Simulations of the Angular Dependence of the Dipole-Dipole Interaction

Matan Peleg
Ursinus College, mapeleg@ursinus.edu

Jacob T. Paul
Ursinus College

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

Part of the Atomic, Molecular and Optical Physics Commons, and the Quantum Physics Commons

Click here to let us know how access to this document benefits you.

Recommended Citation
https://digitalcommons.ursinus.edu/physics_astro_sum/1

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.
Final Project by Matan Peleg for summer fellows 2015:
I worked primarily on constructing the geometries and
testing and writing code for the final simulations. The
code for a cylinder of s atoms with a ball of p atoms
is in my source code. The directory for the source code
is: /home/mapeleg/source/ssppAngle/ssppAngle.c. I also
incorporated the use of the struct for use in the header. Below
is the final copy of my angularAnalysis notebook. *

Plotting our data for the anisotropy of the dipole-dipole interaction

Color maps for intensity plots -- just execute this cell to get some color scales for your plots

(* just the colors themselves *)
colorBlend[u_] := Blend[{{0, RGBColor[0, 0, 9/16]}, {1/9, Blue}, {23/63, Cyan},
  {13/21, Yellow}, {47/63, Orange}, {55/63, Red}, {1, RGBColor[1/2, 0, 0]}}, u];
Graphics[Raster[{Range[100]/100}, ColorFunction -> colorBlend[H &]],
  AspectRatio -> .1]

(* with power law scaling *)
cMap2[u_] := colorBlend[u];
colorMap[u_] := colorBlend[u^0.4];
Graphics[
  Raster[{Range[100]/100}, ColorFunction -> colorMap[H &]], AspectRatio -> .1]

cm = {0, 12, 27, 40, 50, 55, 63}/63;
colorMapDiss[u_?NumericQ] :=
  Blend[{{cm[[1]], RGBColor[0, 0, 0.55]}, {cm[[2]], Blue},
    {cm[[3]], Cyan}, {cm[[4]], Yellow}, {cm[[5]], Orange},
    {cm[[6]], Red}, {cm[[7]], RGBColor[0.5, 0, 0]}}, u] /; 0 <= u <= 1;
colorMap[u_] := colorMapDiss[u^0.4];
Graphics[Raster[{Range[100]/100}, ColorFunction -> colorMap[H &]], AspectRatio -> .1]

Load the Data

This cell will load your data files. Change simulationName to be the name you gave your simulation in
the config file. Change the directory to the place on your laptop where you store your data. After you
enter the cell, your data will be stored in a list of lists cleverly named data.

```
in[8] = simulationName = "xNSFTest_txx";
dataX = {};
SetDirectory["/home/mapeleg/research/angular/simulationData/"] <> simulationName <> "/";
For[i = 0, i ≤ Length[FileNames[]] - 1, i++,
    filename = simulationName <> "_<ToString[i]<>".txt";
    istrm = OpenRead[filename ];
    a = Read[istrm ];
    Close[istrm ];
    dataX = Append[dataX, a];
];
simulationName = "zNSFTest_txx";
dataZ = {};
SetDirectory["/home/mapeleg/research/angular/simulationData/"] <> simulationName <> "/";
For[i = 0, i ≤ Length[FileNames[]] - 1, i++,
    filename = simulationName <> "_<ToString[i]<>".txt";
    istrm = OpenRead[filename ];
    a = Read[istrm ];
    Close[istrm ];
    dataZ = Append[dataZ, a];
];
in[18] = (*This cell creates the difference data and linearly 
    extrapolates it to better demonstrate the angular dependence.*)
data = {};
For[i = 1, i ≤ Length[dataX[[1, 1, 2]]] - 1, i++,
    d1 = dataX[[1, 1, 3, i]] - dataZ[[1, 1, 3, i]]; 
    d2 = dataX[[1, 1, 3, i+1]] - dataZ[[1, 1, 3, i+1]]; 
    del = d2 - d1;
    For[j = 0, j ≤ 4, j++,
        AppendTo[data, d1 + (0.2 * del * j)]; 
    ];
];
```
In[1]:= DensColor[z_] := RGBColor[If[z < 0, -z, 0], (z + Abs[z])/2]
DensColor[z_] := RGBColor[If[z < 0, (-z)^(1/2), If[z == 0, 1, 0]],
  If[z == 0, 1, 0], If[z > 0, z^(1/2), If[z == 0, 1, 0]]]
myblend = Blend[{{1, Red}, {.6, Orange}, {.3, Yellow}, {0, Black}, {-0.3, Green},
  {-0.6, Blue}, {-1, Purple}}, If[# < 0, -(-#)^(1/2), #^(1/2)] &];
Graphics[Table[{myblend[x], Disk[({8 (x+1)}, 0)]}, {x, -1, 1/8}]]

