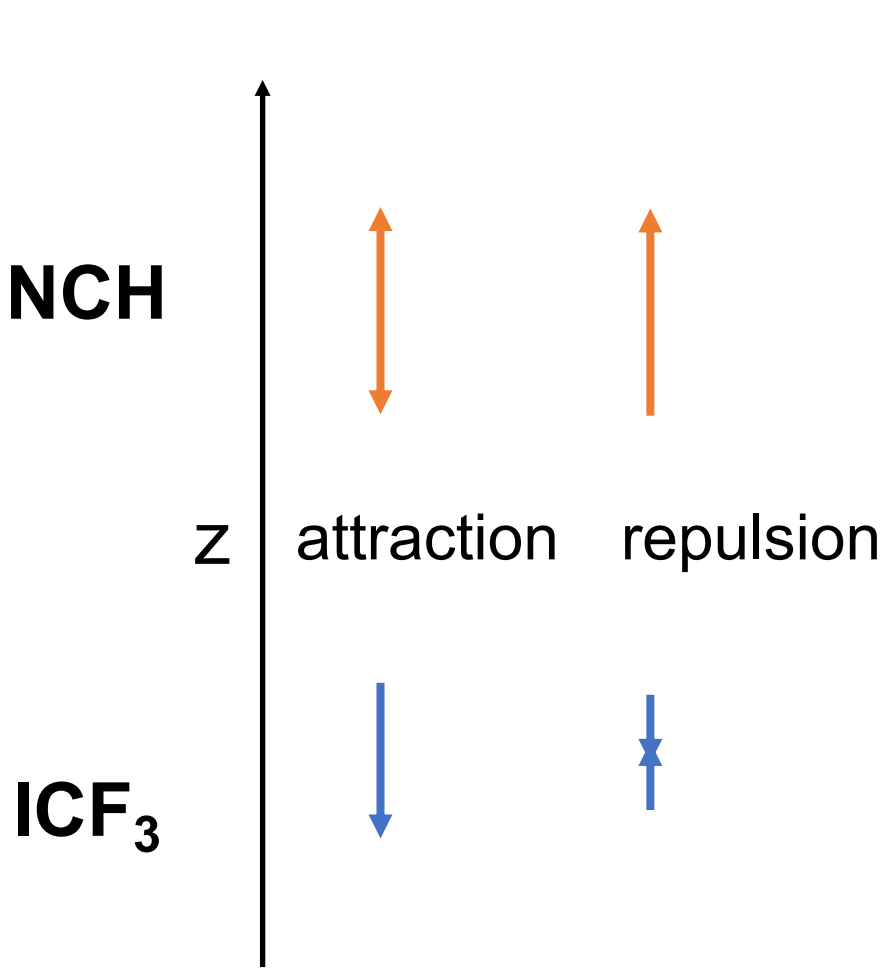
Dipole-dipole interaction

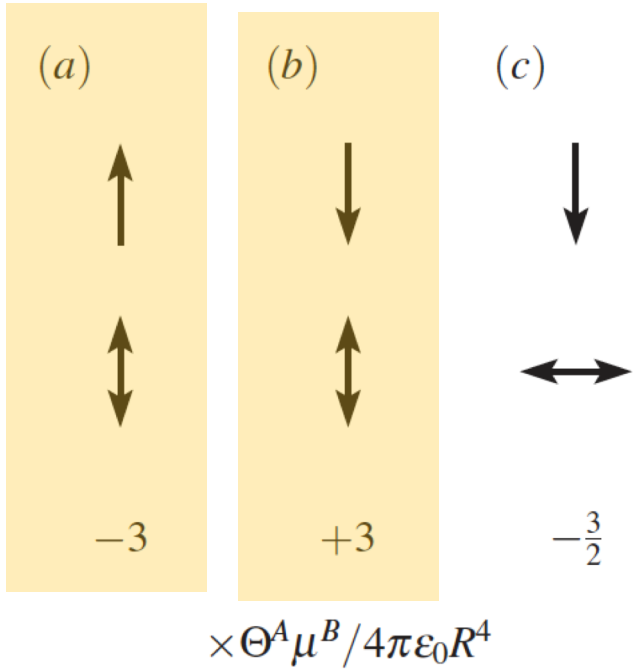




NOTE: if you take +2 here, the dipole you put into the equation should be the "magnitude" of the dipole. That is, they should be both positive ($\mu_A, \mu_B > 0$) .

