

```

> #H202_v2 MolDyn
> # March 2006
> # coordinates based on Redington
> # cleaned up and ready for prime time!
> restart;
> with(inttrans):#in order to use the Fourier Transform
> with(linalg):
> printlevel := 0;
> angleRAD := evalf(180/Pi);
> m := 8:# FFT power of 2
The program depends on picking a number of data points as a power to 2 (for efficient Fourier Transformation.,n_stop
= ).Choosing numbers of the order of 5 or 6 leads to swift execution while using numbers of the order of 10 or 11 are
quite slow, although this clearly depends on the CPU speed.
> h := 1.86e-16;
h is the time step, in seconds, and should be modified by the user.
> norm_mode := 6; #choose your normal mode.
norm_mode runs from 1 to 6 in this demonstration case (hydrogen peroxide).
> N_Avogadro := 6.02214199e23;
> m_H_1 := 1.007825035/N_Avogadro;#grams/atom
> m_H_2 := 2.014101779/N_Avogadro;
> m_H := m_H_1;# in case you choose m_H_2 for deuterium
> m_O_16 := 15.99491463/N_Avogadro;#again, one could choose an isotope
> m_O_17 := 16.9991312/N_Avogadro;
> m_O_18 := 17.9991603/N_Avogadro;
> m_O := m_O_16;
> #m_O := m_O*100;#for catching nu_4
What follows is the equilibrium geometric data on hydrogen peroxide that will be assumed:
> r_e := 0.965e-8;#in cm (this is the OH bond length
> r_OO_e := 1.464e-8;
> theta_e := evalf((99.6)*Pi/180);# this is the H-O-O bond angle (eq)
> phi_e := evalf(111.8*(Pi/180));# this is the H-O-O-H bond angle (eq)
> print ('theta_e (equilibrium) = ',evalf(theta_e*180/Pi), ' degrees ');
> print ('phi_e (equilibrium) = ',evalf(phi_e*180/Pi), ' degrees ');

Obviously, this needs to be re-coded for molecules other than hydrogran peroxide. We will not note here
any further when adjustments need to be made to the code, but it should be clear that additional lines of
similar code will be required for more complicated molecules, and extra angle definitions may also be required.

Further, of course, with more atoms comes more partial derivatives, more normal modes, more forces, more
position vectors, more velocity vectors etc..

What follows are the definitions of the trial values of the various force constants in the molecular force field function
(which is defined below)
> kOH := 1020*6.95*10^(-14)*10^(16);#ergs/molecule/cm^2=dyne/cm
> # set to value which generates experimental frequency for vu_1
> #kOH := kOH*0.9;#used to change for sensitivity computation
> k_OO := 500*6.95*10^(-14)*10^(16);#ergs/molecule
> k_theta := 137.0*6.95*10^(-14);#dynes-cm/radians^2
> V_O := 20*6.95*10^(-14);#ergs/molecule
> #kOH := kOH*100;
> #k_OO := k_OO*100;
> #k_theta := k_theta*100;
Utility functions follow:
> mag := proc (RealPart,ImaginaryPart)
> sqrt(RealPart^2+ImaginaryPart^2);
> end proc:
> 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;
Preparatory to obtaining a function (V_eff) for the potential energy function, we define to vectors which will be used
to create this function.
> #calculate the derivatives of the potential with respect to the 12
> coordinates.
> t1 := [x11,x12,x13]-[011,012,013]:
> t2 := [x21,x22,x23]-[021,022,023]:
> t3 := [011,012,013]- [021,022,023];
> t1_mag := sqrt(t1[1]^2+t1[2]^2+t1[3]^2):#01-H1
> t2_mag := sqrt(t2[1]^2+t2[2]^2+t2[3]^2):#02-H2
> t3_mag := sqrt(t3[1]^2+t3[2]^2+t3[3]^2):#01-02
> #theta_123 :=
> evalf(arccos((t1[1]*t3[1]+t1[2]*t3[2]+t1[3]*t3[3])/(t1_mag*t3_mag)));
> theta_123 := angle(t1,t3):
> #theta_234 :=
> evalf(arccos((t3[1]*t2[1]+t3[2]*t2[2]+t3[3]*t2[3])/(t3_mag*t2_mag)));
> theta_234 := angle(-t3,t2):
> V_1 := crossprod(t1,t3):
> V_2 := crossprod(t2,t3):
> phi := angle(V_1,V_2):
> V_eff := (k0H/2)*((t1_mag - r_e)^2+(t2_mag -
> r_e)^2)+(k_00/2)*(t3_mag - r_00_e)^2
> +(V_0/2)*(cos(phi) - cos(phi_e))^2
> +(k_theta/2)*(theta_123 - theta_e)^2
> +(k_theta/2)*(theta_234 - theta_e)^2
> :
> d11 := diff(V_eff,x11):
> d12 := diff(V_eff,x12):
> d13 := diff(V_eff,x13):
> d21 := diff(V_eff,x21):
> d22 := diff(V_eff,x22):
> d23 := diff(V_eff,x23):
> d011 := diff(V_eff,011):
> d012 := diff(V_eff,012):
> d013 := diff(V_eff,013):
> d021 := diff(V_eff,021):
> d022 := diff(V_eff,022):
> d023 := diff(V_eff,023):
> V := proc(r_H_1,r_H_2,r_01,r_02);
> return(evalf(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],
> 011=r_01[1],012=r_01[2],013=r_01[3],021=r_02[1],022=r_02[2],023=r_02[3
> ],V_eff)));
> end proc;
> angle_123 := proc(r_H_1,r_H_2,r_01,r_02) local t1,t3;
> t1 := r_H_1-r_01;
> t3 := r_02-r_01;
> return(evalf(angle(t1,t3)));
> end proc;
> angle_1234 := proc(r_H_1,r_H_2,r_01,r_02) local t1,t2,t3,V_1,V_2;
> t1 := r_H_1-r_01;
> t2 := r_H_2-r_02;
> t3 := r_02-r_01;
> V_1 := crossprod(t1,t3);
> V_2 := crossprod(t2,t3);
> return(evalf(angle(V_1,V_2)));
> end proc;
> E_phi := proc(r_H_1,r_H_2,r_01,r_02)local phi;
```

```

> phi := angle_1234(r_H_1,r_H_2,r_01,r_02);
> #print ('cos( ',phi,) = ',cos(phi),'cos( ',phi_e,) =
> ',cos(phi_e),'term = #' ,evalf((V_0/2)*(cos(phi) - cos(phi_e))^2));
> return(evalf((V_0/2)*(cos(phi) - cos(phi_e))^2));
> end proc;
> mYsub := proc(term) local 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],
> 011=r_01[1],012=r_01[2],013=r_01[3],021=r_02[1],022=r_02[2],023=r_02[3]
> ],
> term):
> return(-evalf(t5));
> end proc;
> mYsubP := proc(term) local 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],x22=r
> _H_2_p[2],x23=r_H_2_p[3],
> 011=r_01_p[1],012=r_01_p[2],013=r_01_p[3],021=r_02_p[1],022=r_02_p[2],
> 023=r_02_p[3],
> term):
> return(-evalf(t5));
> end proc;

