June 2006

# Molecular Mechanics of Water Monomer

Carl W. David
*University of Connecticut*, Carl.David@uconn.edu

# Molecular Dynamics of Water Monomer[1]

C. W. David
Department of Chemistry
University of Connecticut
Storrs, Connecticut 06269-3060
Carl.David@uconn.edu
6/15/2006

## Abstract

Simulation of the classical molecular dynamics of a water molecule can be useful in explaining nomral modes of motion, Fourier Transforms, and fundamental frequencies of vibration, as illustrated herein.

## Introduction

When learning about molecular vibrations, students can become mystified about the relationship of normal modes to actual atomic coordinates and their time variations. They also need help in understanding why the potential energy function, usually expressed in simple terms vis-à-vis bond length extensions, bond angle distortions and dihedral angle deformations, needs to be re-constituted into forms appropriate to those normal modes. Finally, the static diagrams used in illustrating normal modes do not necessarily convey the compete information needed to aquire mastery of this particular subject.

It is possible to carry out a molecular dynamics simulation of a simple molecule, water in our case, which allows students to not only "see" each aspect of the motion (and of the simulation itself) but manipulate the parameters in their potential energy model to more fully appreciate molecular vibrations (and

[1] This manuscript was denied publication.

rotations and translations, *vide ante*). In earlier work the HCℓ molecule was used to illustrate how molecular dynamics could be carried out. Since the simulations presented here are done in a symbolic algebraic system, students remain close to the analytical formulations they are used to without becoming distracted by high level coding in Fortran or other (more efficient) programming language. Finally, an appreciation of the Fourier Transform which allows "picking out" the vibrational frequencies from the molecular dynamics run's output data can be enhanced with a hands on activity which transcends the use of them as applied herein, and extends the understanding to other spheres in chemistry where this transform is used.

## The Molecular Mechanics Model

We are concerned here with a single molecule of water (larger systems might soon exhaust the resources of table top computers). The simplest forcefield for the molecule $H_1$-O-$H_2$ is

$$V = \frac{k_{OH}}{2}\left((r_{OH_1} - r_e)^2 + (r_{OH_2} - r_e)^2\right) + \frac{k_\vartheta}{2}(\vartheta - \vartheta_e)^2 \qquad (1)$$

where each OH distance is distorted relative to the equilibrium value (~0.95 Å), and the $H_1$-O-$H_2$ bond angle, $\vartheta$, is distorted relative to its equilibrium value of about 104.5°. $H_1$ and $H_2$ are labels which allow us to distinguish the two protons when and if we choose to treat HDO or HTO or DTO, the three isotopomers with unsymmetrical masses, or $D_2O$ or $T_2O$.

The problem for programmers in dealing with the vibrational motions of this molecule is evaluating the force on each nucleus. These forces consists of

three components on each atom(nucleus), x-, y- and z- each, and corresponds vectorially to the equation $\vec{F}_i = -\nabla V$ where i =1 and 3, say, refer to protons, and i=2 refers to the oxygen. The choice is arbitrary. Since $\nabla = \hat{i}\frac{\partial}{\partial x} + \hat{j}\frac{\partial}{\partial y} + \hat{k}\frac{\partial}{\partial z}$, it is a purely mechanical task to obtain expressions for the nine partial derivatives required to obtain the three force vectors. We use Maple to do this, since we want those expressions for instantaneous evaluation when needed during a single step of the molecular dynamics simulation. Then, since $\vec{F}_i = m_i\vec{a}_i$, we can calculate the acceleration each nucleus experiences, given the force acting on it, and from there calculate the change in velocity (and the change therefore of position), i.e., integrate Newton's equations for all nine "coordinates" pertinant to the (water) molecule.

Once we have the simulation running (debugged), we can change the parameters of the potential energy model and predict the values of the frequencies which we expect to see in the IR (i.e., predict the normal modes' frequencies of vibration). Working in 9 coordinates, while needing only three to describe the vibrations means that we have six "unused" coordinates, three for translation and three for rotation, which we could activate by suitable choices of initial positions and velocities of the 3 atoms (9 initial conditions in position, 9 initial conditions in velocity). Whether translating as an entity (or not) and whether rotating (or not) the vibrations (when small) remain essentially the same.

Finally, to "understand" the normal modes, we can set the water molecule's initial nuclear coordinates such that one of the three normal modes is (approximately) dominant, and watch for the dominant frequency of vibration which emerges, thus assigning (at least initial displacements) a normal mode to its associated frequency of vibration. Tinkering with the force constants while

exercising a particular normal mode allows us to obtain the value of that force constant which generates a frequency which agrees best with the experimental one.

