7-24-2015

Toward Analog Quantum Computing: Simulating Designer Atomic Systems

Jacob L. Bigelow
Ursinus College, jabigelow@ursinus.edu

Veronica L. Sanford
Ursinus College, vesanford@ursinus.edu

Jacob Bigelow

Throughout the summer we spent three days a week at Ursinus college running simmulations on the super computer and two days a week we would travel to Bryn Mawr college to run experiments in their lab using their lasere apperatus. We were looking at the dipole dipole interactions of atoms in either a s or p energy state. We would write code for simulations in the language C to test different arrangements of the these atoms and look at the energy exchange that took place. We had to learn how be proffecient in C, as well as using the program *Mathematica*, and using the super computer here at Ursinus to run the simulations we desired. We also were required to analytically solve integrals in order to understand the energy exchange taking place.

For our final product we presented our results from the simulations we ran at Summer Fellows 17th Annual Symposium and turning in the *Mathematica* notebooks we used to analyze our results. The notebooks were used to study the interactions resulting from the simulations we ran on the super computer. Other notebooks were made in order to make sure our simulations would run the way we expected and needed them to. For things like making sure our atoms in the simulations were being uniformly randomly distributed, but mostly the notebooks were used to compare and contrast different arrangements of groups of p and s atoms. We were looking for any interesting effects the geomotries of these spheres might have on the energy exchange.

* Made changes ssppAngle.c to add geometries 4 and 5 which can be found in /home/jabigelow/summerFellows/multivac/source/angular

* changed header data in ssppAngle.c in order to make it work with Dr. Carrol's new version of the compilation code

* then took angular dependence out of ssppAngle.c and make it into spNoAngle.c which can be found in /home/jabigelow/sumerFellows/source/spNoAngle

* made new geometries 6 and 7 for spNoAngle.c

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

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

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

Out[2]=

Out[5]=

Out[9]=

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[10]:=
```
simulationName = "in4Middle";
data = {};
SetDirectory[
    "/home /jabigelow/summerFellows /simulationData /" <> simulationName <> "/"];
For[i = 0, i ≤ Length[FileNames []] - 1, i++,
    filename = simulationName <> "_" <> ToString[i] <> ".txt";
    istrm = OpenRead[filename ];
    a = Read[istrm ];
    Close[istrm ];
    data = Append[data, a];
  ];
```

In[14]:=
```
simulationName = "line355";
data2 = {};
SetDirectory[
    "/home /jabigelow/summerFellows /simulationData /" <> simulationName <> "/"];
For[i = 0, i ≤ Length[FileNames []] - 1, i++,
    filename = simulationName <> "_" <> ToString[i] <> ".txt";
    istrm = OpenRead[filename ];
    a = Read[istrm ];
    Close[istrm ];
    data2 = Append[data2, a];
  ];
```

In[18]:=
```
simulationName = "separated433";
data3 = {};
SetDirectory[
    "/home /jabigelow/summerFellows /simulationData /" <> simulationName <> "/"];
For[i = 0, i ≤ Length[FileNames []] - 1, i++,
    filename = simulationName <> "_" <> ToString[i] <> ".txt";
    istrm = OpenRead[filename ];
    a = Read[istrm ];
    Close[istrm ];
    data3 = Append[data3, a];
  ];
```

Get familiar with the data

For now, let's assume you only vary one parameter in your simulation runs.

First, enter the following cell to see the length of each of the dimensions of your data. If you have have only varied one parameter, you should get two numbers here: the
first is the number of different instances of the varied parameter that you simulated. For example, if you ran the simulation for radii from 10 - 15 μm in steps of 1 μm, you would have a 6 here. The second number is the length of each of these sub-lists of data. This should always say 2: the first element is the

actual data, and the second element is the list of parameters for that simulation.

```
Dimensions [data3]
```

```
{71, 2}
```