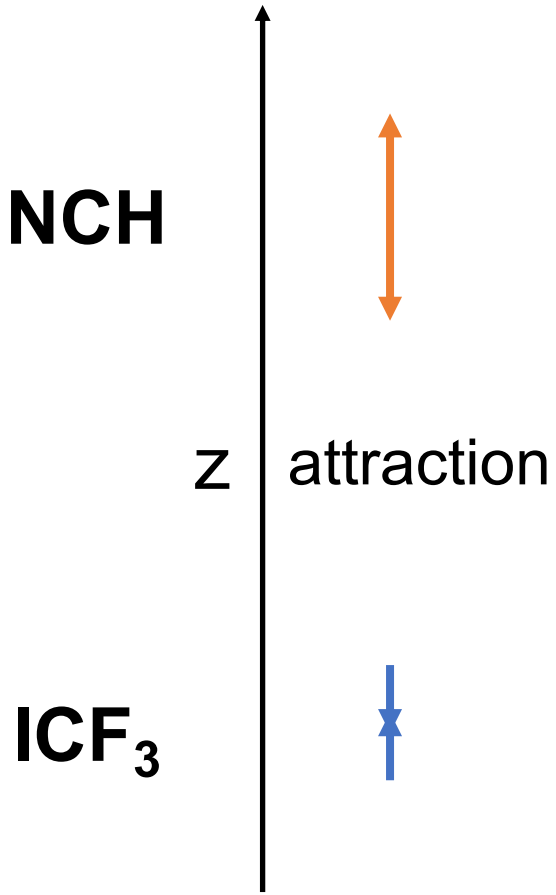
Dipole-quadrupole interaction

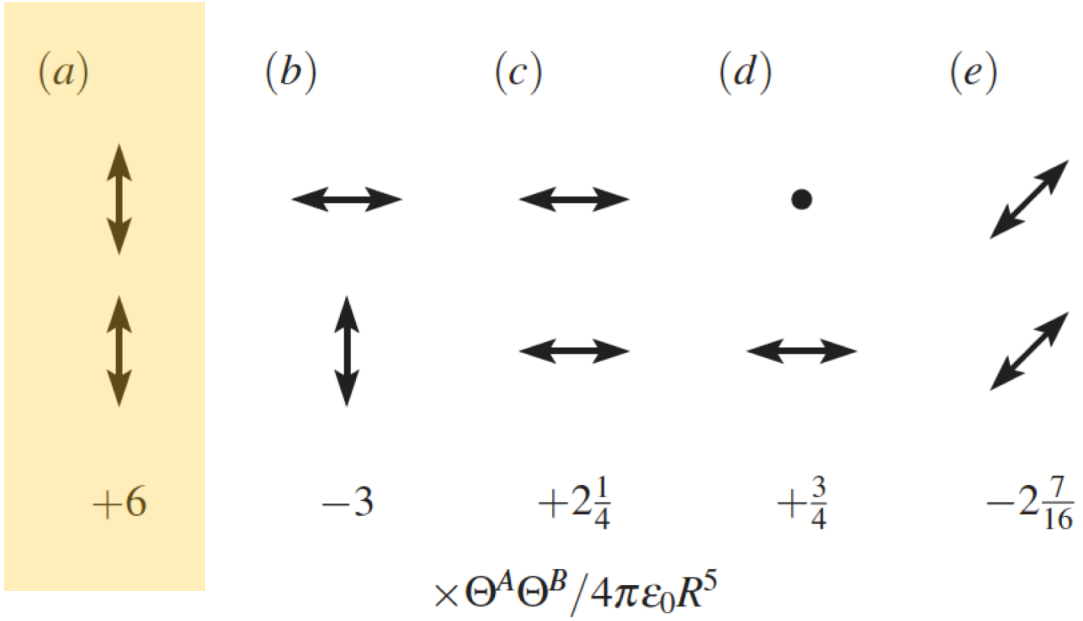




NOTE: The same note, you should put the "magnitude" of the dipole and quadrupole.