## Implementing The Molecular Mechanics Model in Maple

All Maple programs begin with some prologue material. In our case, we need to make the Fourier Transform available, hence "inttrans" in the following code fragment:

```
restart;
with(inttrans):
delta_r := 0.1e-8;
delta_theta := 0.1;
h := 1.0e-16;#(this is the time step in seconds)GOOD VALUE for m=10
kOH := 7.76e5;#dynes/cm, pg 218 Barrow
kalpha := 0.699e5*r_e^2;#dynes/rad
m := 10;# FFT power of 2
```

We choose to use 0.1 Angstrom as the amount of displacement an OH bond will be subject to and 0.1 degrees as the amount that the H-O-H angle will be distorted from its equilibrium position. Further, we pick a time step (in seconds) corresponding to 10 picoseconds.

Lastly, we choose the numerical values of the force constants we are going to use in modeling the motion of this molecule, as well as the power of two ($2^m$) setting for the number of Fourier Transform points that we are going to collect.

Next, we define a magnitude function:

```
mag := proc (RealPart,ImaginaryPart)
sqrt(RealPart^2+ImaginaryPart^2);
end proc;
printlevel := 0;
```

and set the printlevel to a non-debugging value of 0.

We will need the distance of separation between two nuclei, and define a function which will obtain that value:

```
dist := proc(r1,r2) local t;
t := sqrt((r1[1]-r2[1])^2+(r1[2]-r2[2])^2+(r1[3]-r2[3])^2);
return(t);
end proc;
```

We are ready to compute the forces as *functions*. We define one of the H atoms coordinates as $x_{11}$, $x_{12,}$ and $x_{13,}$ i.e., $\vec{r}_1 = x_{11}\hat{i} + x_{12}\hat{j} + x_{13}\hat{k}$, while we define the oxygen's coordinates simply by $O_1$, $O_2$, and $O_3$. This means that the difference vector $\vec{r}_{12} = (x_{11} - O_1)\hat{i} + (x_{12} - O_2)\hat{j} + (x_{13} - O_3)\hat{k} = \vec{r}_{OH_1}$ which we here designate as $\vec{t}_1$ is given by:

```
t1 := [x11,x12,x13]-[O1,O2,O3]:
t2 := [x21,x22,x23]-[O1,O2,O3]:
```

We need the magnitudes of these vectors to calculate the forces, so we use our previously defined mag function:

```
t1_mag := sqrt(t1[1]^2+t1[2]^2+t1[3]^2):
t2_mag := sqrt(t2[1]^2+t2[2]^2+t2[3]^2):
```

Lastly, we need to know the magnitude of the instantaneous value of the H-O-H bond angle which we obtain using one of the vector dot product formulations:

```
theta_mag := arccos((t1[1]*t2[1]+t1[2]*t2[2]+t1[3]*t2[3])/(t1_mag*t2_mag)):
```

i.e., $\vartheta = \cos^{-1}\left(\dfrac{\vec{r}_{OH_1} \bullet \vec{r}_{OH_2}}{\left|\vec{r}_{OH_1}\right|\left|\vec{r}_{OH_2}\right|}\right)$. Now, we have all the functions necessary to obtain a functional description of the potential energy function for an isolated water molecule, subject to the simplest model possible. We have for the simplest potential energy model (translated into Maple):

```
t3 := (kOH/2)*((t1_mag-r_e)^2+(t2_mag-r_e)^2)+(kalpha/2)*(theta_mag-
theta_e)^2:
```

We note that $r_e$ is the designated "equilibrium" bond length, the value of r for which the potential energy curve has a minimum. Now we can take the nine partial derivatives required to get the three components of force for each of the three nuclei:

```
d11 := diff(t3,x11):
d12 := diff(t3,x12):
d13 := diff(t3,x13):
d21 := diff(t3,x21):
d22 := diff(t3,x22):
d23 := diff(t3,x23):
dO1 := diff(t3,O1):
dO2 := diff(t3,O2):
dO3 := diff(t3,O3):
```

For future use, we define a potential energy function which will be useful later (the last line is "returned" by the subroutine as its "value"):

```
V := proc(r_H_1,r_H_2,r_O) local t1,t2,t1_mag,t2_mag,theta_mag;
t1 := r_H_1-r_O;
t2 := r_H_2-r_O;
t1_mag := sqrt(t1[1]^2+t1[2]^2+t1[3]^2);
t2_mag := sqrt(t2[1]^2+t2[2]^2+t2[3]^2);
theta_mag := arccos((t1[1]*t2[1]+t1[2]*t2[2]+t1[3]*t2[3])/(t1_mag*t2_mag)):
(kOH/2)*((t1_mag-r_e)^2+(t2_mag-r_e)^2)+(kalpha/2)*(theta_mag-
theta_e)^2:
end proc;
```

We need some water-specific constants defined:

```
r_e := 0.9584e-8;#cm
theta_e := evalf((104.5)*Pi/180);
theta_e_delta := evalf(delta_theta*Pi/180);

m_H_1 := 1.0078/(6.023e23):#grams/atom
m_H := m_H_1;#choose your isotope
```

```
m_O := 16/(6.023e23);#choose your isotope
```

