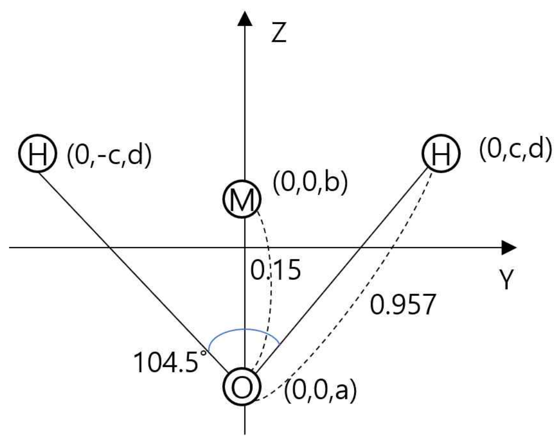
TIP4P, TIP5P Water Model

Consider the TIP4P (Rapaport 8.3) and the TIP5P (Lecture note - Force Fields) water models.

1. Make coordinate files of one water molecule for these models. They are text files of a format element charge x y z (Cartesian coordinates; one line per atom)

TIP4P



Put the center of mass as the origin, for z-coordinates $aM_O + 2dM_H = 0$,where $M_O = 16M_H$

we get $d = -8a$

for $r_{OH} = 0.957\,\text{Å}$, $r_{OM} = 0.15\,\text{Å}$, $\angle HOH = 104.5^\circ$

and $\sigma = 3.154\,\text{Å}$

$$c = 0.957 \times \sin\left(\frac{104.5^\circ}{2}\right)/3.154 = 0.239914$$

$$d - a = 0.957 \times \cos\left(\frac{104.5^\circ}{2}\right)/3.154 = 0.185762 = -9a$$

$$a = -0.020640,\ d = 0.165121$$

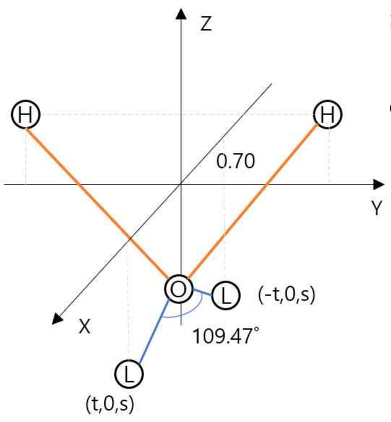
$$b = \frac{0.15}{3.154} + a = 0.026918$$

$$r_O = (0,\ 0,\ -0.0206)$$

$$r_M = (0,\ 0,\ 0.0269)$$

$$r_H = (0,\ \pm 0.240,\ 0.165)$$

TIP5P



Same way with TIP4P, we can easily get coordinates of H, O, changing $\sigma = 3.120\,\text{Å}$.

$r_O = (0,\ 0,\ -0.0209)$, $r_H = (0,\ \pm 0.243,\ 0.167)$

components L_1, L_2 would be on the x-z plane

$$t = 0.70 \times \sin\left(\frac{109.47^\circ}{2}\right)/3.120 = 0.183187$$

$$s = -0.7 \times \cos\left(\frac{109.47}{2}\right)/3.12 + r_{O,z} = -0.150401$$

$$r_L = (\pm 0.183,\ 0,\ -0.150)$$

Coordinates

```
   atom  charge  x     y       z
0     O    0.00  0  0.00 -0.0206
1     M   -1.04  0  0.00  0.0269
2    H1    0.52  0  0.24  0.1650
3    H2    0.52  0 -0.24  0.1650
--------------------------------------------------------------------------------
   atom  charge      x      y       z
0     O   0.000  0.000  0.000 -0.0209
1    L1  -0.241  0.183  0.000 -0.1500
2    L2  -0.241 -0.183  0.000 -0.1500
3    H1   0.241  0.000  0.243  0.1670
4    H2   0.241  0.000 -0.243  0.1670
```

