Simulations of the Angular Dependence of the Dipole-Dipole Interaction

Jacob T. Paul  
*Ursinus College, japaul@ursinus.edu*

Matan Peleg  
*Ursinus College, mapeleg@ursinus.edu*

Follow this and additional works at: http://digitalcommons.ursinus.edu/physics_astro_sum

**Recommended Citation**
http://digitalcommons.ursinus.edu/physics_astro_sum/2

This Paper is brought to you for free and open access by the Student Research at Digital Commons @ Ursinus College. It has been accepted for inclusion in Physics and Astronomy Summer Fellows by an authorized administrator of Digital Commons @ Ursinus College. For more information, please contact aprock@ursinus.edu.
The following code is a simulation of two atoms moving between the d, p, and manifold states.

(* define codes for atomic states *)
dneg = 1;
dpos = 2;
pneg = 3;
ppos = 4;
mneg = 5;
mpos = 6;
(* define list of states *)
phi = {{dneg, dneg}, {dneg, dpos}, {dpos, dneg}, {dpos, dpos},
{pneg, mneg}, {pneg, mpos}, {mpos, mneg}, {mpos, mpos},
{mneg, pneg}, {mneg, ppos}, {mpos, pneg}, {mpos, ppos}};
(* define the possible states *)
mjArray = {0, 0, 0, 0, 0, 0};
mjArray[[dneg]] = -5/2;
mjArray[[dpos]] = 5/2;
mjArray[[pneg]] = -3/2;
mjArray[[ppos]] = 3/2;
mjArray[[mneg]] = -3/2;
mjArray[[mpos]] = 3/2;
(* List of constant variables, 
H is the Matrix we plan to fill and make hermatian , 
k is the number of atoms we use, 
here we are using one and minimize code by using k and k+1, 
theta is the angle for the two atoms , 
mu and nu are distances between states where R is the distance between atoms . *)
H = ConstantArray[0, {12, 12}];
k = 1;
θ = π/4;
μ = 4;
ν = 1;
R = 1;
(* Angular is a function that takes in theta and delta 
j 1 and 2. Here is where we decide that delta MJ is one of 
three options that we derived from a series of integrals. *)
Angular[θ_, Δmj1_, Δmj2_] := Module[{},
If[Δmj1 + Δmj2 == 0, Return[1 - (3*Cos[θ]^2)]];
If[Abs[Δmj1 + Δmj2] == 1, Return[(3/2)*Sin[θ]*Cos[θ]]];
If[Abs[Δmj1 + Δmj2] == 2, Return[(1/4)*Sin[θ]^2]];
Return[0];
]
(* The major for loops are below that create and fill 
H. Inside this loop we have four sets of four if statements 
that are the allowed states for the atoms to move , 
before this though we initialize u as an equation that accounts for 
distance between states and the atoms and calls angular. The first 
set of if statements is for when dneg or dpos goes to ppos or pneg, 
and if dneg or dpos goes to mneg or mpos . The second set of if statements 
is if dpos or dneg goes to mneg or mpos then dneg or dpos goes to 
ppos or pneg. The third case is for when the first atom starts in 
ppos or pneg and goes to dneg or dpos and the second starts in mneg 
or mpos and moves to dpos or dneg. The last case is for when the 
first atom starts in mneg or mpos and moves to dneg or dpos and 
the second atom starts in ppos or pneg and moves to dpos or dneg. *)
For $i = 1, i \leq 12, i++$
    For $j = i, j \leq 12, j++$
        $\Delta m_{j1} = \text{mArray}[[i, k]] - \text{mArray}[[i, k + 1]]$
        $\Delta m_{j2} = \text{mArray}[[j, k]] - \text{mArray}[[i, k + 1]]$

