MD simulation of a single water molecule

Specifications

Geometry of the water molecule

We use the specification for the flexible TIP3P-FB water model (Wang, Martinez, Pande, J. Phys. Chem. Lett., 5:1885, 2014), which is used in the Amber MD force field. Distances are here given in nanometers.

In[1]:=
```
distanceOH = Quantity[0.1011811, "nm"]
```
Out[1]=
```
0.101181 nm
```

In[2]:=
```
distanceHH = Quantity[0.1638684, "nm"]
```
Out[2]=
```
0.163868 nm
```

The water molecule can be viewed as an isosceles triangle, with oxygen at the top and the two H atoms at the base. Determine first the distance of the oxygen atom from the base:

In[3]:=
```
dOH = QuantityMagnitude[distanceOH]
```
Out[3]=
```
0.101181
```

In[4]:=
```
dHH = QuantityMagnitude[distanceHH]
```
Out[4]=
```
0.163868
```

In[5]:=
```
eq = (dHH / 2) ^ 2 + h ^ 2 == dOH ^ 2
```
Out[5]=
```
0.00671321 + h^2 == 0.0102376
```

In[6]:=
```
soln = Solve[eq, h]
```
Out[6]=
```
{{h → -0.0593667}, {h → 0.0593667}}
```

In[7]:=
```
hsol = soln[[2, 1, 2]]
```
Out[7]=
```
0.0593667
```

Define now the positions of the atoms:

In[8]:=
```
ROxygen = {0., 0., 0.}
```
Out[8]=
```
{0., 0., 0.}
```

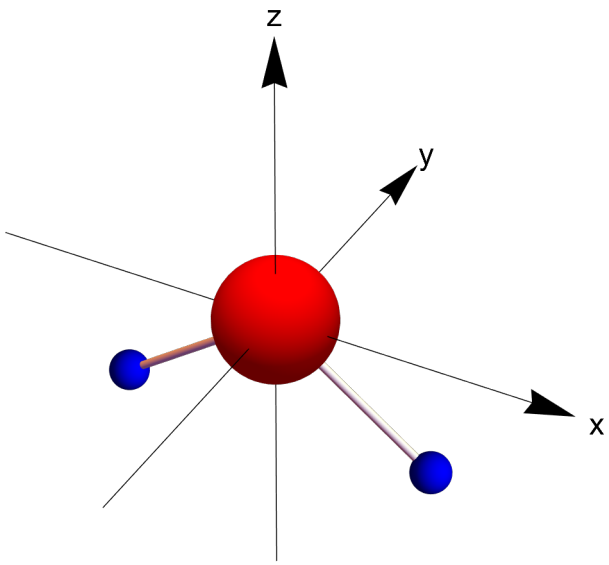
In[9]:=
```
RH1 = {dHH / 2, 0, -hsol}
```
Out[9]=
```
{0.0819342, 0, -0.0593667}
```

In[10]:=
```
RH2 = {-dHH / 2, 0, -hsol}
```

Out[10]=
```
{-0.0819342, 0, -0.0593667}
```

In[11]:=
```
Graphics3D[{{Red, Sphere[ROxygen, 0.03]}, {Blue, Sphere[RH1, 0.01]},
  {Blue, Sphere[RH2, 0.01]}, Tube[{ROxygen, RH1}, 0.002],
  Tube[{ROxygen, RH2}, 0.002], Arrow[{{-0.15, 0, 0}, {0.15, 0, 0}}],
  Arrow[{{0, -0.15, 0}, {0, 0.15, 0}}], Arrow[{{0, 0, -0.15}, {0, 0, 0.15}}],
  Text[Style["x", 16], {0.16, 0, 0}], Text[Style["y", 16], {0, 0.16, 0}],
  Text[Style["z", 16], {0, 0, 0.16}]}, Boxed → False]
```

Out[11]=



Atomic masses

Here a.m.u. is the atomic mass unit

In[12]:=
```
massH = Quantity[1, "amu"]
```

Out[12]=
```
1 u
```

In[13]:=
```
massO = Quantity[16, "amu"]
```

Out[13]=
```
16 u
```

In[14]:=
```
UnitConvert[massH, "kg"]
```

Out[14]=
```
1.66053907 × 10^-27 kg
```

In[15]:=
```
UnitConvert[massO, "kg"]
```

Out[15]=
```
2.65686251 × 10^-26 kg
```

Potential

This is the force constant of flexible TIP3P water potential of the Amber MD program