2. Make a program code to calculate dipoles and quadrupoles from the coordinate files you made.

Method

Call coordinates of water model and calculate dipole and quadrupole into Debye unit

Python Code

- Step 1 : Coordinates

import *.txt files of coordinates of TIP4P, TIP5P change their data type to array to calculate

```python
import numpy as np
import pandas as pd
import math

"""call coordinate files of water model as dataframe"""
tip4p_df = pd.read_csv('TIP4P.txt',delimiter='  ', names=['atom','charge','x','y','z']) #TIP4P water model
tip5p_df = pd.read_csv('TIP5P.txt',delimiter='  ', names=['atom','charge','x','y','z']) #TIP5P water model

"""check coordinates"""
print(tip4p_df)
print('------------------------------------------------------------------------------------')
print(tip5p_df)

"""change type from dataframe to numpy"""
TIP4P=tip4p_df.to_numpy()
TIP5P=tip5p_df.to_numpy()
```

- Step 2 : Declare parameters

Declare parameters considering their unit (Å, C, Debye)

```python
"""parameters to calculate dipole and quadrupole"""
angstrom = 1 * 10**(-10)  # 1 Angstrom = 1E-10 m
charge = 1.60219 * 10**(-19)  # 1 e = 1.60219E-19 C
debye = 3.34*10**(-30) # 1 Debye = 3.34E-30 Cm
sigma1 = 3.154 # simga of TIP4P
sigma2 = 3.120 # sigma of TIP5P
```

- Step 3 : Preprocessing

Initialize lists and get r_n^2 of each atoms

```python
"""get r^2 to calculate quadrupole / TIP4P, TIP5P"""
for i in range (4):
    arr1 = np.array(TIP4P[i,2:])
    sq=np.dot(arr1, arr1)
    r_square_tip4p.append(sq)
for i in range (5):
    arr1 = np.array(TIP5P[i,2:])
    sq=np.dot(arr1, arr1)
    r_square_tip5p.append(sq)
```

- **Step 4 : Iteration for TIP4P**

  get $q_k \tilde{r}_k$ for dipole, $3\tilde{x}_{k,i}\tilde{x}_{k,j} - \tilde{r}_k^2 \delta_{ij}$ for quadrupole

```
#get dipole and quadrupole for TIP4P
for i in model_list:
    if i=='TIP4P':
        model = TIP4P

        ###get dipole###
        for j in range(4) :  #iteration for all atoms
            for k in range (2,5):  # iteration x,y,z for dipole
                qr=model[j,k] * model[j,1]  #sum of q * r
                qr_list.append(qr)

            ###get quadrupole###
            #calculate elements of quadrupole
            q= model[j,1]  #charge for each atom
            rn2_ = r_square_tip4p[j]  #r^2

            #get the terms to be sum [q*(3*x_i*x_j-r^2*delta]
            xx=(3*(model[j,2]**2) - rn2_) * q
            xy=3*(model[j,2]*model[j,3]) * q
            xz=3*(model[j,2]*model[j,4]) * q
            yy=(3*(model[j,3]**2) - rn2_) * q
            yz=3*(model[j,3]*model[j,4]) * q
            zz=(3*(model[j,4]**2) - rn2_) * q

            #make lists of quarupole for each coordinates
            q_xx.append(xx)
            q_xy.append(xy)
            q_xz.append(xz)
            q_yy.append(yy)
            q_yz.append(yz)
            q_zz.append(zz)
```