```

For each normal mode, we will present the starting coordinates employed, obtained from the JmakeMol program (CDNT converted to java).

hydrogen peroxide coordinates undistorted, r(OH)=0.965,r(OO)=1.464; theta(HOO)= 99.6 degrees and phi(HOOH)= 111.8 degrees, given in pdb format: HETATM 1 H 1 0.000 0.000 0.000 1.00 0.00

HETATM 2 O 1 0.000 0.000 -0.965 1.00 0.00

HETATM 3 O 1 1.443 0.000 -1.209 1.00 0.00

HETATM 4 H 1 1.543 -0.883 -1.584 1.00 0.00

We had some premature truncations in function evaluations with zero set to "0", so for safety sake, we used a small value instead. Perhaps this workaround is unnecessary.

```
> zero := 1e-20;
```

```
> if norm_mode = 0 then
```

There is no such thing as a normal mode of zero, so this one represents using any starting coordinates you desire.

```
> r_01 := [zero ,zero,-0.965]*1e-8:#set this in common, but change
```

```
> later
```

```
> r_02 := [ 1.542, zero, -1.226]*1e-8:
```

```
> r_H_1 := [zero,zero,zero]*1e-8:
```

```
> r_H_2 := [1.642, -0.883, -1.601]*1e-8:
```

```
> elif norm_mode = 1 then
```

start of mode 1, symmetric OH stretch. rOH increased to 1.065 from 0.965 for both OH HETATM 1 H 1 0.000 0.000 0.000 1.00 0.00

HETATM 2 O 1 0.000 0.000 -1.065 1.00 0.00

HETATM 3 O 1 1.443 0.000 -1.309 1.00 0.00

HETATM 4 H 1 1.554 -0.975 -1.723 1.00 0.00 end of mode 1, symmetric OH stretch. rOH increased to 1.065 from 0.965

```
> print(' symmetric (OH) stretch normal mode = 1');
```

```
> r_01 := [zero ,zero,-1.065]*1e-8:#set this in common, but change
```

```
> later
```

```
> r_02 := [1.443 , zero, -1.309]*1e-8;
```

```
> r_H_1 := [zero,zero,zero]*1e-8:
```

```
> r_H_2 := [1.554, -0.975 , -1.723 ]*1e-8:
```

```
> legend := 'symmetric (OH) stretch normal mode = 1';
```

```
> elif norm_mode = 2 then
```

start of mode 2, O-O-H symmetric bend, 99.6 -> 100.6 (1 degree distortion) note small errors in truncated coordinates here result in slight changes in the angles reported HETATM 1 H 1 0.000 0.000 0.000 1.00 0.00

HETATM 2 O 1 0.000 0.000 -0.965 1.00 0.00

HETATM 3 O 1 1.439 0.000 -1.234 1.00 0.00

```

HETATM 4 H 1 1.549 -0.881 -1.613 1.00 0.00 end of mode 2, O-O-H symmetric bend, 99.6 -> 100.6
> print(' theta varying normal mode ');
> r_01 := [zero ,zero,-0.965]*1e-8;#set this in common, but change
> later
> r_02 := [ 1.439, zero,-1.234]*1e-8;
> r_H_1 := [zero,zero,zero]*1e-8:
> r_H_2 := [ 1.549, -0.881, -1.613 ]*1e-8:
> legend := 'symmetric (OHH) bend normal mode = 2';
> elif norm_mode = 3 then
> #ANTISYMMETRIC STRETCH
> print(' 0-0 stretch normal mode');
HETATM 1 H 1 0.000 0.000 0.000 1.00 0.00
HETATM 2 O 1 0.000 0.000 -0.965 1.00 0.00
HETATM 3 O 1 1.542 0.000 -1.226 1.00 0.00
HETATM 4 H 1 1.642 -0.883 -1.601 1.00 0.00
> r_01 := [zero ,zero,-0.965]*1e-8;#set this in common, but change
> later
> r_02 := [1.542 , zero, -1.226]*1e-8;
> r_H_1 := [zero,zero,zero]*1e-8:
> r_H_2 := [1.642, -0.883 , -1.601 ]*1e-8:
> legend := 'anti symmetric (OH) stretch normal mode = 3';
> elif norm_mode = 4 then
start of dihedral angle mode, 111.8->112.8 (phi) mode 4 HETATM 1 H 1 0.000 0.000 0.000 1.00 0.00
HETATM 2 O 1 0.000 0.000 -0.965 1.00 0.00
HETATM 3 O 1 1.443 0.000 -1.209 1.00 0.00
HETATM 4 H 1 1.541 -0.877 -1.600 1.00 0.00
#endif of dihedral angle mode
> legend := 'normal mode = 4, dihedral angle (phi)';
> print(' dihedral angle (phi) mode 4 (2 degree distortion)');
> r_01 := [zero ,zero,-0.965]*1e-8;#set this in common, but change
> later
> r_02 := [ 1.443, zero,-1.209]*1e-8;
> r_H_1 := [zero,zero,zero]*1e-8:
> r_H_2 := [ 1.541, -0.877, -1.600 ]*1e-8:
> elif norm_mode = 5 then
# begin of mode 5 of asym OH stretch #HETATM 1 H 1 0.000 0.000 0.000 1.00 0.00
#HETATM 2 O 1 0.000 0.000 -0.865 1.00 0.00
#HETATM 3 O 1 1.443 0.000 -1.109 1.00 0.00
#HETATM 4 H 1 1.554 -0.975 -1.523 1.00 0.00
#endif of asym OH stretch
> r_01 := [zero ,zero,-0.865]*1e-8;#set this in common, but change
> later
> r_02 := [ 1.443, zero,-1.109]*1e-8;
> r_H_1 := [zero,zero,zero]*1e-8:
> r_H_2 := [ 1.554, -0.975, -1.523 ]*1e-8:
> legend := 'normal mode = 5, asymmetric O-H stretch';
> elif norm_mode = 6 then
begin of mode 6, asymmetric O-O-H ebnd HETATM 1 H 1 0.000 0.000 0.000 1.00 0.00
HETATM 2 O 1 0.000 0.000 -0.965 1.00 0.00
HETATM 3 O 1 1.448 0.000 -1.184 1.00 0.00
HETATM 4 H 1 1.570 -0.881 -1.559 1.00 0.00 end of asym O-O-H bend
> r_01 := [zero ,zero,-0.965]*1e-8;#set this in common, but change
> later
> r_02 := [ 1.448, zero,-1.184]*1e-8;
> r_H_1 := [zero,zero,zero]*1e-8:
> r_H_2 := [ 1.570, -0.881, -1.559 ]*1e-8:
> legend := 'symmetric (OHH) bend normal mode = 2';
> end if;
I found it useful for documentation purposes to print out some controlling information:
> print ('h1-h2 dist = ',dist(r_H_1,r_H_2));