In[16]:=
```
Na = Quantity[1, "AvogadroNumber"]
```

Out[16]= 1 Avogadro number

In[17]:=
```
Ktip3pAmber = Quantity[553.0, "kcal/Angstroms^2"] / Na
```

Out[17]= $553.\ \text{kcal}_{\text{th}}/(\text{Avogadro number}\ \text{Å}^2)$

This is the force constant in a.m.u/ps^2

In[18]:=
```
Ktip3p = UnitConvert[Ktip3pAmber, "amu/picoseconds^2"]
```

Out[18]= $231\,375.\ \text{u}/\text{ps}^2$

The same force constant for O-H, and H-H

In[19]:=
```
K12 = QuantityMagnitude[Ktip3p]
```

Out[19]= 231 375.

In[20]:=
```
K13 = QuantityMagnitude[Ktip3p]
```

Out[20]= 231 375.

In[21]:=
```
K23 = QuantityMagnitude[Ktip3p]
```

Out[21]= 231 375.

Combined position vector

In[22]:=
```
r = {x1, y1, z1, x2, y2, z2, x3, y3, z3}
```

Out[22]= {x1, y1, z1, x2, y2, z2, x3, y3, z3}

Subvectors for O, H1, H2

In[23]:=
```
rOxygen = r[[1 ;; 3]]
```

Out[23]= {x1, y1, z1}

In[24]:=
```
rH1 = r[[4 ;; 6]]
```

Out[24]= {x2, y2, z2}

In[25]:=
```
rH2 = r[[7 ;; 9]]
```

Out[25]= {x3, y3, z3}

Distances

In[26]:=
```
dOH
```

Out[26]= 0.101181

In[27]:=
```
dHH
```

Out[27]= 0.163868

Functional form of the potential

In[28]:=
```
U = K12 (Sqrt[(rOxygen - rH1).(rOxygen - rH1)] - dOH)^2 +
  K13 (Sqrt[(rOxygen - rH2).(rOxygen - rH2)] - dOH)^2 +
  K23 (Sqrt[(rH2 - rH1).(rH2 - rH1)] - dHH)^2
```

Out[28]=
$$231\,375.\left(-0.101181+\sqrt{(x1-x2)^2+(y1-y2)^2+(z1-z2)^2}\right)^2+231\,375.\left(-0.101181+\sqrt{(x1-x3)^2+(y1-y3)^2+(z1-z3)^2}\right)^2+231\,375.\left(-0.163868+\sqrt{(-x2+x3)^2+(-y2+y3)^2+(-z2+z3)^2}\right)^2$$

The potential for the equilibrium position vanishes

In[29]:=
```
U /. {x1 → ROxygen[[1]], y1 → ROxygen[[2]], z1 → ROxygen[[3]], x2 → RH1[[1]],
  y2 → RH1[[2]], z2 → RH1[[3]], x3 → RH2[[1]], y3 → RH2[[2]], z3 → RH2[[3]]}
```