        $u = ((\mu * n) / R') \text{Angular}[	heta, \Delta m_{j1}, \Delta m_{j2}];$
    If $\phi[i, k] = dneg \lor \phi[i, k] = dpos,$
        If $\phi[j, k] = ppos \lor \phi[j, k] = pneg,$
            If $\phi[i, k + 1] = dneg \lor \phi[i, k + 1] = dpos,$
                If $\phi[j, k + 1] = mpos \lor \phi[j, k + 1] = mneg,$
                    $H[i, j] = u;$
                    $H[j, i] = u;$
                $]; ]; ];$
            $]; ]; ];$
        $]; ]; ];$
    If $\phi[i, k] = dneg \lor \phi[i, k] = dpos,$
        If $\phi[j, k] = mpos \lor \phi[j, k] = mneg,$
            If $\phi[i, k + 1] = dneg \lor \phi[i, k + 1] = dpos,$
                If $\phi[j, k + 1] = ppos \lor \phi[j, k + 1] = pneg,$
                    $H[i, j] = u;$
                    $H[j, i] = u;$
                $]; ]; ];$
        $]; ]; ];$
    If $\phi[i, k] = ppos \lor \phi[i, k] = pneg,$
        If $\phi[j, k] = dpos \lor \phi[j, k] = dneg,$
            If $\phi[i, k + 1] = mneg \lor \phi[i, k + 1] = mpos,$
                If $\phi[j, k + 1] = dpos \lor \phi[j, k + 1] = dneg,$
                    $H[i, j] = u;$
                    $H[j, i] = u;$
                $]; ]; ];$
        $]; ]; ];$
    If $\phi[i, k] = mneg \lor \phi[i, k] = mpos,$
        If $\phi[j, k] = dpos \lor \phi[j, k] = dneg,$
            If $\phi[i, k + 1] = pneg \lor \phi[i, k + 1] = ppos,$
                If $\phi[j, k + 1] = dpos \lor \phi[j, k + 1] = dneg,$
                    $H[i, j] = u;$
                    $H[j, i] = u;$
                $]; ]; ];$
        $]; ]; ];$
dataPlot = {}; 
\( h = 1; \)
\( \delta = 10; \)

(* Here we create the eigenvectors and eigenvalues of H, 
then create S and S dagger to create the time evolution U. The for loop for the 
matrix Y allows us to pass the eigenvalues of a certain value through U. *)
evec = Normalize /@ Eigenvectors[N[H]]; 
eval = Eigenvalues[N[H]]; 
S = Transpose[evec]; 
Sdag = ConjugateTranspose[S]; 
Y = IdentityMatrix[12]; 
For[w = 1, w \leq 12, w++, 
    Y[[w, w]] = eval[[w]]; 
]; 
U[t_] := S.MatrixExp[\(i \cdot t \cdot Y/h\)].Sdag; 
initState = {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}; 
probSum = 0; 
count = 0; 
For[t = 0, t \leq 10, t = t + .01, 
    currentState = U[t].initState 
    probS = Sum[currentState[[qq]]*Conjugate[currentState[[qq]]], {qq, 1, 4}]; 
    AppendTo[dataPlot, {t, probS}]; 
];

(* The plot below is a variation of the rabi 
oscillation of different frequencies using the data from H*)
plot1 = ListPlot[dataPlot, Joined -> True, PlotRange -> All];
\begin{align*}
\text{plot2} &= \text{ListPlot}\{\text{dataPlot}, \text{Joined} \to \text{True}, \text{PlotRange} \to \text{All}\} (\pi/4) \\
\text{plot3} &= \text{ListPlot}\{\text{dataPlot}, \text{Joined} \to \text{True}, \text{PlotRange} \to \text{All}\} (\pi/2) \\
\text{plot4} &= \text{ListPlot}\{\text{dataPlot}, \text{Joined} \to \text{True}, \text{PlotRange} \to \text{All}\} (5\pi/4)
\end{align*}
plot5 = ListPlot[dataPlot, Joined -> True, PlotRange -> All] (*π*)

plot2 = ListPlot[dataPlot, Joined -> True, PlotRange -> All] (*3π/2*)