- **Step 5 : Calculate sum and change units. Print out. (TIP4P)**

```
# sum the elements for each coordinates
sum_q_xx = sum(q_xx)
sum_q_xy = sum(q_xy)
sum_q_xz = sum(q_xz)
sum_q_yy = sum(q_yy)
sum_q_yz = sum(q_yz)
sum_q_zz = sum(q_zz)

#bind them one place to calculate parameters
sum_quad_list = [sum_q_xx, sum_q_xy, sum_q_xz, sum_q_yy, sum_q_yz , sum_q_zz]
arr_quad=np.array(sum_quad_list) #chane from list to array

#Final step : calculate the parameters to get result in unit (Debye)
dipole = sum(qr_list) * sigma1 * angstrom * charge / debye  # Dipole : calculate and change dimension to Debye
quadrupole = arr_quad * (sigma1**2)* angstrom * charge / (2*debye ) # Quadrupole : calculate and change dimension to Debye
Qt = (quadrupole[3]-quadrupole[0])/2  # Quadrupole total

#print out Dipole and Quadrupole
print(i)
print("Dipole : %.2f D" % dipole)
print("Quadrupole_xx : %.2f DÅ" % quadrupole[0])
print("Quadrupole_yy : %.2f DÅ" % quadrupole[3])
print("Quadrupole_zz : %.2f DÅ" % quadrupole[5])
print("Quadrupole_T : %.2f DÅ" % Qt)
print()
```

- **Step 6 : Iteration for TIP5P**

```
#get dipole and quadrupole for TIP5P
    else :
        qr_list=[]
        q_xx=[]
        q_xy=[]
        q_xz=[]
        q_yy=[]
        q_yz=[]
        q_zz=[]

        model = TIP5P
        for j in range(5) :
            ###get dipole###
            for k in range (2,5):
                qr=model[j,k] * model[j,1]
                qr_list.append(qr)

            ###get quadrupole###
            # calculate elements of quadrupole
            q= model[j,1]
            rn2_ = r_square_tip5p[j]
            xx=(3*(model[j,2]**2) - rn2_) * q
            xy=3*(model[j,2]*model[j,3]) * q
            xz=3*(model[j,2]*model[j,4]) * q
            yy=(3*(model[j,3]**2) - rn2_) * q
            yz=3*(model[j,3]*model[j,4]) * q
            zz=(3*(model[j,4]**2) - rn2_) * q

            # make lists of quarupole for each coordinates
            q_xx.append(xx)
            q_xy.append(xy)
            q_xz.append(xz)
            q_yy.append(yy)
            q_yz.append(yz)
            q_zz.append(zz)
```

- **Step 7 : Calculate sum and change units. Print out. (TIP5P)**

```
# sum the elements for each coordinates
sum_q_xx = sum(q_xx)
sum_q_xy = sum(q_xy)
sum_q_xz = sum(q_xz)
sum_q_yy = sum(q_yy)
sum_q_yz = sum(q_yz)
sum_q_zz = sum(q_zz)
sum_quad_list = [sum_q_xx, sum_q_xy, sum_q_xz, sum_q_yy, sum_q_yz , sum_q_zz]

arr_quad=np.array(sum_quad_list)
dipole = sum(qr_list) * sigma1 * angstrom * charge / debye  # Dipole : calculate and change dimension to Debye
quadrupole = arr_quad * (sigma1**2)* angstrom * charge / (2*debye ) # Quadrupole : calculate and change dimension to Debye
Qt = (quadrupole[3]-quadrupole[0])/2  # Quadrupole total
# print out Dipole and Quadrupole
print(i)
print("Dipole : %.2f D" % dipole)
print("Quadrupole_xx : %.2f DÅ" % quadrupole[0])
print("Quadrupole_yy : %.2f DÅ" % quadrupole[3])
print("Quadrupole_zz : %.2f DÅ" % quadrupole[5])
print("Quadrupole_T : %.2f DÅ" % Qt)
print()
```

Result

	μ (Å)	$Q_{xx}(D\text{Å})$	$Q_{yy}(D\text{Å})$	$Q_{zz}(D\text{Å})$	$Q_T(D\text{Å})$
TIP4P	2.17	−2.09	2.20	−0.11	2.14
TIP5P	2.31	−1.51	1.68	−0.17	1.60