Out[29]=
```
0.
```

Forces

In[30]:=
```
MatrixForm[F = -Grad[U, r]]
```

Out[30]//MatrixForm=

$$\begin{pmatrix}
-\frac{462\,750.(x1-x2)\left(-0.101181+\sqrt{(x1-x2)^2+(y1-y2)^2+(z1-z2)^2}\right)}{\sqrt{(x1-x2)^2+(y1-y2)^2+(z1-z2)^2}}-\frac{462\,750.(x1-x3)\left(-0.101181+\sqrt{(x1-x3)^2+(y1-y3)^2+(z1-}\right.}{\sqrt{(x1-x3)^2+(y1-y3)^2+(z1-z3)^2}} \\
-\frac{462\,750.(y1-y2)\left(-0.101181+\sqrt{(x1-x2)^2+(y1-y2)^2+(z1-z2)^2}\right)}{\sqrt{(x1-x2)^2+(y1-y2)^2+(z1-z2)^2}}-\frac{462\,750.(y1-y3)\left(-0.101181+\sqrt{(x1-x3)^2+(y1-y3)^2+(z1-}\right.}{\sqrt{(x1-x3)^2+(y1-y3)^2+(z1-z3)^2}} \\
-\frac{462\,750.\left(-0.101181+\sqrt{(x1-x2)^2+(y1-y2)^2+(z1-z2)^2}\right)(z1-z2)}{\sqrt{(x1-x2)^2+(y1-y2)^2+(z1-z2)^2}}-\frac{462\,750.\left(-0.101181+\sqrt{(x1-x3)^2+(y1-y3)^2+(z1-z3)^2}\right)(z}{\sqrt{(x1-x3)^2+(y1-y3)^2+(z1-z3)^2}} \\
\frac{462\,750.(x1-x2)\left(-0.101181+\sqrt{(x1-x2)^2+(y1-y2)^2+(z1-z2)^2}\right)}{\sqrt{(x1-x2)^2+(y1-y2)^2+(z1-z2)^2}}+\frac{462\,750.(-x2+x3)\left(-0.163868+\sqrt{(-x2+x3)^2+(-y2+y3)^2+(-z2}\right.}{\sqrt{(-x2+x3)^2+(-y2+y3)^2+(-z2+z3)^2}} \\
\frac{462\,750.(y1-y2)\left(-0.101181+\sqrt{(x1-x2)^2+(y1-y2)^2+(z1-z2)^2}\right)}{\sqrt{(x1-x2)^2+(y1-y2)^2+(z1-z2)^2}}+\frac{462\,750.(-y2+y3)\left(-0.163868+\sqrt{(-x2+x3)^2+(-y2+y3)^2+(-z2}\right.}{\sqrt{(-x2+x3)^2+(-y2+y3)^2+(-z2+z3)^2}} \\
\frac{462\,750.\left(-0.101181+\sqrt{(x1-x2)^2+(y1-y2)^2+(z1-z2)^2}\right)(z1-z2)}{\sqrt{(x1-x2)^2+(y1-y2)^2+(z1-z2)^2}}+\frac{462\,750.(-z2+z3)\left(-0.163868+\sqrt{(-x2+x3)^2+(-y2+y3)^2+(-z2}\right.}{\sqrt{(-x2+x3)^2+(-y2+y3)^2+(-z2+z3)^2}} \\
\frac{462\,750.(x1-x3)\left(-0.101181+\sqrt{(x1-x3)^2+(y1-y3)^2+(z1-z3)^2}\right)}{\sqrt{(x1-x3)^2+(y1-y3)^2+(z1-z3)^2}}-\frac{462\,750.(-x2+x3)\left(-0.163868+\sqrt{(-x2+x3)^2+(-y2+y3)^2+(-z2}\right.}{\sqrt{(-x2+x3)^2+(-y2+y3)^2+(-z2+z3)^2}} \\
\frac{462\,750.(y1-y3)\left(-0.101181+\sqrt{(x1-x3)^2+(y1-y3)^2+(z1-z3)^2}\right)}{\sqrt{(x1-x3)^2+(y1-y3)^2+(z1-z3)^2}}-\frac{462\,750.(-y2+y3)\left(-0.163868+\sqrt{(-x2+x3)^2+(-y2+y3)^2+(-z2}\right.}{\sqrt{(-x2+x3)^2+(-y2+y3)^2+(-z2+z3)^2}} \\
\frac{462\,750.\left(-0.101181+\sqrt{(x1-x3)^2+(y1-y3)^2+(z1-z3)^2}\right)(z1-z3)}{\sqrt{(x1-x3)^2+(y1-y3)^2+(z1-z3)^2}}-\frac{462\,750.(-z2+z3)\left(-0.163868+\sqrt{(-x2+x3)^2+(-y2+y3)^2+(-z2}\right.}{\sqrt{(-x2+x3)^2+(-y2+y3)^2+(-z2+z3)^2}}
\end{pmatrix}$$

The sum of all forces is zero and there is no global translational acceleration

In[31]:=
```
FOxy = F[[1 ;; 3]] // Simplify;
```

In[32]:=
```
FH1 = F[[4 ;; 6]] // Simplify;
```

In[33]:=
```
FH2 = F[[7 ;; 9]] // Simplify;
```

In[34]:=
```
FOxy + FH1 + FH2
```

Out[34]=
```
{0., 0., 0.}
```

The forces on the atoms are zero in the equilibrium configuration

In[35]:=
```
F0 = F /. {x1 → ROxygen[[1]], y1 → ROxygen[[2]], z1 → ROxygen[[3]], x2 → RH1[[1]],
    y2 → RH1[[2]], z2 → RH1[[3]], x3 → RH2[[1]], y3 → RH2[[2]], z3 → RH2[[3]]}
```
Out[35]=
```
{0., 0., 0., 0., 0., 0., 0., 0., 0.}
```