(*Below are a list of the frequencies calculated using the above code at different angles that allows us to test the angular dependency.*)
\[ \text{freq}_0 = \frac{7}{(7.5 \times 10^{-7})}; \]
\[ \text{freq}_{\pi 12} = \frac{6}{(7.5 \times 10^{-7})}; \]
\[ \text{freq}_{\pi 6} = \frac{10}{(1.8 \times 10^{-6})}; \]
\[ \text{freq}_{\pi 4} = \frac{10}{(4.5 \times 10^{-6})}; \]
\[ \text{freq}_{\pi 3} = \frac{8}{(7.25 \times 10^{-6})}; \]
\[ \text{freq}_{\pi 2} = \frac{8}{(7.2 \times 10^{-6})}; \]
\[ \text{freq}_{\pi 4} = \frac{8}{(3.6 \times 10^{-6})}; \]
\[ \text{freq}_{\pi 3} = \frac{10}{(1.8 \times 10^{-6})}; \]
\[ \text{freq}_{11\pi 12} = \frac{7}{(8.75 \times 10^{-7})}; \]
\[ \text{freq}_{\pi 12} = \frac{8}{(9 \times 10^{-7})}; \]

\[ \text{freqs} = \{\text{freq}_0, \text{freq}_{\pi 12}, \text{freq}_{\pi 6}, \text{freq}_{\pi 4}, \text{freq}_{\pi 3}, \text{freq}_{5\pi 12}, \text{freq}_{\pi 2}, \text{freq}_{7\pi 12}, \text{freq}_{2\pi 3}, \text{freq}_{3\pi 4}, \text{freq}_{5\pi 6}, \text{freq}_{11\pi 12}, \text{freq}_{\pi}\}; \]

freqplot = {};
angle = \pi;
For[\[a = \text{Length}[\text{freqs}], a > 0, a = a - 1, \]
   AppendTo[freqplot, \{-angle, freqs[[a, 1]]\}];
   angle = angle - \pi/12;]

angle = 0;
For[\[b = 1, b \leq \text{Length}[\text{freqs}], b = b + 1, \]
   AppendTo[freqplot, \{angle, freqs[[b, 1]]\}];
   angle = angle + \pi/12;]

(* These plots help us understand the strength of the 
energy exchange at different angles. As you can see on the 
second graph it is a complete 360 degrees and shows us the 
strongest and weakest angles on interaction of the dipoles. *)
plotFreq = ListPlot[freqplot, Joined -> True, PlotRange -> All, PlotStyle -> Red]
polarPlot = ListPolarPlot[freqplot, Joined -> True, PlotRange -> All, PlotStyle -> Purple]
Comparison of X, Y, and Z Orientation of the energy exchange.

The main goal of this notebook is to compare two files and how they exchange energy. This is done by running two simulations projected from two different orientations then comparing the data through the time evolution.

```plaintext
simulationName = "Cylinder705X";
data1 = {};
SetDirectory["/home/japaul/Research/simulationData/" <> simulationName <> "];
(* loop over all lengths for both data 1 and 2 *)
For[i = 0, i <= Length[FileNames[]] - 1, i++,
    filename = simulationName <> "_" <> ToString[i] <> ".txt";
    istrm = OpenRead[filename ];
    a1 = Read[istrm ];
    Close[istrm ];
    data1 = Append[data1, a1];
];
simulationName2 = "Cylinder705Z";
data2 = {};
SetDirectory["/home/japaul/Research/simulationData/" <> simulationName2 <> "];
For[i = 0, i <= Length[FileNames[]] - 1, i++,
    filename = simulationName2 <> "_" <> ToString[i] <> ".txt";
    istrm = OpenRead[filename ];
    a2 = Read[istrm ];
    Close[istrm ];
    data2 = Append[data2, a2];
];

(* To achieve the data we need we use the following *)
(* loop to iterate through both sets of data achieved *)
(* from the files compiled through the super computer *)
Length[data2[[5, 1, 3]]]
```
(*data=();
For[j=1,j≤12,j++,
dontCare=();
For[i=1,i≤2000,i++,
  AppendTo[dontCare,data1[[j,1,3,i]]-data2[[j,1,3,i]]];
  AppendTo[data,dontCare];
];*)

data = {};
For[i=1, i≤1001, i++,
  AppendTo[data, data1[[1, 1, 3, i]]-data2[[1, 1, 3, i]]];
];

(* The list density plot can be set with data[[]] to the
  time evolution we wish to see, as the number increases we
  can see how the energy exchanges differently through time. *)