Now enter the following cell, where we access the parameters for the first instance of the parameter that you varied (if you didn't vary any parameters, this will be the only instance!). By looking at your configuration file, you should be able to figure out what these are. This is one change you'll make to Monocle -- it would be a good deal easier if these were labeled!

```
data[[3, 2]]
```

```
{{12, 8, 10, 0.05, 3., 5}, {10, 20, 40, 0, 0, 0, 0}, {708}, {0, 0, 0, 0, 2}}
```

```
data[[4, 2, 2, 2]]
```

```
10
```

```
{{7, 6, 2, 0.001`, 3.`, 3}, {45, 5, 0, 0, 0, 0, 0}, {100}, {0, 0, 0, 2, 2}}
```

```
{{7, 6, 2, 0.001, 3., 3}, {45, 5, 0, 0, 0, 0, 0}, {100}, {0, 0, 0, 2, 2}}
```

This particular data contrains *lots* of stuff. The first element in the actual data for your first parameter instance is *data[[1,1,1]]* -- the first "1" says access the first parameter instance, the second "1" says access the data (not the parameter list), and the third "1" says get the first element of the data.

Here, we have stored a list of ordered triplets: {time in μs, total probability of initial *s* atoms being in the *p* state, total probability of initial *p* atoms being in the *p* state}

This is not what we're truly interested in, but it's a nice quick measure of what's going on. Enter the following cell to see what that data looks like.

In[22]:=
```
data[[1, 1, 1]];
```

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[data[[1, 1, 2]]]
```

```
201
```

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

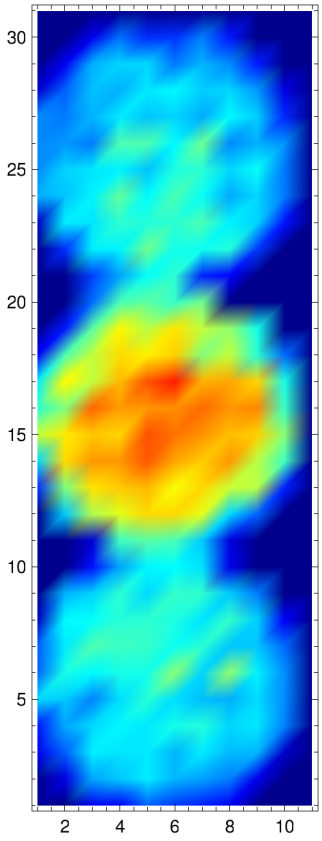
at $t = 0$ with p state character only in the center, your plot at 0 μs should be some red at the center with dark blue everywhere else.

This is the main way we analyzed our data looking at the different amounts of energy exchange between the different groups in the different arrangments. We would also look to see if changing the amount of atoms in each group effected the dipole-dipole interaction.

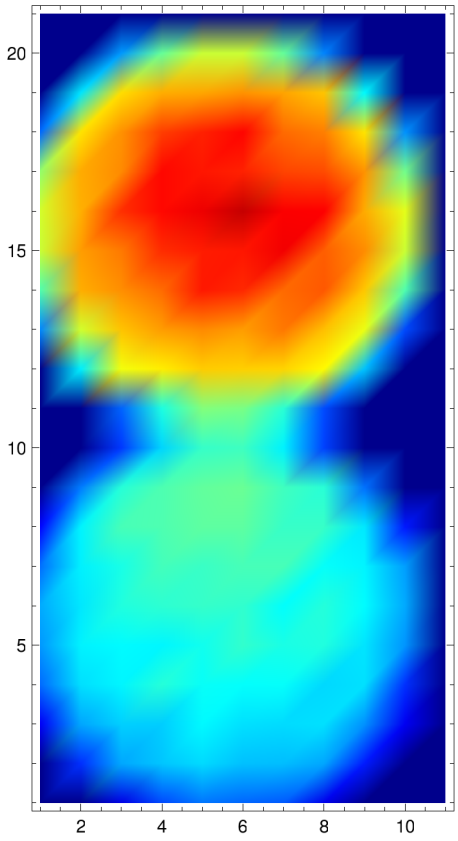
```
data[[1, 2]]
```