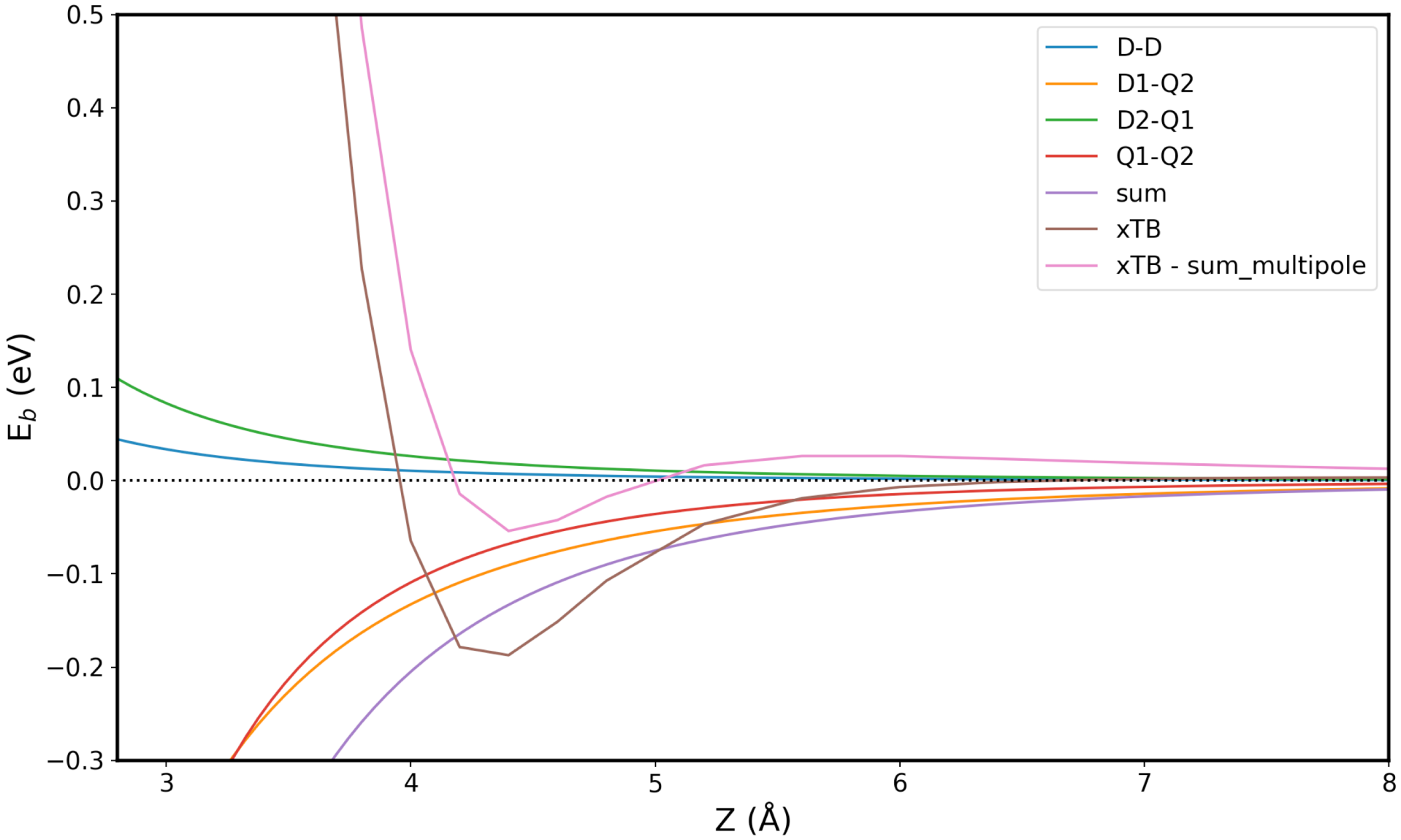
Quadrupole-quadrupole interaction



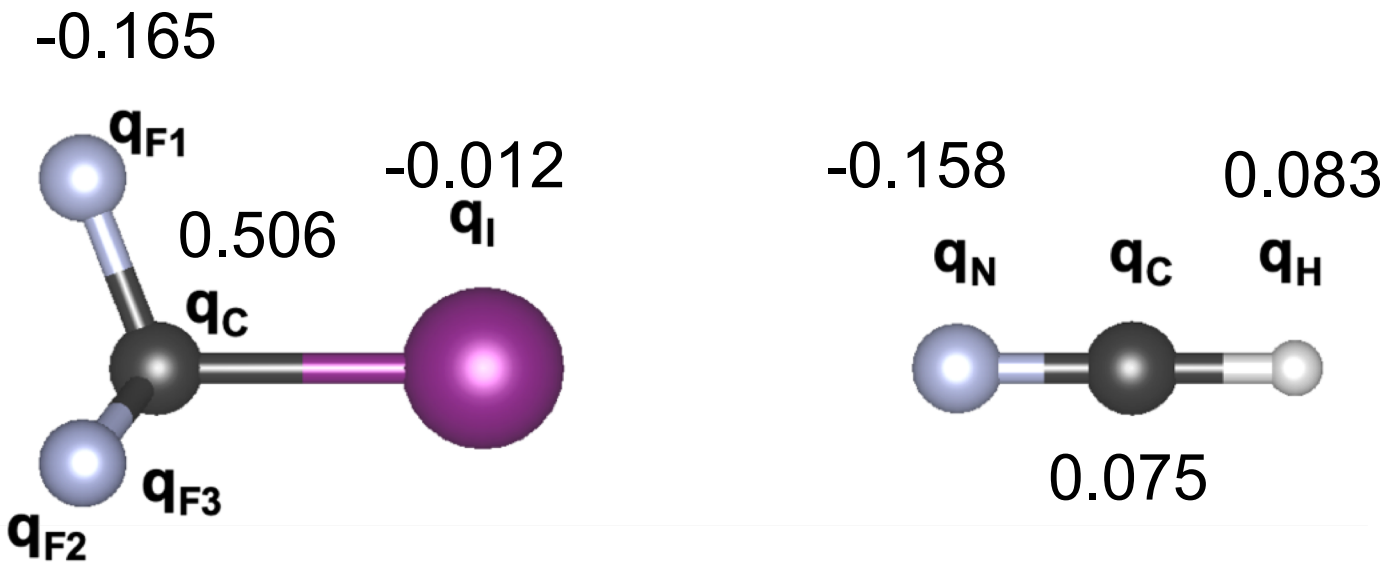


NOTE: You could put +6. In this