```

```

> print ('01-h1 dist = ',dist(r_01,r_H_1));
> print ('02-h2 dist = ',dist(r_02,r_H_2));
> print ('01-02 dist = ',dist(r_01,r_02));
> t1 := r_H_1-r_01;
> t2 := r_H_2-r_02;
> t3 := r_02-r_01;
> t4 := r_H_2-r_H_1;
> print('angle_123 = ',evalf(angle(t1,t3)*180/Pi),` and angle_234
> =`,evalf(angle(-t3,t2)*180/Pi));
> V_1 := crossprod(t1,t3);
> V_2 := crossprod(t2,t3);
> phi := evalf(angle(V_1,V_2)*180/Pi);
> print('phi = ',phi);
> print ('starting H1 coords = ',r_H_1);
> print ('starting H2 coords = ',r_H_2);
> print ('starting O coords = ',r_01);
> print ('starting O coords = ',r_02);
Initialization of velocities:
> 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_01 := [0,0,0];
> v_01_p := [0,0,0];
> v_02 := [0,0,0];
> v_02_p := [0,0,0];
Initializing forces:
> F_H_1 := [0,0,0];
> F_H_2 := [0,0,0];
> F_01 := [0,0,0];
> F_02 := [0,0,0];
> F_H_1_p := [0,0,0];
> F_H_2_p := [0,0,0];
> F_01_p := [0,0,0];
> F_02_p := [0,0,0];
Calculate the initial potential energy:
> V_start :=V(r_H_1,r_H_2,r_01,r_02);
> print ('starting potential energy = ',V_start);
> n_stop := 2^m;
> l := array(1..n_stop);
> y := array(1..n_stop);
And away we go (this dates me, from Jackie Gleason ...)
> for i from 1 by 1 to n_stop do
I found it valuable to check on values at the beginning and end of the run:
> printlevel := 0;# DEBUG STATEMENTS
> if i=1 or i = n_stop then printlevel := 5; end if;
Start: evaluate the instantaneous force on each nucleus in each direction, x, y, and z
> F_H_1[1] := mYsub(d11);
> F_H_1[2] := mYsub(d12);
> F_H_1[3] := mYsub(d13);
> F_H_2[1] := mYsub(d21);
> F_H_2[2] := mYsub(d22);
> F_H_2[3] := mYsub(d23);
> F_01[1] := mYsub(d011);
> F_01[2] := mYsub(d012);
> F_01[3] := mYsub(d013);
> F_02[1] := mYsub(d021);
> F_02[2] := mYsub(d022);
> F_02[3] := mYsub(d023);