(substituting 2.014102 amu for 1.0078 would allow us to treat $D_2O$, *vide infra*)

It is time to set up the normal mode indexing that will be used. Our indexing system defines "norm_mode" = 0 to mean *we* will set the initial coordinates, while "normal mode" =1, 2, or 3 means we are attempting the textbook standard displacements according to which normal mode we are interested in, as indicated in the code:

```
norm_mode := 1;
r_O := [0,0,0];#set this in common, but change later
if norm_mode = 0 then
  r_H_1 := [0.9e-8,0.5e-8,0]:
  r_H_2 := [-0.7e-8,0.6e-8,0]:
elif norm_mode = 2 then
#WAGGING
print(` theta varying normal mode`);
  x_H_1 := r_e*cos(theta_e/2+theta_e_delta);
  y_H_1 := r_e*sin(theta_e/2+theta_e_delta);
print (`x,y = `,x_H_1, y_H_1);
  x_H_2 :=  x_H_1;
  y_H_2 := - y_H_1;
  r_H_1 := [x_H_1,y_H_1,0];
  r_H_2 := [x_H_2,y_H_2,0];
print (`h-h dist = `,dist(r_H_1,r_H_2));
elif norm_mode = 1 then
#SYMMETRIC STRETCH
print(` symmetric stretch normal mode`);

  x_H_1 := (r_e+delta_r)*cos(theta_e/2);
  y_H_1 := (r_e+delta_r)*sin(theta_e/2);
  x_H_2 := x_H_1;
  y_H_2 := -y_H_1;
  r_H_1 := [x_H_1,y_H_1,0];
  r_H_2 := [x_H_2,y_H_2,0];
print (`h-h dist = `,dist(r_H_1,r_H_2));

elif norm_mode = 3 then
```

```
#ANTISYMMETRIC STRETCH
print(` ANTI symmetric stretch normal mode`);

  x_H_1 := (r_e+0.2e-8)*cos(theta_e/2);
  y_H_1 := (r_e+0.2e-8)*sin(theta_e/2);
  x_H_2 := x_H_1;
  y_H_2 := -y_H_1;
  r_H_1 := [x_H_1,y_H_1,0];
  r_H_2 := [x_H_2,y_H_2,0];
r_O := [0.1e-8,0,0];

print (`h-h dist = `,dist(r_H_1,r_H_2));

end if;
print (`starting H1 coords = `,r_H_1);
print (`starting H2 coords = `,r_H_2);
print (`starting O coords = `,r_O);
```

We have included some explanatory printing to help understand what is going on.

Arbitrarily, we assign all the nuclei zero velocity (initially) although experimentation here is certainly encouraged. We also zero the forces!

```
v_H_1 := [0,0,0]:
v_H_2 := [0,0,0]:
v_H_1_p := [0,0,0]:
v_H_2_p := [0,0,0]:
v_O := [0,0,0]:
v_O_p := [0,0,0]:

F_H_1 := [0,0,0]:
F_H_2 := [0,0,0]:
F_O := [0,0,0]:
F_H_1_p := [0,0,0]:
F_H_2_p := [0,0,0]:
F_O_p := [0,0,0]:
```

Next, we compute the potential energy at the start of the motion:

```
V_start :=V(r_H_1,r_H_2,r_O):
print (`starting potential energy = `,V_start);
```

Here, we use “m” defined at the outset, and set up some arrays which will be used to gather information:

```
n_stop := 2^m;
l := array(1..n_stop);
y := array(1..n_stop);
```

All the preparatory matters have been taken care of and we are ready to start simulating, one time step at a (computer) cycle. We need a control statement:

```
For I from 1 by 1 to n_stop do
```

And then, as the saying went, “away we go”.