case, you should include the "sign" in your quadrupoles.

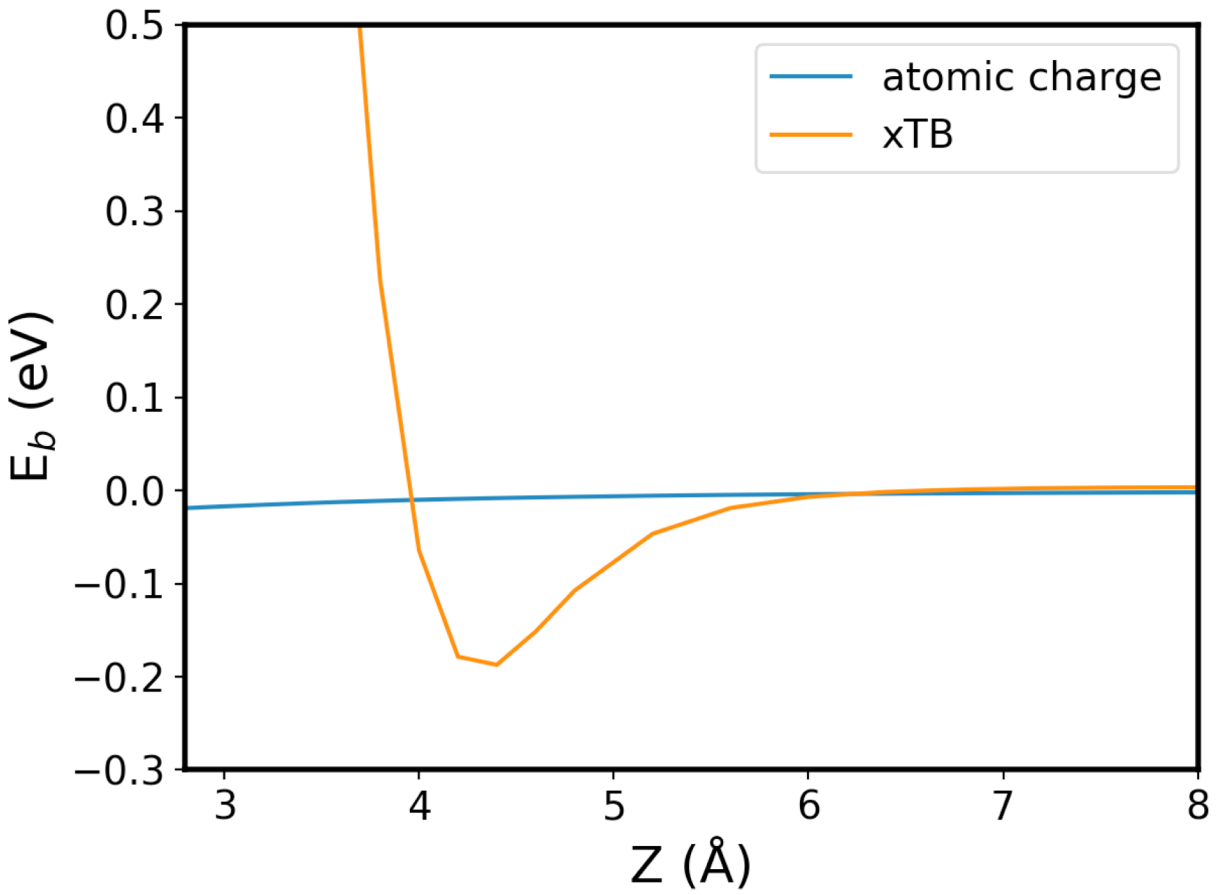
Comparison between xTB and multipole-multipole