```

```

Calculate the next coordinates via the Verlet 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_01_p := r_01 + h*v_01 + ((h^2)/(2*m_0))*F_01;
> r_02_p := r_02 + h*v_02 + ((h^2)/(2*m_0))*F_02;
Calculate the primed force under the new coordinates:
> F_H_1_p[1] := mYsubP(d11):
> F_H_1_p[2] := mYsubP(d12);
> F_H_1_p[3] := mYsubP(d13):
> F_H_2_p[1] := mYsubP(d21):
> F_H_2_p[2] := mYsubP(d22):
> F_H_2_p[3] := mYsubP(d23):
> F_01_p[1] := mYsubP(d011):
> F_01_p[2] := mYsubP(d012):
> F_01_p[3] := mYsubP(d013):
> F_02_p[1] := mYsubP(d021):
> F_02_p[2] := mYsubP(d022):
> F_02_p[3] := mYsubP(d023):
Calculate the new velocities a la the Verlet algorithim:
> 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_01_p := v_01 + (h/(2*m_0))*(F_01_p + F_01):
> v_02_p := v_02 + (h/(2*m_0))*(F_02_p + F_02):
Update the positions from primed back to original:
> r_H_1 := r_H_1_p:#update coordinates
> r_H_2 := r_H_2_p:
> r_01 := r_01_p:
> r_02 := r_02_p:
Update the velocities from primed back to original:
> v_H_1 := v_H_1_p:#update velocities
> v_H_2 := v_H_2_p:
> v_01 := v_01_p:
> v_02 := v_02_p:
Calculate the precursors to the kinetic energy:
> v_H_1_mag_sq := (dotprod(v_H_1,v_H_1));#for kinetic energy
> v_H_2_mag_sq := (dotprod(v_H_2,v_H_2));
> v_01_mag_sq := (dotprod(v_01,v_01));
> v_02_mag_sq := (dotprod(v_02,v_02));
Ignore the imaginary part of the Fourier Transform:
> y[i] := 0;#imaginary part
> KE(i) :=
> 1/2*(m_H*(v_H_1_mag_sq+v_H_2_mag_sq)+m_0*(v_01_mag_sq+v_02_mag_sq)):#KE
> PE(i) := V(r_H_1,r_H_2,r_01,r_02):
> E_total(i) := evalf(KE(i)+PE(i)):
> E_totaloverV[i] := (E_total(i)/V_start);
> E_totaloverV_plot(i) := (E_total(i)/V_start)*100;
> l[i] := dist(r_H_1,r_H_2)*1e8:#convert to angstrom
> if (norm_mode = 1 or norm_mode = 5) then l[i] := dist(r_01,r_H_1);
> elif (norm_mode = 2 or norm_mode = 6) then l[i] :=
> angle_123(r_H_1,r_H_2,r_01,r_02);
> elif (norm_mode = 3) then l[i] := dist(r_01,r_02)*1e8;
> elif (norm_mode = 4) then l[i] :=
> angle_1234(r_H_1,r_H_2,r_01,r_02);
> elif (norm_mode = 5) then l[i] := dist(r_01,r_H_1);
> end if;
> r_plot(i) := l[i];
> theta_plot(i) := angle_123(r_H_1,r_H_2,r_01,r_02)*angleRAD;
> phi_plot(i) := angle_1234(r_H_1,r_H_2,r_01,r_02)*angleRAD;
> #used in debugging normal mode 3 :energy_of_phi(i) :=
> E_phi(r_H_1,r_H_2,r_01,r_02);

```

```

> printlevel := 0;
> end do;
> #cwd := [[ n, energy_of_phi(n)] $n=1..n_stop];
> #plot(cwd, x=0..n_stop, style=LINE,symbol=circle);
And so we're done. We've done n_stop time steps, saved the distances, angles, and energies All we need to do is
reset the variable "i", why I don't know, and then do the Fourier Transform
> i := 'i':#reset the variable "i"
> FFT(m,l,y);
The normal mode dependent legend is now printed
> print(legend);
We could be printing a distance and we could be printing an angle.
> PLOT(POINTS(seq([i,r_plot(i)],i=1..n_stop),
> SYMBOL(DIAMOND),LEGEND('r_plot() versus t')));
Now we print the angles:
> print(' theta_plot(i):');
> PLOT(POINTS(seq([i,theta_plot(i)],i=1..n_stop),
> SYMBOL(DIAMOND),LEGEND('theta versus t')));
> print(' phi_plot(i):');
> PLOT(POINTS(seq([i,phi_plot(i)],i=1..n_stop),
> SYMBOL(DIAMOND),LEGEND('phi versus t')));
> i := 'i':#reset the variable "i"
We plot only part of the Fourier Transform, since most of it is uninteresting (however, we show how to do the whole
thing in case you want to).
> 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..10)],title="TRUNCATED Fourier
> Transform");#40 for m=11
> #use the line above to see more detail, changing 10 to some low number
> in order to
> #discern where the maximum is.
The following printing allows us to easily compute the spectroscopic quantities needed:
> 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):');
> 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_plot(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]);
We've been unable to label these plots with proper axes, for reasons beyond my comprehension, so rather than
waste time, I leave these as they are. Someone who can do a better job is hereby cordially invited to do so and
explain to me how to label these plots better. Thanks. Carl David Department of Chemistry University of
Connecticut Storrs, Connecticut 06269-3060 Carl.David@uconn.edu (860)486-3217

Warning, the name hilbert has been redefined
Warning, the protected names norm and trace have been redefined and
unprotected

```

```

printlevel := 0
m := 8
h := 0.186 10-15
norm_mode := 6
N_Avogadro := 0.602214199 1024
m_H_1 := 0.1673532502 10-23
m_H_2 := 0.3344494006 10-23
m_H := 0.1673532502 10-23
m_O_16 := 0.2656017519 10-22
m_O_17 := 0.2822771570 10-22
m_O_18 := 0.2988830275 10-22
m_O := 0.2656017519 10-22
r_e := 0.965 10-8
r_OO_e := 0.1464 10-7
theta_e := 1.738347935
theta_e (equilibrium) = , 99.59999997, degrees
phi_e (equilibrium) = , 111.8000000, degrees
kOH := 708900.0000
k_OO := 347500.0000
k_theta := 0.9521500000 10-11
V_O := 0.1390000000 10-11
t3 := [-O21 + O11, -O22 + O12, -O23 + O13]

E_phi := proc(r_H_1, r_H_2, r_O1, r_O2)
local phi;
phi := angle_1234(r_H_1, r_H_2, r_O1, r_O2);
return evalf(1/2 * V_O * (cos(phi) - cos(phi_e))2)
end proc

zero := 0.1 10-19
h1 - h2 dist = , 0.2381499948 10-7
O1 - h1 dist = , 0.9650000000 10-8
O2 - h2 dist = , 0.9652305424 10-8
O1 - O2 dist = , 0.1464467480 10-7
t1 := [0., 0., 0.965 10-8]
t2 := [0.122 10-8, -0.881 10-8, -0.375 10-8]
t3 := [0.1448 10-7, 0., -0.219 10-8]
t4 := [0.1570 10-7, -0.881 10-8, -0.1559 10-7]
angle_123 = , 98.60040962, and angle_234 = , 100.5487384
V_1 := [-0., 0.1397320 10-15, 0.]
V_2 := [0.192939 10-16, -0.516282 10-16, 0.1275688 10-15]
phi := 111.8092058
phi = , 111.8092058
starting H1 coords = , [0.1 10-27, 0.1 10-27, 0.1 10-27]
starting H2 coords = , [0.1570 10-7, -0.881 10-8, -0.1559 10-7]
starting O coords = , [0.1 10-27, 0.1 10-27, -0.965 10-8]
starting O coords = , [0.1448 10-7, 0.1 10-27, -0.1184 10-7]
starting potential energy = , 0.1069137112 10-11
n_stop := 256
l := array(1..256, [])