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

Out[4]=
Out[6]=

Plot the data

Exercise 1: Plot total probability of initial s atoms being in the p state vs. time

The part of the data that we’re really interested in is stored in data[[1,1,2]]. This is an array of 1 μm resolution (we can change that) pixels. In each pixel we store the total p state probability at that time. Note what your maximum time and time resolution were from your simulation; this tells you how many items should be in that list. You should compare your calculated length to the actual length and make sure it makes sense to you:

Length[dataX[[1, 1, 2]]]

251

A particular time t in your data corresponds to some item n in your list depending on the total time simulated and resolution. The results for the 9th time you simulated would be stored in data[[1,1,2,9]].

Now let’s plot some of this data. The cell below uses ListDensityPlot to create a heat map of p state probability. Redder means more p state character, bluer means less p state character. Since you start at \( t = 0 \) with p state character only in the center, your plot at 0 μs should be some red at the center with dark blue everywhere else.

Try executing the following cell for different times by changing the fourth index in the first occurrence of data. Note that we divide by the maximum value of data[[1,1,2,1]] to normalize all the data the same way. (Why choose that time?)
PlotX = ListDensityPlot[dataX[[1, 1, 2, 25]]/Max[dataX[[1, 1, 2, 1]]],
PlotRange -> All, ColorFunction -> colorMap,
ColorFunctionScaling -> False, AspectRatio -> 7/30, ImageSize -> Large]

PlotZ = ListDensityPlot[dataZ[[1, 1, 2, 25]]/Max[dataZ[[1, 1, 2, 1]]],
PlotRange -> All, ColorFunction -> colorMap,
ColorFunctionScaling -> False, AspectRatio -> 7/30, ImageSize -> Large]

PlotDel = ListDensityPlot[data[[25]], PlotRange -> All,
ColorFunctionScaling -> False, AspectRatio -> 7/30, ImageSize -> Large]

We can also make a movie of the data to watch the energy exchange happen. This might not be interesting if the evolution of your energy exchange is too rapid. You could try a lower density or smaller dipole moment to slow things down so you can see some evolution. You can figure out what all of the parameters to this command mean!
ListAnimate[Table[ListDensityPlot[dataX[[1, 1, 2, i]]/Max[dataX[[1, 1, 2, 1]]],
    PlotRange → All, ColorFunction → colorMap, ColorFunctionScaling → False,
    AspectRatio → 12/60], {i, 1, Length[dataX[[1, 1, 2]]], 1}]]

ListAnimate[Table[ListDensityPlot[dataZ[[1, 1, 2, i]]/Max[dataZ[[1, 1, 2, 1]]],
    PlotRange → All, ColorFunction → colorMap, ColorFunctionScaling → False,
    AspectRatio → 12/60], {i, 1, Length[dataZ[[1, 1, 2]]], 1}]]

In[20] := ListDensityPlot[data[[10]], PlotRange → All,
    ColorFunction → myblend, ColorFunctionScaling → False, AspectRatio → 1/6]

In[21] := movie = ListAnimate[
    Table[ListDensityPlot[data[[t]], PlotRange → All, ColorFunction → myblend,
        ColorFunctionScaling → False, AspectRatio → 1/6], {t, 1, 350, 1}]]
CylinderEvo4 = 
Table[ListDensityPlot[data[[t]], PlotRange -> All, ColorFunction -> myblend, 
ColorFunctionScaling -> False, AspectRatio -> 1/6], {t, 1, 300, 1}];
Export["~/home /mapeleg /Documents /CylinderEvo4.gif", CylinderEvo4]