Initialization of positions and velocities

Positions

In[36]:=
```
R0 = Flatten[{ROxygen, RH1, RH2}]
```
Out[36]=
```
{0., 0., 0., 0.0819342, 0, -0.0593667, -0.0819342, 0, -0.0593667}
```

Velocities

In[37]:=
```
kT = Quantity[1, "BoltzmannConstant"] * Quantity[293, "Kelvins"]
```
Out[37]=
```
293 K k
```

Thermal velocity of oxygen $v_{\text{th},O} = \sqrt{\frac{kT}{M_O}}$

In[38]:=
```
UnitConvert[Sqrt[kT / massO], "nanometers/picoseconds"]
```
Out[38]=
```
0.390203276 nm/ps
```

In[39]:=
```
vthOxygen =
 QuantityMagnitude[UnitConvert[Sqrt[kT / massO], "nanometers/picoseconds"]]
```
Out[39]=
```
0.390203276
```

Thermal velocity of hydrogen $v_{\text{th},H} = \sqrt{\frac{kT}{M_H}}$

In[40]:=
```
UnitConvert[Sqrt[kT / massH], "nanometers/picoseconds"]
```
Out[40]=
```
1.560813105 nm/ps
```

In[41]:=
```
vthH1 = QuantityMagnitude[UnitConvert[Sqrt[kT / massH], "nanometers/picoseconds"]]
```
Out[41]=
```
1.560813105
```

In[42]:=
```
vthH2 = QuantityMagnitude[UnitConvert[Sqrt[kT / massH], "nanometers/picoseconds"]]
```
Out[42]=
```
1.560813105
```

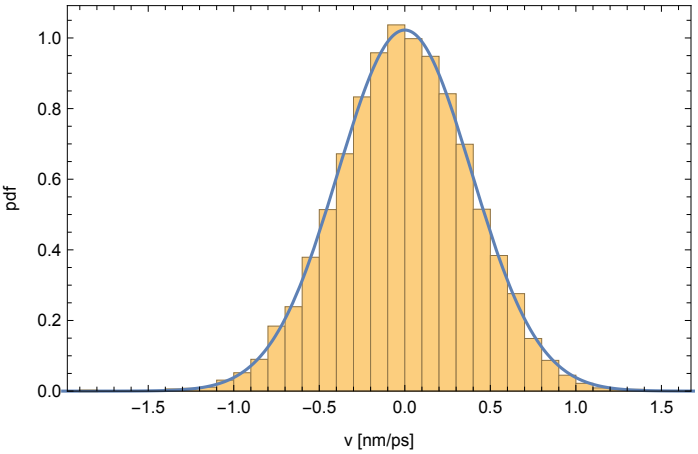
Chose velocities drawn from a Maxwell distribution

In[43]:=
```
V0oxy = RandomVariate[NormalDistribution[0, vthOxygen], 3]
```
Out[43]=
```
{-0.00529634, 0.125866, -0.0146292}
```

In[44]:=
```
vsample = RandomVariate[NormalDistribution[0, vthOxygen], 10 000];
```

In[45]:=
```
Show[Histogram[vsample, 30, PDF, Frame → True, FrameLabel → {"v [nm/ps]", "pdf"}],
 Plot[PDF[NormalDistribution[0, vthOxygen], v], {v, -2, 2}]]
```

Out[45]=



In[46]:=
```
V0h1 = RandomVariate[NormalDistribution[0, vthH1], 3]
```

Out[46]=
```
{-0.15357, -0.552495, -0.509473}
```

In[47]:=
```
V0h2 = RandomVariate[NormalDistribution[0, vthH2], 3]
```

Out[47]=
```
{-0.148851, -0.397633, -3.23434}
```

Vector of all initial velocities. These velocities include still a global translational and

In[48]:=
```
V0 = Flatten[{V0oxy, V0h1, V0h2}]
```

Out[48]=
```
{-0.00529634, 0.125866, -0.0146292, -0.15357,
 -0.552495, -0.509473, -0.148851, -0.397633, -3.23434}
```

Mass matrix

In[49]:=
```
mH = QuantityMagnitude[massH]
```

Out[49]=
```
1
```

In[50]:=
```
mO = QuantityMagnitude[massO]
```

Out[50]=
```
16
```

Diagonal mass matrix for a convenient formulation of the MD algorithm

In[51]:=
```
MatrixForm[M = DiagonalMatrix[N[{mO, mO, mO, mH, mH, mH, mH, mH, mH}]]]
```