```

```

y := array(1..256, [])
printlevel := 5
F_H_1 := 0.000313427132
F_H_2 := -0.2017655501 10-7
F_H_3 := -0.2073556130 10-17
F_H_2_1 := -0.000340930719
F_H_2_2 := -0.0000574773375
F_H_2_3 := 0.0000283239399
F_O1_1 := -0.0003554082724
F_O1_2 := -0.0002122534423
F_O1_3 := -0.0002884370236
F_O2_1 := 0.000382911860
F_O2_2 := 0.000269750954
F_O2_3 := 0.0002601130849
r_H_1_p := [0.3239651768 10-11, -0.2085493100 10-15, -0.2133273220 10-25]
r_H_2_p := [0.1569647607 10-7, -0.8810594098 10-8, -0.1558970724 10-7]
r_O1_p := [-0.2314688156 10-12, -0.1382355356 10-12, -0.9650187852 10-8]
r_O2_p := [0.1448024938 10-7, 0.1756822750 10-12, -0.1183983059 10-7]
F_H_1_p1 := 0.000313037402
F_H_1_p2 := -0.1702988744 10-7
F_H_1_p3 := -0.2462090849 10-6
F_H_2_p1 := -0.000340541720
F_H_2_p2 := -0.0000571614428
F_H_2_p3 := 0.0000283895049
F_O1_p1 := -0.0003548065851
F_O1_p2 := -0.0002119933577
F_O1_p3 := -0.0002878569552
F_O2_p1 := 0.000382310903
F_O2_p2 := 0.000269171830
F_O2_p3 := 0.0002597136588
v_H_1_p := [34813.30754, -2.067602000, -13.68210350]
v_H_2_p := [-37870.15595, -6370.600245, 3151.627089]
v_O1_p := [-2486.805199, -1485.492928, -2017.883528]
v_O2_p := [2679.414441, 1887.028928, 1820.164469]
r_H_1 := [0.3239651768 10-11, -0.2085493100 10-15, -0.2133273220 10-25]
r_H_2 := [0.1569647607 10-7, -0.8810594098 10-8, -0.1558970724 10-7]
r_O1 := [-0.2314688156 10-12, -0.1382355356 10-12, -0.9650187852 10-8]
r_O2 := [0.1448024938 10-7, 0.1756822750 10-12, -0.1183983059 10-7]
v_H_1 := [34813.30754, -2.067602000, -13.68210350]
v_H_2 := [-37870.15595, -6370.600245, 3151.627089]
v_O1 := [-2486.805199, -1485.492928, -2017.883528]
v_O2 := [2679.414441, 1887.028928, 1820.164469]
v_H_1_mag_sq := 0.1211966573 1010
v_H_2_mag_sq := 0.1484666012 1010
v_O1_mag_sq := 0.1246274327 108
v_O2_mag_sq := 0.1405313861 108

```

```

 $y_1 := 0$ 
 $\text{KE}(1) := 0.2608584372 \cdot 10^{-14}$ 
 $\text{PE}(1) := 0.1066526829 \cdot 10^{-11}$ 
 $\text{E\_total}(1) := 0.1069135413 \cdot 10^{-11}$ 
 $E\_totaloverV_1 := 0.9999984109$ 
 $\text{E\_totaloverV\_plot}(1) := 99.99984109$ 
 $l_1 := 2.381056920$ 
 $l_1 := 1.720513067$ 
 $r\_plot(1) := 1.720513067$ 
 $\text{theta\_plot}(1) := 98.57813731$ 
 $\text{phi\_plot}(1) := 111.8111746$ 
 $printlevel := 5$ 
 $F\_H\_1\_1 := 0.000305197137$ 
 $F\_H\_1\_2 := -0.1307095325 \cdot 10^{-5}$ 
 $F\_H\_1\_3 := 0.00001013809045$ 
 $F\_H\_2\_1 := -0.000323655776$ 
 $F\_H\_2\_2 := -0.0000629361585$ 
 $F\_H\_2\_3 := 0.0000247124729$ 
 $F\_O1\_1 := -0.0004355431840$ 
 $F\_O1\_2 := -0.0002046896954$ 
 $F\_O1\_3 := -0.0002815525987$ 
 $F\_O2\_1 := 0.000454001824$ 
 $F\_O2\_2 := 0.000268932950$ 
 $F\_O2\_3 := 0.0002467020350$ 
 $r\_H\_1\_p := [0.2091066882 \cdot 10^{-9}, 0.5439666931 \cdot 10^{-10}, -0.4734796627 \cdot 10^{-10}]$ 
 $r\_H\_2\_p := [0.1541498208 \cdot 10^{-7}, -0.8779725932 \cdot 10^{-8}, -0.1560383090 \cdot 10^{-7}]$ 
 $r\_O1\_p := [0.1383543539 \cdot 10^{-9}, -0.3352356787 \cdot 10^{-11}, -0.9670962101 \cdot 10^{-8}]$ 
 $r\_O2\_p := [0.1434642879 \cdot 10^{-7}, -0.1982673767 \cdot 10^{-11}, -0.1181518309 \cdot 10^{-7}]$ 
 $F\_H\_1\_p_1 := 0.000306807507$ 
 $F\_H\_1\_p_2 := -0.1280782595 \cdot 10^{-5}$ 
 $F\_H\_1\_p_3 := 0.00001615062335$ 
 $F\_H\_2\_p_1 := -0.000325547785$ 
 $F\_H\_2\_p_2 := -0.0000720879761$ 
 $F\_H\_2\_p_3 := 0.0000212023886$ 
 $F\_O1\_p_1 := -0.0004431445982$ 
 $F\_O1\_p_2 := -0.0002065306221$ 
 $F\_O1\_p_3 := -0.0002885202233$ 
 $F\_O2\_p_1 := 0.000461884876$ 
 $F\_O2\_p_2 := 0.000279899381$ 
 $F\_O2\_p_3 := 0.0002511672102$ 
 $v\_H\_1\_p := [-3619.46631, -6486.936901, -42717.70251]$ 
 $v\_H\_2\_p := [54448.33646, 65621.51210, 25733.25480]$ 
 $v\_O1\_p := [41248.60908, 2842.055909, -2321.204911]$ 
 $v\_O2\_p := [-44451.28966, -6568.071878, 3391.379644]$ 
 $r\_H\_1 := [0.2091066882 \cdot 10^{-9}, 0.5439666931 \cdot 10^{-10}, -0.4734796627 \cdot 10^{-10}]$ 
 $r\_H\_2 := [0.1541498208 \cdot 10^{-7}, -0.8779725932 \cdot 10^{-8}, -0.1560383090 \cdot 10^{-7}]$ 