Inside each time step, preparatory to calculating the instantaneous force on each nucleus, we substitute the now current values of all the coordinates into the appropriate partial derivative terms, and evaluate them numerically. The minus sign is used to connect the force to “minus” the gradient, as noted above. We have :

```
t5 := subs(x11=r_H_1[1],x12=r_H_1[2],x13=r_H_1[3],
x21=r_H_2[1],x22=r_H_2[2],x23=r_H_2[3],
O1=r_O[1],O2=r_O[2],O3=r_O[3],
d11):
F_H_1[1] := -evalf(t5):

t5 := subs(x11=r_H_1[1],x12=r_H_1[2],x13=r_H_1[3],
x21=r_H_2[1],x22=r_H_2[2],x23=r_H_2[3],
O1=r_O[1],O2=r_O[2],O3=r_O[3],
d12):
F_H_1[2] := -evalf(t5):
```

```
t5 := subs(x11=r_H_1[1],x12=r_H_1[2],x13=r_H_1[3],
x21=r_H_2[1],x22=r_H_2[2],x23=r_H_2[3],
O1=r_O[1],O2=r_O[2],O3=r_O[3],
d13):
F_H_1[3] := -evalf(t5):

t5 := subs(x11=r_H_1[1],x12=r_H_1[2],x13=r_H_1[3],
x21=r_H_2[1],x22=r_H_2[2],x23=r_H_2[3],
O1=r_O[1],O2=r_O[2],O3=r_O[3],
d21):
F_H_2[1] := -evalf(t5):

t5 := subs(x11=r_H_1[1],x12=r_H_1[2],x13=r_H_1[3],
x21=r_H_2[1],x22=r_H_2[2],x23=r_H_2[3],
O1=r_O[1],O2=r_O[2],O3=r_O[3],
d22):
F_H_2[2] := -evalf(t5):

t5 := subs(x11=r_H_1[1],x12=r_H_1[2],x13=r_H_1[3],
x21=r_H_2[1],x22=r_H_2[2],x23=r_H_2[3],
O1=r_O[1],O2=r_O[2],O3=r_O[3],
d23):
F_H_2[3] := -evalf(t5):

t5 := subs(x11=r_H_1[1],x12=r_H_1[2],x13=r_H_1[3],
x21=r_H_2[1],x22=r_H_2[2],x23=r_H_2[3],
O1=r_O[1],O2=r_O[2],O3=r_O[3],
dO1):
F_O[1] := -evalf(t5):
t5 := subs(x11=r_H_1[1],x12=r_H_1[2],x13=r_H_1[3],
x21=r_H_2[1],x22=r_H_2[2],x23=r_H_2[3],
O1=r_O[1],O2=r_O[2],O3=r_O[3],
dO2):
F_O[2] := -evalf(t5):

t5 := subs(x11=r_H_1[1],x12=r_H_1[2],x13=r_H_1[3],
x21=r_H_2[1],x22=r_H_2[2],x23=r_H_2[3],
O1=r_O[1],O2=r_O[2],O3=r_O[3],
dO3):
F_O[3] := -evalf(t5):
```

Having the instantaneous forces, we now can obtain the (primed) new positions of the nuclei as predicted from the old ones, the velocities of the nuclei and the forces on the nuclei, according to Verlet's algorithm.

```
r_H_1_p := r_H_1 + h*v_H_1 + ((h^2)/(2*m_H))*F_H_1;
r_H_2_p := r_H_2 + h*v_H_2 + ((h^2)/(2*m_H))*F_H_2;

r_O_p := r_O + h*v_O + ((h^2)/(2*m_O))*F_O;
```

(Were we to allow the two masses of the H nuclei to be different, e.g., allowing one to be about double of the other (for example), then we could treat HDO as well. This would require slight coding changes in the center line (above). Of course, we could add code for Tritium in the mass definitions, and treat all the isotopomers.)

Once we've "moved" the nuclei, we need to recalculate the forces, which is indicated below (some repetitive code has been removed).

```
t5 :=
subs(x11=r_H_1_p[1],x12=r_H_1_p[2],x13=r_H_1_p[3],x21=r_H_2_p[1],x2
2=r_H_2_p[2],x23=r_H_2_p[3],
O1=r_O_p[1],O2=r_O_p[2],O3=r_O_p[3],
d11):
F_H_1_p[1] := -evalf(t5):

t5 :=
subs(x11=r_H_1_p[1],x12=r_H_1_p[2],x13=r_H_1_p[3],x21=r_H_2_p[1],x2
2=r_H_2_p[2],x23=r_H_2_p[3],
O1=r_O_p[1],O2=r_O_p[2],O3=r_O_p[3],
d12):
F_H_1_p[2] := -evalf(t5):
#repetitive code omitted here
t5 :=
subs(x11=r_H_1_p[1],x12=r_H_1_p[2],x13=r_H_1_p[3],x21=r_H_2_p[1],x2
2=r_H_2_p[2],x23=r_H_2_p[3],
```

```
O1=r_O_p[1],O2=r_O_p[2],O3=r_O_p[3],
dO3):
F_O_p[3] := -evalf(t5):
```

```
We also need to recompute the velocities (primed) to their new values, based
on the new forces (primed)
v_H_1_p := v_H_1 + (h/(2*m_H))*(F_H_1_p + F_H_1):
v_H_2_p := v_H_2 + (h/(2*m_H))*(F_H_2_p + F_H_2):
v_O_p := v_O + (h/(2*m_O))*(F_O_p + F_O):

r_H_1 := r_H_1_p:#update coordinates
r_H_2 := r_H_2_p:
r_O := r_O_p:
```

(We are again forced to change the coding slightly if we are interested in treating water with two unequal hydrogen isotopes.) Next, we need to update the coordinates and velocities from the new values in this cycle to the "to be old values" for the next cycle, i.e.,

```
v_H_1 := v_H_1_p:#update velocities
v_H_2 := v_H_2_p:
v_O := v_O_p:
```