```
{{12, 8, 10, 0.05, 3., 5}, {10, 0, 40, 0, 0, 0, 0}, {708}, {0, 0, 0, 0, 2}}
```

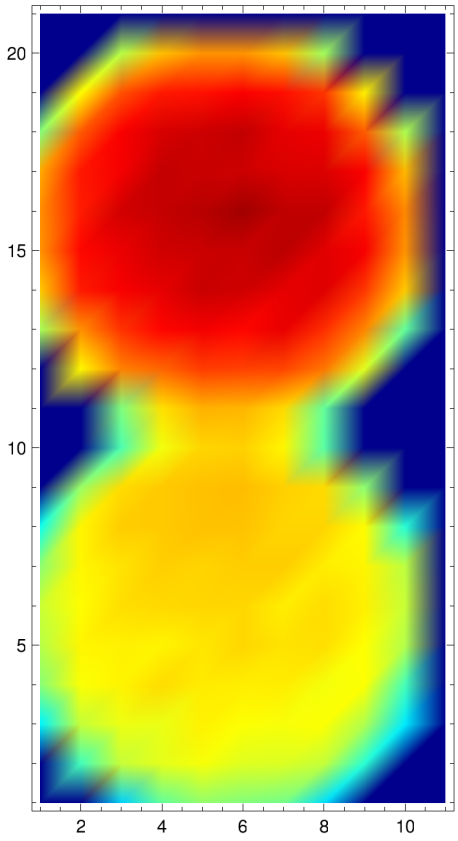
```
dataIndex = 1;
time = 200;
ListDensityPlot[data[[dataIndex, 1, 2, time ]]/Max[data[[dataIndex, 1, 2, 1]]],
  PlotRange → All, ColorFunction→ colorMap, ColorFunctionScaling→ False,
  AspectRatio→ Length[data[[dataIndex, 1, 2, time ]]]/
      Length[data[[dataIndex, 1, 2, time , 1]]], ImageSize → Medium ]
```



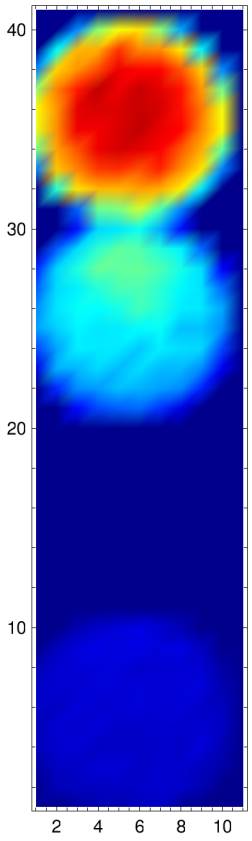
```
ListDensityPlot[dataSum [[dataIndex, 1, 2, time ]]/Max[dataSum [[dataIndex, 1, 2, 1]]],
  PlotRange→All, ColorFunction→colorMap, ColorFunctionScaling→False,
  AspectRatio→Length[dataSum [[dataIndex, 1, 2, time ]]]/
      Length[dataSum [[dataIndex, 1, 2, time , 1]]], ImageSize →Medium ]
```



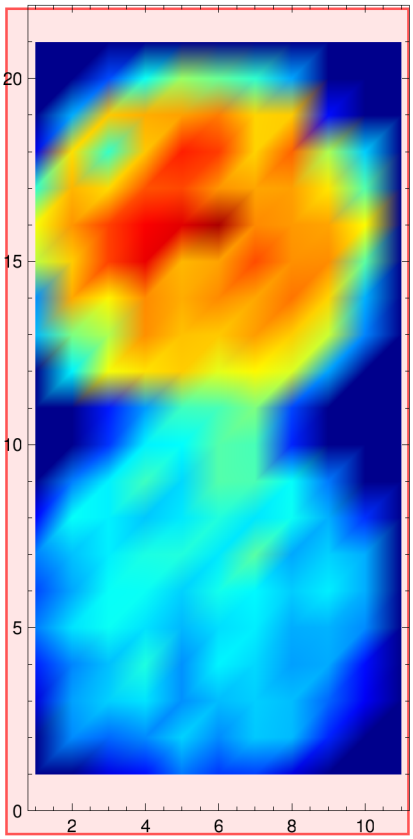
```
ListDensityPlot[dataSum [[dataIndex, 1, 2, time ]]/Max[dataSum [[dataIndex, 1, 2, 1]]],
  PlotRange→All, ColorFunction→colorMap, ColorFunctionScaling→False,
  AspectRatio→Length[dataSum [[dataIndex, 1, 2, time ]]]/
      Length[dataSum [[dataIndex, 1, 2, time , 1]]], ImageSize →Medium ]
```