DensColor[z_] := RGBColor[If[z < 0, -z, 0], (z + Abs[z]) / 2]
DensColor[z_] :=
  RGBColor[If[z < 0, (-z)^2, If[z == 0, 1, 0]], If[z == 0, 1, 0], If[z > 0, z^2, If[z == 0, 1, 0]]]
myblend = (Blend[{{1, Red}, {0.6, Orange}, {0.3, Yellow}, {0, Black}},
  {-0.3, Green}, {-0.6, Blue}, {-1, Purple}]] &;
Graphics[Table[{{myblend [x], Disk[{8 (x + 1), 0}]}, {x, -1, 1, 1/8}]]]

myblend = (Blend[{{1, Red}, {0.1, Yellow}, {0, Black}, {-0.1, Green}, {-1, Blue}},
  If[# < 0, -(#)^2, #]] &);
Graphics[Table[{{myblend [x], Rectangle[{5 (x + 1), 0}]}, {x, -1, 1, 1/100}]]]

(*BarLegend[{{ColorFunction->myblend ,(-1,1)}}]*)

ListDensityPlot[data[[1]], PlotRange→All,
  ColorFunction→myblend , ColorFunctionScaling→False, AspectRatio→1/6]
movie = ListAnimate[
    Table[ListDensityPlot[data[[t]], PlotRange -> All, ColorFunction -> myblend, 
        ColorFunctionScaling -> False, AspectRatio -> 1/6], {t, 1, 1001, 1}]
]

(*This small bit of code takes the above animation and exports it as a .gif file 
that allows us to use it in presentations and for various other things.*)
CylinderEvo3 = Table[ListDensityPlot[data[[t]], PlotRange -> All, ColorFunction -> 
                      myblend, ColorFunctionScaling -> False, AspectRatio -> 1/6], {t, 1, 1001, 1}];
Export["/home/japaul/Documents/CylinderEvo3.gif", CylinderEvo3]

(*The following two plots are just the plots of data1 and data2. Here you 
can see the difficulty of seeing a change in the energy dispersion so we 
want to look at the plots above to see the difference in energy exchange.*)
ListDensityPlot[data1[[3, 1, 3, 2000]]/Max[data1[[1, 1, 2, 1]]], PlotRange -> All, 
        ColorFunction -> DensColor, ColorFunctionScaling -> False, AspectRatio -> 1/6]

ListDensityPlot[data2[[1, 1, 3, 1000]]/Max[data2[[1, 1, 2, 1]]], PlotRange -> All, 
        ColorFunction -> DensColor, ColorFunctionScaling -> False, AspectRatio -> 1/6]
(*This small amount of code is used to show
the exponentially increasing size of or matrices,
and thus demonstrate our need for the super computer. n is the
total number of atoms and ns is the number of s atoms n -
ns is the number of p atoms in the simulation. *)
Size[n_, ns_] := Binomial[n, ns] * 2^n
Table[Size[t, t - 3], {t, 1, 100, 5}]
{0, 1280, 337920, 36700160, 2789212160, 174483046400, 9652938997760,
90657063895040, 23441587904184320, 1068197536616939520, 46893731119995289600,
1997436506731362385920, 82987289901600845332480, 3376492035251796327792640,
134953428164207286094397440, 5311717819931776936365260800,
206291101859040322371861872640, 791817397627911358332332523317760,
300782291345168845392240212705280, 11320119860038088555365559624007680}
(*The following code is used to create three cylinders at different sizes, angles, and orientations to model our series of lasers in the experiment. Where the two smaller cylinders intersect is a overlapping volume full of p atoms, where the larger cylinder intersects that area are the remaining s atoms. In this picture one cylinder is much larger than the one it is overlapping, this was done to show there are two different lasers, in reality the lasers are the same size.*)

Plot1 = Show[Graphics3D[{Opacity[.3], Red, Cylinder[{{0, 0, 0}, {1, 1, 1}}, $\frac{1}{16}$]}, Axes -> False, Boxed -> False];
Plot2 = Show[Graphics3D[{Opacity[.3], Blue, Cylinder[{{0, 0, 0}, {0, 0, 1}}, $\frac{1}{16}$]}, Axes -> False, Boxed -> False];
Plot3 = Show[Graphics3D[{Opacity[.3], Red, Cylinder[{{0, 0, 0}, {1, 1, 1}}, $\frac{1}{8}$]}, Axes -> False, Boxed -> False];(*776*)
Show[Plot1, Plot2, Plot3]