Mulliken charge

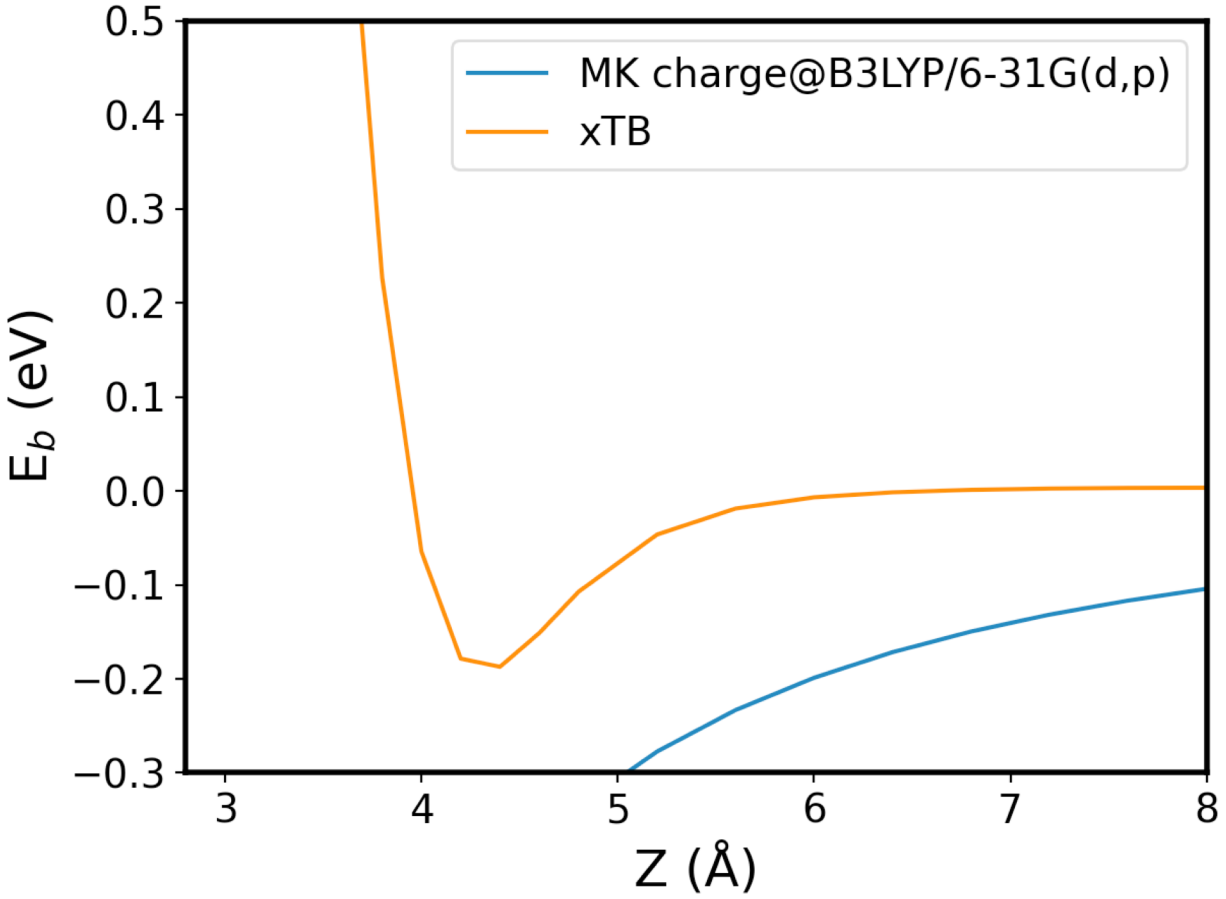


Comparison between xTB and Mulliken charge



- Doesn’t really work…
- Nearly not interacting based on atomic charge model.
- Mulliken charge is not good for this purpose.