We've achieved what was required, one cycle of the simulation. Now we need to do some auxiliary computations, which will allow analysis of our results. First, we compute the magnitudes of the velocities of the three nuclei, precursors of the computation of the total kinetic energy of the molecule:

```
v_H_1_mag := sqrt(v_H_1[1]^2+v_H_1[2]^2+v_H_1[3]^2);#for kinetic
energy
v_H_2_mag := sqrt(v_H_2[1]^2+v_H_2[2]^2+v_H_2[3]^2);
v_O_mag := sqrt(v_O[1]^2+v_O[2]^2+v_O[3]^2);
```

Next, we zero the imaginary part of the Fourier transform output (not really necessary) and then compute the kinetic energy, the potential energy, and the total energy at this time step.We have:

```
 y[i] := 0;#imaginary part
```

```
 KE(i) :=
1/2*(m_H*v_H_1_mag^2+m_H*v_H_2_mag^2+m_O*v_O_mag^2):#
 PE(i) := V(r_H_1,r_H_2,r_O):
 E_total(i) := evalf(KE(i)+PE(i)):
 E_totaloverV[i] := (E_total(i)/V_start)*100;
 E_totaloverV_plot(i) := (E_total(i)/V_start)*100;
```

Lastly, but most importantly from the perspective of visualization, we compute the $H_1$-$H_2$ distance (instantaneously) for two purposes. First, we wish to plot this resultant value set as a function to "time" (hence r_plot) and second, we will carry out our Fourier transform on it also (hence l[i]),

```
 l[i] := dist(r_H_1,r_H_2)*1e8:#convert to angstrom
 r_plot(i) := l[i]:
 yy[i] := 0;#imaginary part
```

and then we're done, with a single time step. The loop proceeds n_stop times.

```
end do;
```

Next, we carry out the Fourier Transform on l[i],y[i]

```
i := 'i':
FFT(m,l,y);
```

and plot the internuclear (proton-proton) separation as a function of "time":

```
print(` r_plot(i):`);
PLOT(POINTS(seq([i,r_plot(i)],i=1..n_stop),
SYMBOL(DIAMOND),LEGEND(`r versus t`)));
```

Then, we plot the Fourier transform of that data:

```
i := 'i':
FreqSpectrum := [seq([(i-1),(2*mag(l[i],y[i])/(2^m))],i=1..floor(((2^m)/2)))]:
#plot([seq(FreqSpectrum[j],j=2..floor(((2^m)/2)))],title="Fourier Transform");
plot([seq(FreqSpectrum[j],j=2..40)],title="Fourier Transform");
```

Note that the commented out line (above) would plot the entire Fourier Transform, but lowering the upper limit of the sequence allows the reader to better see where the maximum is (or maxima are).

Next, we compute some reportable molecularly interesting results of these computations:

```
frequency := number_of_cycles/(h*n_stop);
print(`sec^-1 = `,frequency,` times # of peaks`);
print(`cm^-1 = `,evalf(frequency/3e10),` times # of peaks`);


print(` KE(i):`);
PLOT(POINTS(seq([i,KE(i)],i=1..n_stop),
SYMBOL(CROSS),LEGEND(`kinetic energy versus t`)));
print(` PE(i):`);

PLOT(POINTS(seq([i,PE(i)],i=1..n_stop),
SYMBOL(CIRCLE),LEGEND(`potential energy versus t`)));
print(` E_total(i):`);
l_plot := [[ n, E_totaloverV(n)] $n=1..n_stop]:
 plot(l_plot, n=1..n_stop, style=line,symbol=circle,labels=[`time
 step`,`% deviation of total energy`],labeldirections=[HORIZONTAL,
 VERTICAL]);
```

And we're done. The last bit of code is included for completeness, i.e., it's optional. However, it is really interesting to see that the kinetic and potential energies are $180^{o}$ of out phase, and that when added together give almost a perfectly constant total energy (one has to check the ordinate of the energy plot to see that the variations in total energy are incredibly small, but real). These last plots are not included in this manuscript.

## Using the Code (1)

Choosing force constants, ($7.76 \times 10^5$ dynes/cm for $k_{OH}$ and $0.699 \times 10^5 * (0.9584)^2$ dynes/rad) from older literature allows us to test the basic concept, i.e., run the code using the three normal mode choices, to check that we get suitable harmonic motion. For the bond angle stretch mode, we obtain nice, in fact beautiful, harmonic motion (see Figure 1).
The Fourier Transform of this motion is shown in Figure 2, where one sees that there are about 9 cycles during the total sample.

For the anti-symmetric mode on the other hand, we find, indeed, that the motion is anything but harmonic, which is quite suggestive of the need to improve the initial coordinate guesses that we made. Figure 3 shows this complex motion (as reflected in the $r_{HH}(t)$ data). Perhaps it might be wise to tinker with the initial velocities as well as the initial positions in this case.

## Using the Code (2)

There is no question that seeing the motion vividly displayed this way, accompanied by the Fourier Transform which informs us of the frequencies, is gratifying. But one can do more with the simulation.

