Week 6: Dihedrals and Temperature

2MMN40: Introduction to Molecular Modeling and Simulation
Last update: November 2, 2020

Contents

1 Some Project Advice

After you have implemented the things from this week your simulator is capable enough to do most of the tasks you need in your final project. Hence if you haven't started yet with your final project, this is the moment! You can also find all the parameters you need for you simulations in the final project description.

2 The Assignment

This weeks assignment consists of two parts:

1. Adding dihedral forces, these are forces due to torsion angles and are sometimes called "four-body" forces.
2. Adding a thermostat to control the temperature of the system.

3 Proper Dihedrals

Larger molecules with four or more atoms have additional degrees of freedom. To constrain them we need dihedral potentials.

Dihedrals do not necessarily have to be defined along bonds, often dihedrals are defined between disconnected atoms to make the molecule less “wobbly“, or keep them planar if the molecule has a flat part. These are called improper dihedrals and we will not need them for this course. We will be working with proper dihedrals, they constrain torsion angles. An example can be seen in the two figures below.

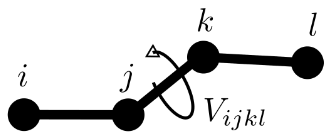

Figure 1. Indices used for dihedrals

The dihedral potential depends on the dihedral angle, care must be taken when defining the dihedral angle. If you take a look at the 3D image, then the angle between the two planes is called the dihedral angle, it is however not immediately apparent which of the two possible angles you should take.

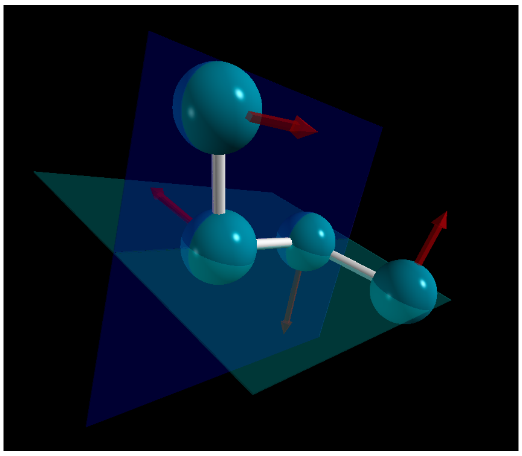

Figure 2. A 3D visualization of a dihedral and the forces resulting from it, made with VPython.

3.1 The Dihedral Potential

The potentials which are used for dihedrals can be simple harmonic potentials such as

$$V(\theta_{ijkl}) = 0.5k_0(\theta_{ijkl} - \theta_0)^2, \tag{1}$$

where θ_{ijkl} is the dihedral angle between the planes spanned by i, j, k and j, k, l. Often the potentials are taken to be periodic e.g:

$$V(\psi) = \frac{1}{2}\left(C_1(1+\cos(\psi)) + C_2(1-\cos(2\psi)) + C_3(1+\cos(3\psi)) + C_4(1-\cos(4\psi))\right) \tag{2}$$

where $\psi = \theta_{ijkl} - 180^o$ and C_1 to C_4 are forcefield constants (this is the potential you need for the final project). This looks more complicated, but realize that taking the derivative of this potential and implementing it as a force is relatively easy.

Derive an equation for the dihedral angle from the positions of i, j, k and l. From the potential given in equation 2, derive the expressions for the forces. Modify your topology and then the rest of your code to include dihedral interactions.

3.2 Getting the directions right

For the bonds and angles getting the direction vectors was relatively easy. For dihedrals it is a bit more complicated because you need to know a trick. The trick is that you only explicitly calculate the forces and directions for atoms i and l (this is more or less the same as the angle potential, why?).