```

```

r_O1 := [0.1383543539 10-9, -0.3352356787 10-11, -0.9670962101 10-8]
r_O2 := [0.1434642879 10-7, -0.1982673767 10-11, -0.1181518309 10-7]
v_H_1 := [-3619.46631, -6486.936901, -42717.70251]
v_H_2 := [54448.33646, 65621.51210, 25733.25480]
v_O1 := [41248.60908, 2842.055909, -2321.204911]
v_O2 := [-44451.28966, -6568.071878, 3391.379644]
v_H_1_mag_sq := 0.1879982995 1010
v_H_2_mag_sq := 0.7933004596 1010
v_O1_mag_sq := 0.1714913025 1010
v_O2_mag_sq := 0.2030558176 1010
y256 := 0
KE(256) := 0.5795136248 10-13
PE(256) := 0.1011156924 10-11
E_total(256) := 0.1069108286 10-11
E_totaloverV_256 := 0.9999730381
E_totaloverV_plot(256) := 99.99730381
l256 := 2.347902318
l256 := 1.713226691
r_plot(256) := 1.713226691
theta_plot(256) := 98.16065872
phi_plot(256) := 112.5737123
256
symmetric (OHH) bend normal mode = 2
theta_plot(i):
phi_plot(i):
frequency := 0.2100134409 1014 number_of_cycles
sec^ - 1 = , 0.2100134409 1014 number_of_cycles, times # of peaks
cm^ - 1 = , 700.0448029 number_of_cycles, times # of peaks
KE(i):
KE(i)
PE(i):
E_total(i):

```

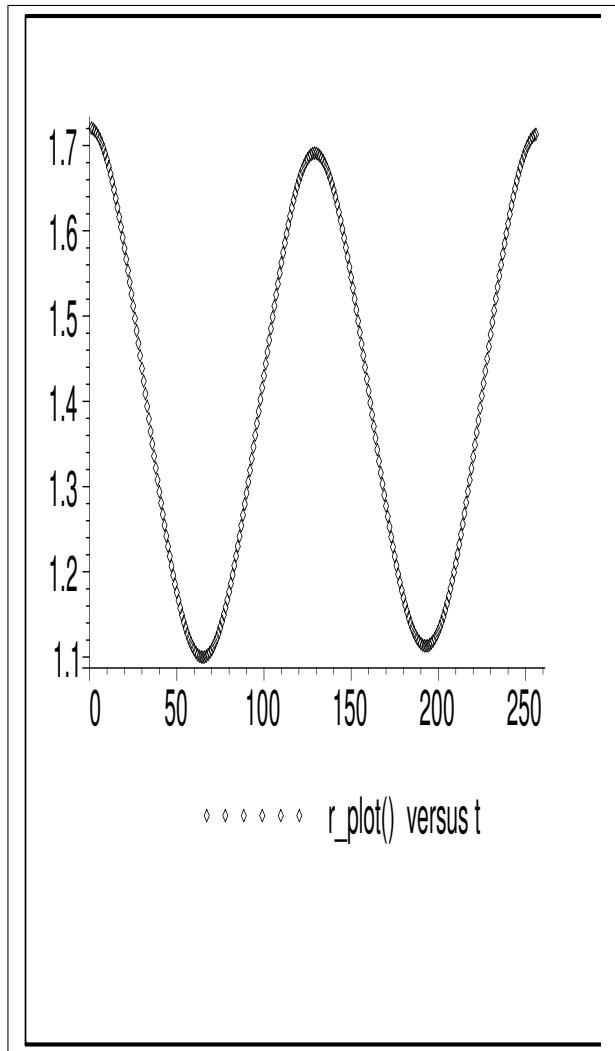


FIG. 1: A plot of the $H_1 - H_2$ distance versus time (step)

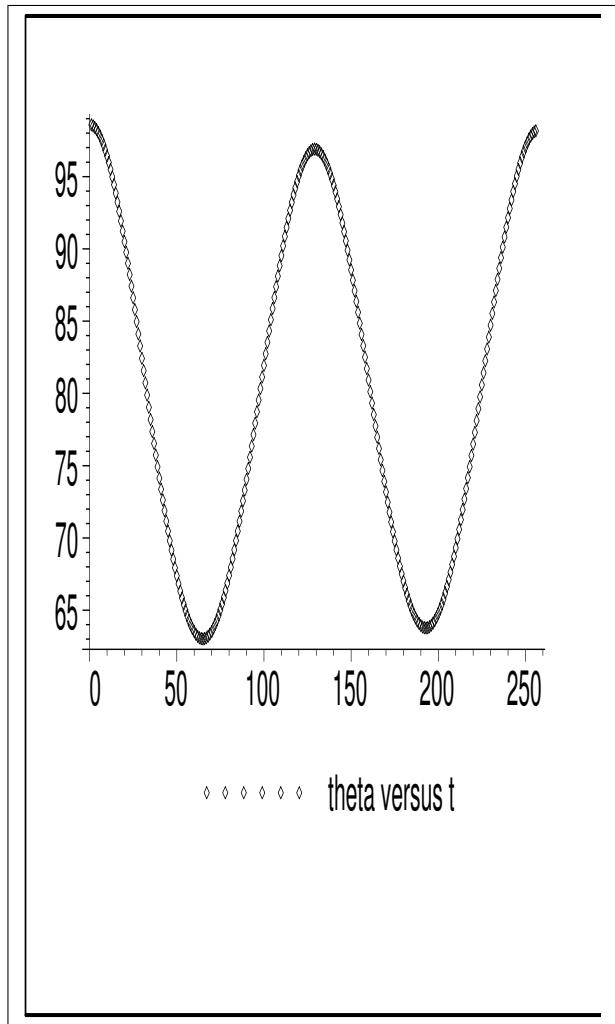


FIG. 2: A plot of the $H_1O_1O_2$ bond angle versus time (step)