First, one can obtain the best force constants (inside the model definition) consonant with the experimental data. Some experimental data is shown in Table 1.

We can adjust the time step (h) in the code, to get as close as possible to an integral number of cycles inside the $2^m$ datum, and then calculate the resultant frequency. Tinkering with the force constants would then allow getting the closest fit of simulation frequencies to experimental frequencies. Then, with just a slight alteration in the coding, one could, for instance, predict the spectrum of HDO. Consider that the frequency ($\upsilon_2$) is given primitively by the expression:

$$\nu = \frac{\#\_of\_cycles}{time\_for\_these\_cycles}$$

which translates to

$$\nu_2 = \frac{\approx 9.5}{h \times n\_stop} = \frac{9.5}{0.965 \times 2048} \times \frac{1}{3*10^{10}} = 1602.3 cm^{-1}$$

Tinkering with the two force constants and getting a closer approximation to an integral number of cycles/experiment (varying h) yields frequencies which can be made to come as close as one desires to the experimental value.

By changing the velocity components, one can see that they are, when not overly large, ignorable. This means that the separation of vibration from rotation and translation is warranted under those assumptions.

By changing the initial coordinates to reflect enormous distortions, one can see the motion deteriorate rapidly into something ostensibly chaotic.

## Using the Code (3)

Another application which might be of interest consists of altering the potential energy model employed. Clearly, one could add cross terms to the simplified function (see Equation 1). Completely different models might also be worthy of exploration. Finally, extending the code to include more than 3 nuclei, or changing the nature of the nuclei considered in the 3 nucleus case, is straightforward.

## Discussion

It is easy to hand wave arguments in elementary physical chemistry which lead students to a glib form of understanding of topics which, upon closer investigation, are really completely confusing to students. The example used here allows a form of concrete realization of vibrational energy concepts which elude understanding under normal teaching conditions. Its inclusion as an

exercise, especially for students who will not continue in physical chemistry, seem warranted. The hard thing is to understand the V=f(nine coordinates) and therefore that $\vec{F}_i$ therefore are also functions of nine coordinates, and is non-trivial to derive and/or evaluate.

# Tables

| | $H_2O$[6] | $D_2O$[5] | $H_2O$(alternate)[7] |
|---|---|---|---|
| $\nu_1$ | 3832.17 | 2763.80 | 3657 |
| $\nu_2$ | 1648.47 | 1206.39 | 1594.7 |
| $\nu_3$ | 3942.53 | 2888.78 | 3755.7 |

Table1: Experimental IR frequencies for water.