The forces for atoms j and k can be obtained from Newton's third law. We have already seen the sum rule,

$$\sum_{x\in\{i,j,k,l\}} \mathbf{F}_x = 0, \tag{3}$$

but the same applies to torques

$$\sum_{x\in\{i,j,k,l\}} \mathbf{r}_{ox} \times \mathbf{F}_x = 0, \tag{4}$$

where o is the center of the bond jk. Now you have two equations and two unknowns and hence they are no longer unknown. Torques are used to describe rotations in physics. What the last equation says is that the torsion angle potential will not lead to rotation of the molecule. This is important debugging information, if your molecules start spinning for no reason in the simulation, there is probably something wrong with the sum of the torques not being equal to zero.

3.3 Visualize the result

After implementing the dihedral, simulate a single ethanol molecule (from the final project) and see if it behaves properly.

4 Temperature and thermostats

So far we have looked at a closed system, which started with some energy and conserved that energy over the simulation, this is known as the NVE ensemble (number of particles, system volume and total energy remain fixed). These systems are, however, not practical from an experimental point of view. In the lab it is easier to keep the temperature fixed. Hence we need to simulate an NVT ensemble (constant number of particles, volume and temperature), if we want to do predictions about an experiment, for example.

The temperature T of a system of particles is given by:

$$E_{\text{kin}} = \frac{1}{2} N_f k_B T, \tag{5}$$

where k_B is the Boltzmann constant, E_{kin} is the kinetic energy and N_f are the degrees of freedom of your system. How can you calculate the kinetic energy from the particle trajectories? How do the degrees of freedom relate to the number of particles N?

Implement a thermometer in your MD code (i.e. a function that computes the temperature). Print the temperature every timestep, what do you notice?

The easiest way of implementing a thermostat (i.e. temperature controller) is velocity rescaling. After every time step, you rescale all the velocities to keep the temperature constant (in physical terms you are adding or subtracting a bit of energy from the system). If you want to achieve a temperature of T_0 what is the scaling factor? Implement this thermostat and output the temperature and energies (potential, kinetic and total) of you system. What do you notice now?

5 (Optional) Visualizing 3D Vectors

Sometimes it is very useful to visualize 3D data to check if all your forces point in the right direction. An easy tool to make a 3D drawing (and animation), such as the one in this document, is VPython. It is easy to install either with `pip3` or `conda`.

If you want to try it out here is part of the code that generates the figure from this document.

```
#!/usr/bin/env python3

"""
DEPENDS ON: vpython
Install (normal): pip3 install vpython
Install (anaconda): conda install -c vpython vpython
Website: https://www.glowscript.org/docs/VPythonDocs/index.html
"""
from vpython import *
scene = canvas(width=1200, height=800)
atom_radius = 0.3
scene.center = vector(0,0,0)
axes = [vector(1,0,0), vector(0,1,0), vector(0,0,1)]

scene.caption= """
To rotate "camera", drag with right button or Ctrl-drag.
To zoom, drag with middle button or Alt/Option depressed, or use scroll wheel.
  On a two-button mouse, middle is left + right.
To pan left/right and up/down, Shift-drag.
"""

atoms = [vector(-0.9, 0.951, 0.309),
                vector(-0.6, 0, 0),
                vector(0.6, 0, 0),
                vector(0.9, -0.454, 0.891)]

for at in atoms:
  atom = sphere()
  atom.pos = at
  atom.radius = atom_radius
  atom.color = vector(0,0.58,0.69)

for i in range(3):
  curve(atoms[i], atoms[i+1],  radius=0.05)

rt = shapes.rectangle(width=3, height=2.6)
obj = extrusion(path=[vec(0,0,0), vec(0,0,-0.01)],
          shape=rt, opacity=0.2, color=color.blue)
obj.rotate(angle=0.1*pi, axis=vector(1,0,0))
lengthScaler = 0.8
dir1 = -(atoms[0]-atoms[1]).cross(atoms[2]-atoms[1])
pointer = arrow(pos=atoms[0], axis=dir1, shaftwidth=0.05, color=color.red,opacity=0.5)
```