```
ListDensityPlot[dataSum [[dataIndex, 1, 2, time ]]/Max[dataSum [[dataIndex, 1, 2, 1]]],
  PlotRange→All, ColorFunction→colorMap, ColorFunctionScaling→False,
  AspectRatio→Length[dataSum [[dataIndex, 1, 2, time ]]]/
      Length[dataSum [[dataIndex, 1, 2, time , 1]]], ImageSize →Medium ]
```

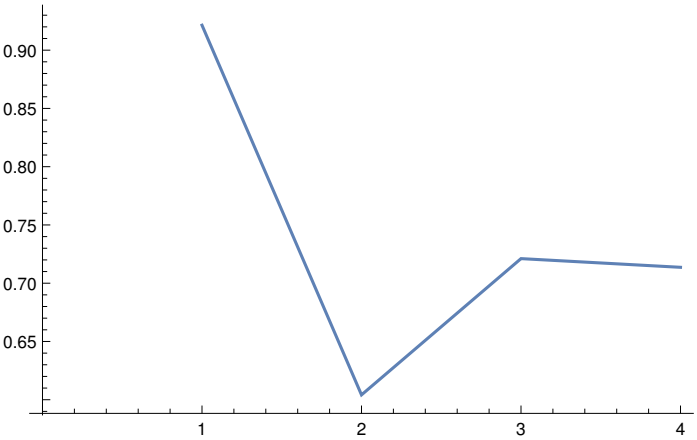


```
ListDensityPlot[data3[[dataIndex, 1, 2, time ]]/Max[data3[[dataIndex, 1, 2, 1]]],
  PlotRange→All, ColorFunction→colorMap,
  ColorFunctionScaling→False, AspectRatio→ar3, ImageSize →Medium ]
```

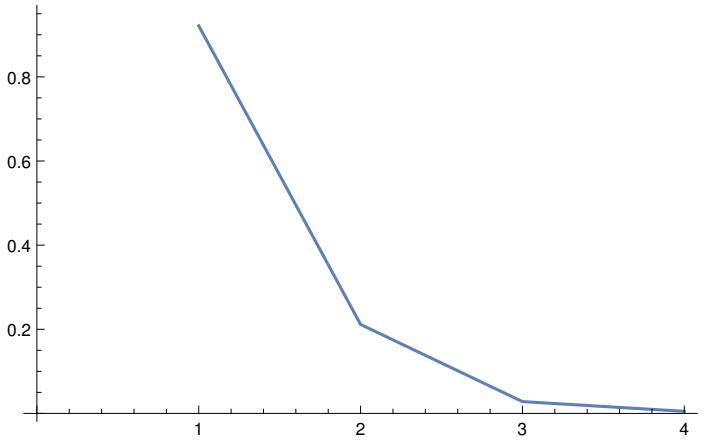


These graph below looks at the sum of the p character for the s group moving away and plots the sum for a particular time as sphere moves away and the graph below this one is plotting the sum of p character in the s group that remains stationary as the other s group is moving away.