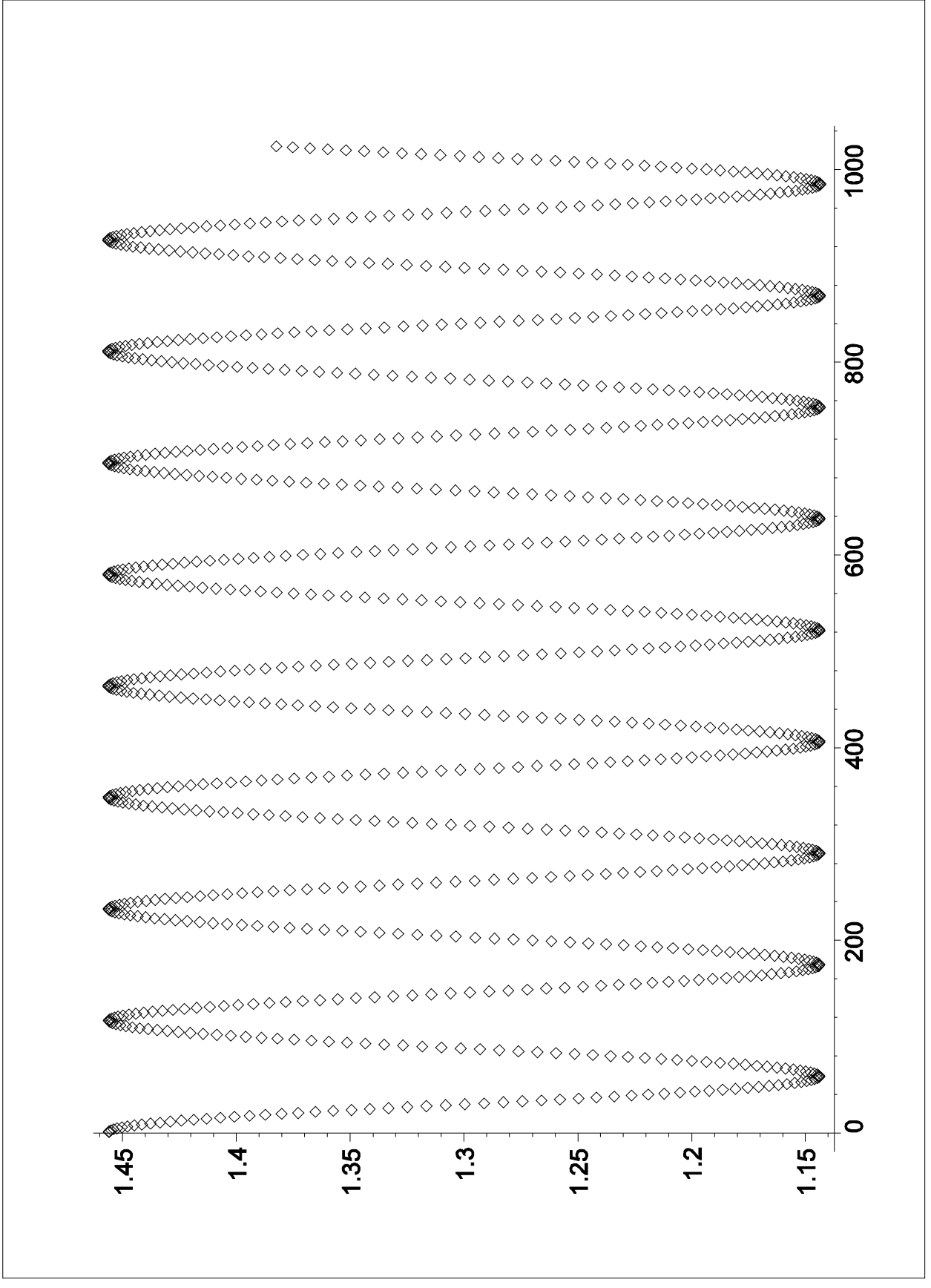


Figure 1: Bond angle stretching normal mode, showing beautiful harmonic motion.

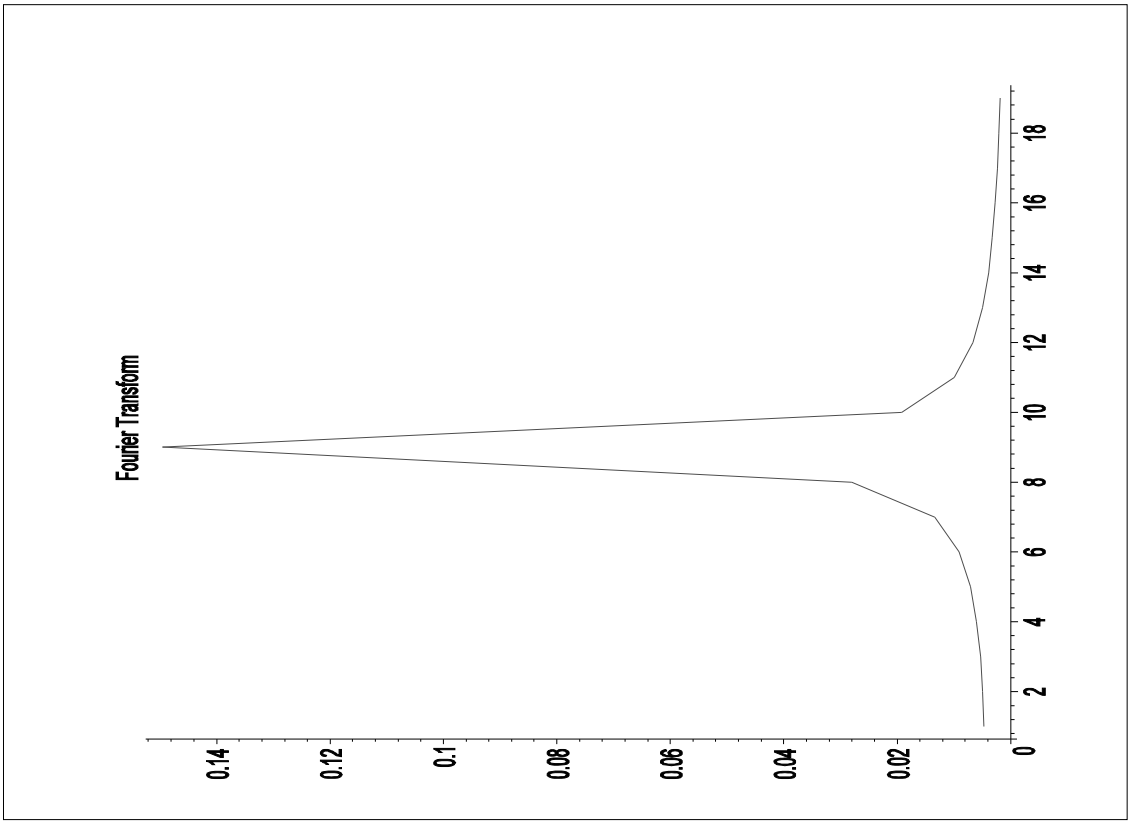


Figure 2. The Fourier Transform of the oscillation shown in Figure 1. There are about 9 cycles in the total motion displayed.