Comparison between xTB and MK charge



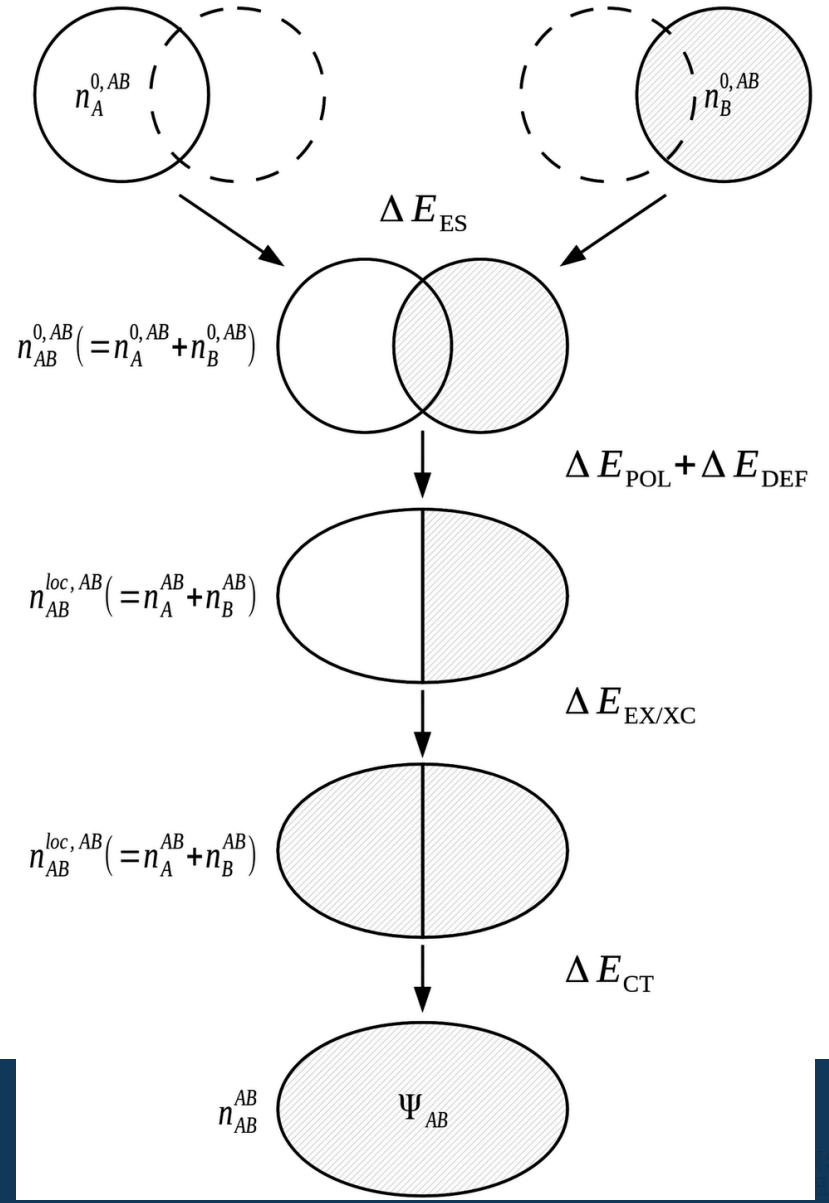
- Supposed to be more accurate than Mulliken charge.
- Common model used in forcefield MD simulations.
- Much stronger attraction.
- Goes to negative faster than multipole-multipole interaction.

Evaluate electrostatic contribution using QM method?

In general, these methods are called energy decomposition analysis (EDA).

$$\Delta E = \Delta E_{\mathrm{ES}} + \Delta E_{\mathrm{POL}} + \Delta E_{\mathrm{CT}} + \Delta E_{\mathrm{EX/XC}} + \Delta E_{\mathrm{DEF}}$$

Scheme of natural EDA.



Readings

- Energy decomposition analysis
  - Chem. Soc. Rev., 2015, 44, 3177-3211
- Molecular interactions
  - The Theory of Intermolecular Forces, 2nd Ed., Anthony Stone