Out[51]//MatrixForm=

$$\begin{pmatrix} 16. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 16. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 16. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 1. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 1. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 1. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. \end{pmatrix}$$

Inverse mass matrix

In[52]:=
```
MatrixForm[Minv = Inverse[M]]
```

Out[52]//MatrixForm=

$$\begin{pmatrix} 0.0625 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0.0625 & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0.0625 & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 1. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 1. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 1. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. \end{pmatrix}$$

MD integrator using the velocity Verlet algorithm

Integration time step, trajectory length

Integration time step

In[53]:=
```
Δt = 0.000001;
```

Number of steps for the full trajectory

In[54]:=
```
Nt = 15 000;
```

Number of steps in the equilibration phase

In[55]:=
```
NtEq = 5000;
```

Gap between scaling steps for the velocities

In[56]:=
```
NgapScale = 50;
```

Initialize empty trajectories

In[57]:=
```
Rtraj = Table[Table[0, 9], Nt];
```

In[58]:=
```
Vtraj = Table[Table[0, 9], Nt];
```

MD loop

In[59]:=
```
Rn = R0
```

Out[59]=
```
{0., 0., 0., 0.0819342, 0, -0.0593667, -0.0819342, 0, -0.0593667}
```

In[60]:=
```
Vn = V0
```

Out[60]=
```
{-0.00529634, 0.125866, -0.0146292, -0.15357,
 -0.552495, -0.509473, -0.148851, -0.397633, -3.23434}
```

In[61]:=
```
Ekin0 = V0.M.V0 / 2
```

Out[61]=
```
5.74348
```

In[62]:=
```
Fn = F /. Table[r[[k]] → Rn[[k]], {k, 1, 9}]
```

Out[62]=
```
{0., 0., 0., 0., 0., 0., 0., 0., 0.}
```

In[63]:=
```
EkinTh = mO * 3 * vthOxygen^2 / 2 + mH * 3 vthH1^2 / 2 + mH * 3 vthH2^2 / 2
```

Out[63]=
```
10.96261897
```

In[64]:=
```
EkinTh = QuantityMagnitude[UnitConvert[9 * kT / 2, "amu*nm^2/ps^2"]]
```

Out[64]=
```
10.96261897
```

In[65]:=
```
Do[If[Mod[i, 1000] == 0, Print["Step: ", i]];
 RnPlus1 = Rn + Δt * Vn;
 FnPlus1 = F /. Table[r[[k]] → RnPlus1[[k]], {k, 1, 9}];
 VnPlus1 = Vn + Δt * Minv.(Fn + FnPlus1) / 2;
 EkinNPlus1 = VnPlus1.M.VnPlus1 / 2;
 If[i ≤ NtEq,
  If[Mod[i, NgapScale] == 0, α = Sqrt[EkinTh / EkinNPlus1], α = 1], α = 1];
 VnPlus1Scaled = α * VnPlus1;
 Rtraj[[i]] = RnPlus1;
 Vtraj[[i]] = VnPlus1Scaled;
 EkinNPlus1Scaled = VnPlus1Scaled.M.VnPlus1Scaled / 2;
 Rn = RnPlus1;
 Vn = VnPlus1Scaled;
 Fn = FnPlus1, {i, 1, Nt}]
```

```
Step: 1000
Step: 2000
Step: 3000
Step: 4000
Step: 5000
Step: 6000
Step: 7000
Step: 8000
Step: 9000
Step: 10 000
Step: 11 000
Step: 12 000
Step: 13 000
Step: 14 000
Step: 15 000
```

Kinetic, potential and total energy

In[66]:=
```
EkinTraj = Table[Vtraj[[i]].M.Vtraj[[i]] / 2, {i, 1, Nt}];
```

In[67]:=
```
EpotTraj = Table[U /. Table[r[[k]] → Rtraj[[i, k]], {k, 1, 9}], {i, 1, Nt}];
```

In[68]:=
```
EtotTraj = Table[Vtraj[[i]].M.Vtraj[[i]] / 2 + U /.
     Table[r[[k]] → Rtraj[[i, k]], {k, 1, 9}], {i, 1, Nt}];
```

In[69]:=
```
ListLinePlot[{EkinTraj, EpotTraj, EtotTraj},
 PlotRange → All, PlotLegends → {"Ekin", "Epot", "Etot"},
 AxesLabel → {"time step", "Energy [a.m.u. nm^2/ps^2 ]"}]
```

Out[69]=