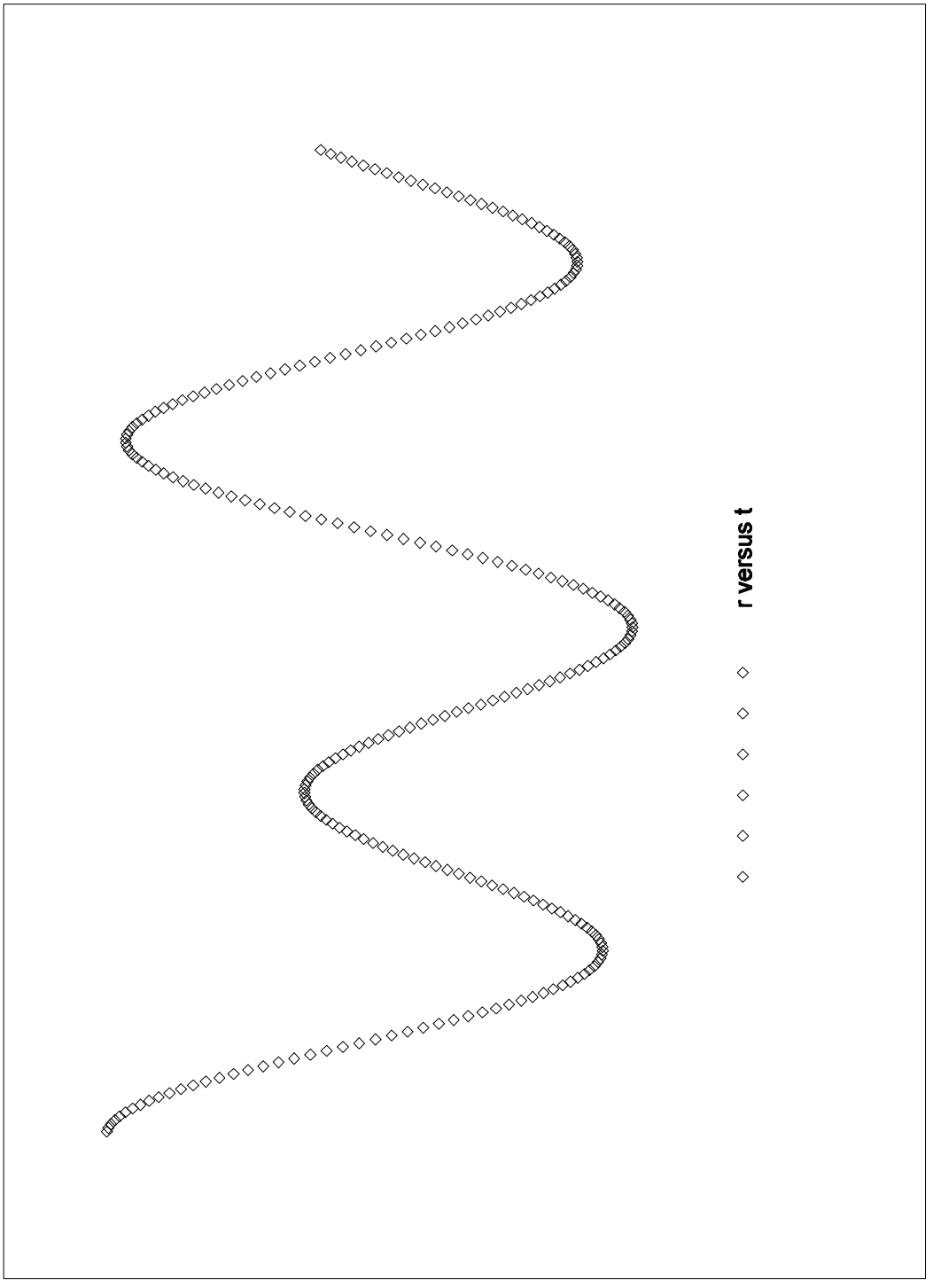
r versus t

Figure 3: The $r_{HH}$ distance as a function of time, showing a complex motion indicative that a single normal mode is not being used.

## Literature Cited


(1) Rempe, S. B.; Jonson, H *Chem. Educator*, **1998**, *3, 1*

(2) David, C. W. *Journal of Chemical Education*, submitted

(3) Verlet, L *Phys. Rev*. **1967***, 98, 159*

(4) 7.684 Millidyne/Angstrom and 0.707*0.954 Millidyne/Angstrom resppectively, Bernath, P. F. *Spectra of Atoms and Molec*ules, Oxford Univ. Press, **1995**, page 223

(5) Barrow, G. M. *Introduction to Molecular Spectroscopy*, **1962**, page 218.

(6) Kuchitsu, K; Bartell, L. S. *J. Chem. Phys*., **1962**, *36, 2460*

(7) Herzberg, G *Molecular Spectra and Molecular Structure*, D. van Nostrand Co., Inc. **1966**, page 585