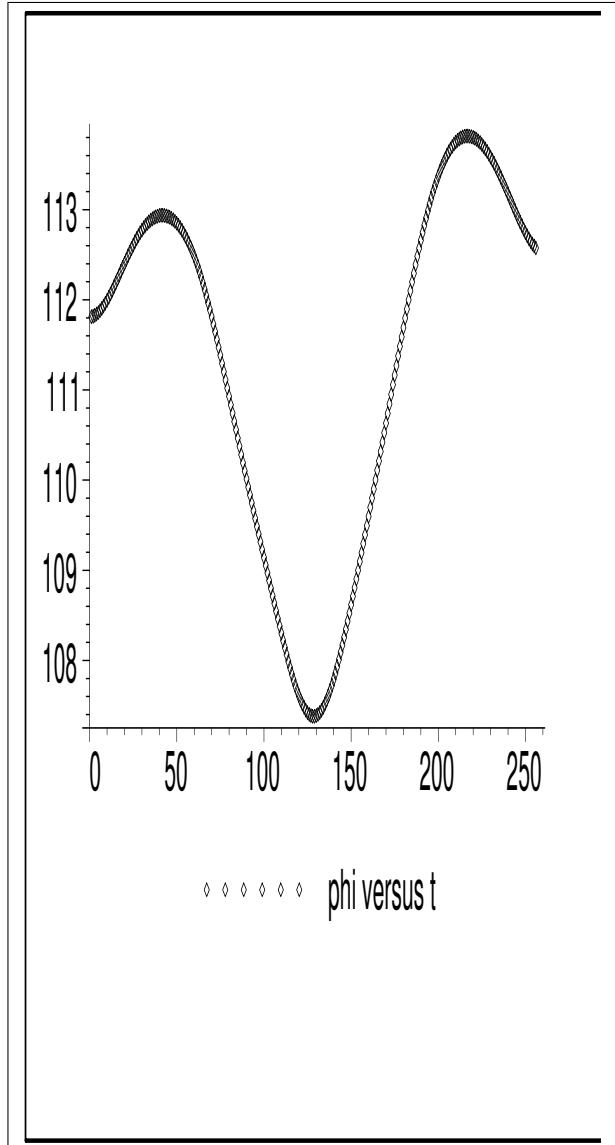


FIG. 3: A plot of the torsional angle versus time (step)

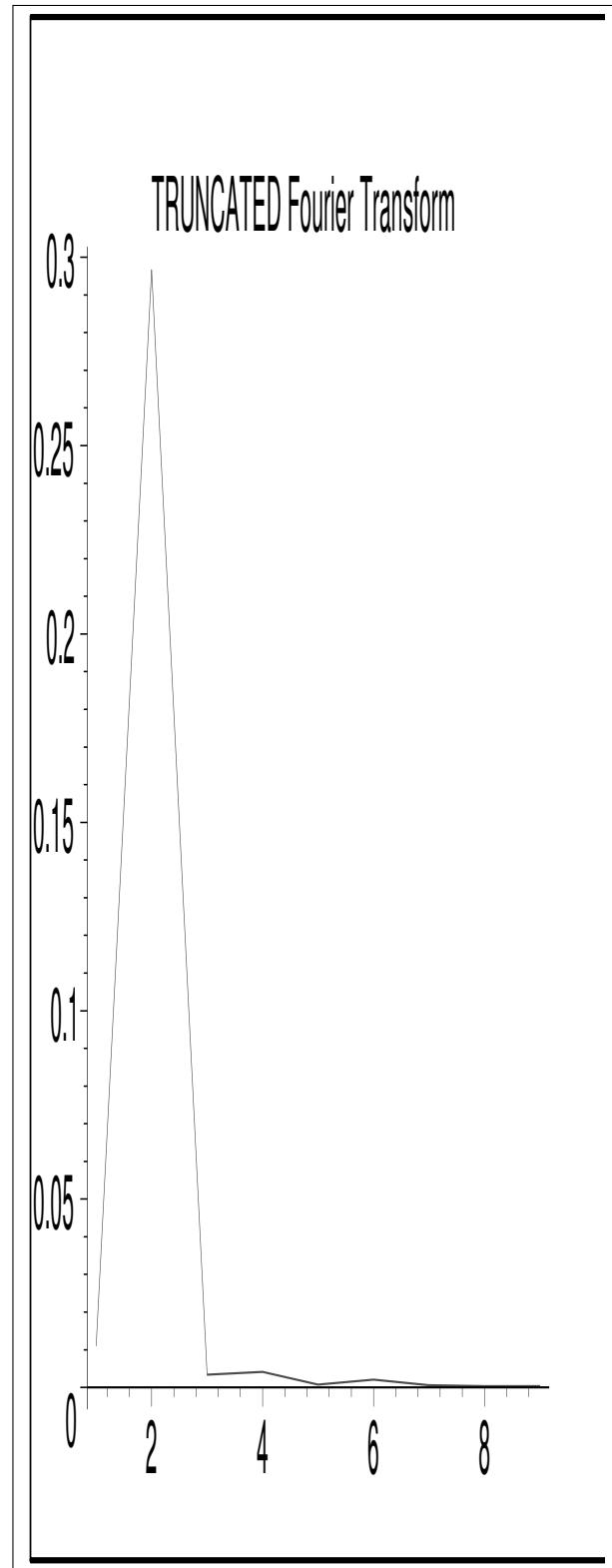


FIG. 4: The Fourier Series indicating a peak!

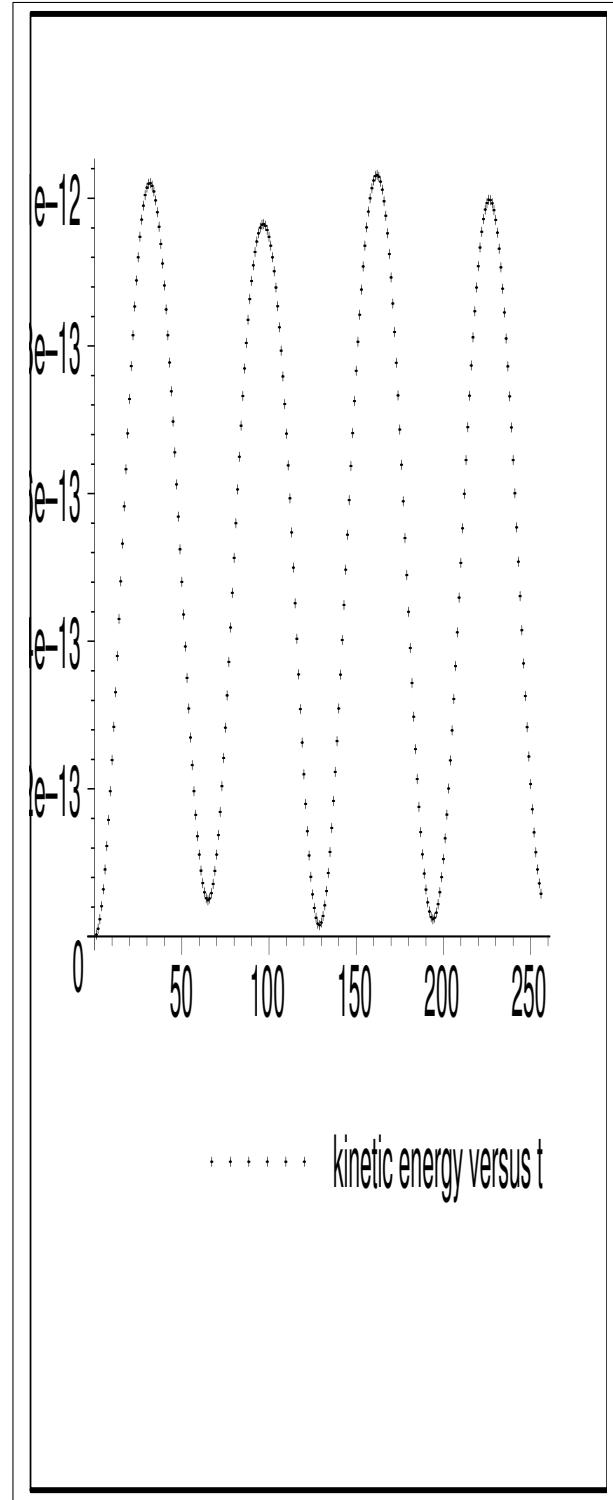


FIG. 5: A plot of the kinetic energy versus time (step) showing a tiny variation!

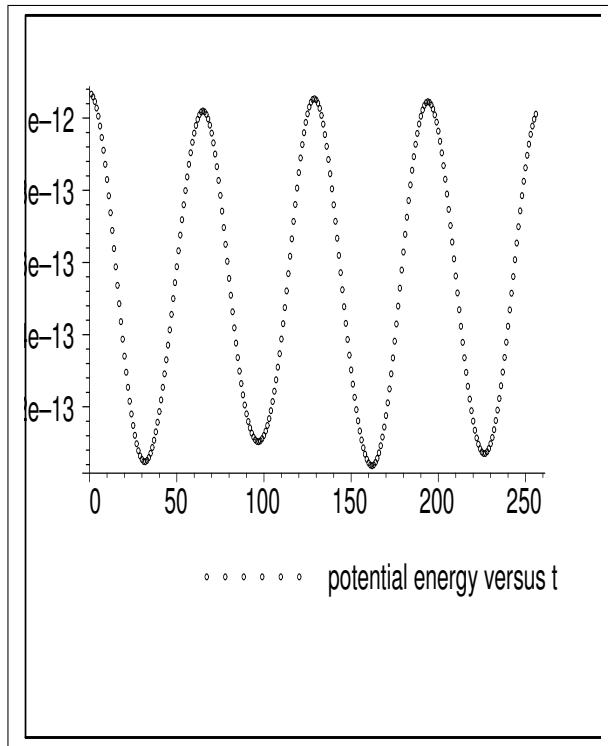


FIG. 6: A plot of the potential energy versus time (step) showing a tiny cyclic variation!

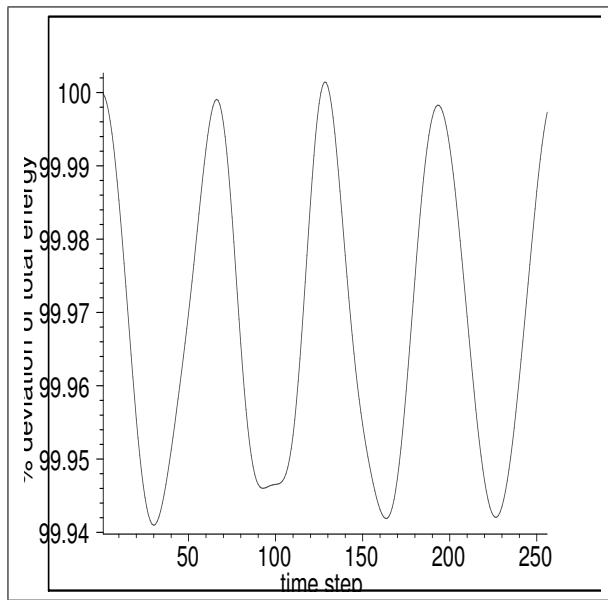


FIG. 7: A plot of the total energy (KE +PE) percent variation as a function of the time (step).