```
mm   =
    Table[Sum [dataSum [[t, 1, 2, 150, i, j]], {i, (t*10)-9, t*10, 1}, {j, 1, 10, 1}]/500,
      {t, 1, Length[dataSum ], 1}];
ListPlot[mm  , Joined→True]
```



```
nn = Table[Sum [dataSum [[t, 1, 2, 150, i, j]], {i, 1, 10, 1}, {j, 1, 10, 1}]/500,
       {t, 1, Length[dataSum ], 1}];
ListPlot[nn, Joined→ True]
```



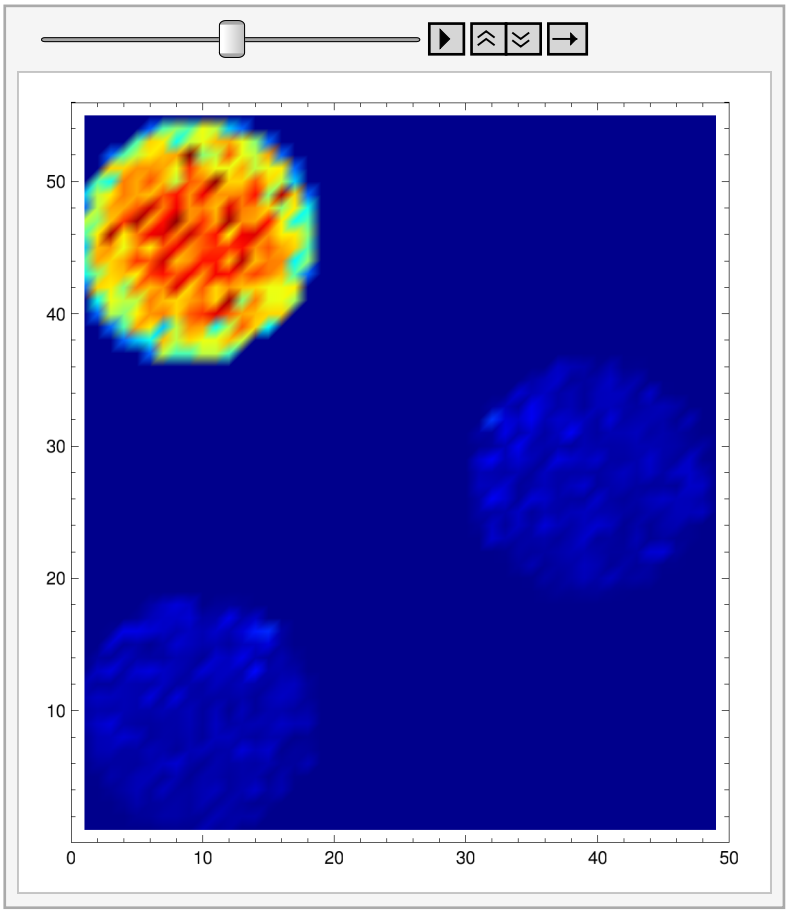
The code below is for the purpose of making movies. We are creating a table of images like you saw above and playing those images as time goes on. The for loop is there to slow down the movie so that you can see spread of p character better, because the spreading can happen so fast sometimes we want to look at more frames than it originally gives us.

```
newData = {};
nStep = 100;
for[t = 1, t ≤ Length[data[[1, 1, 2]]]/5, t = t+1,
    dataA = data[[dataIndex, 1, 2, t]]/Max[data[[dataIndex, 1, 2, 1]]];
    dataB = data[[dataIndex, 1, 2, t+1]]/Max[data[[dataIndex, 1, 2, 1]]];
    diff = (dataA-dataB)/nStep;
    AppendTo[newData, dataA];
    for[s = 1, s ≤ nStep-1, s = s+1,
      AppendTo[newData, dataA+ (s*diff)];
    ];
  ];
```

```
x = Table[ListDensityPlot[newData, PlotRange→ All,
        ColorFunction→ colorMap, ColorFunctionScaling→ False,
        AspectRatio→ Length[newData[[dataIndex, 1, 2, time ]]]/
            Length[newData[[dataIndex, 1, 2, time , 1]]]],
      {j, 1, Length[newData[[1, 1, 2]]], 1}];
```

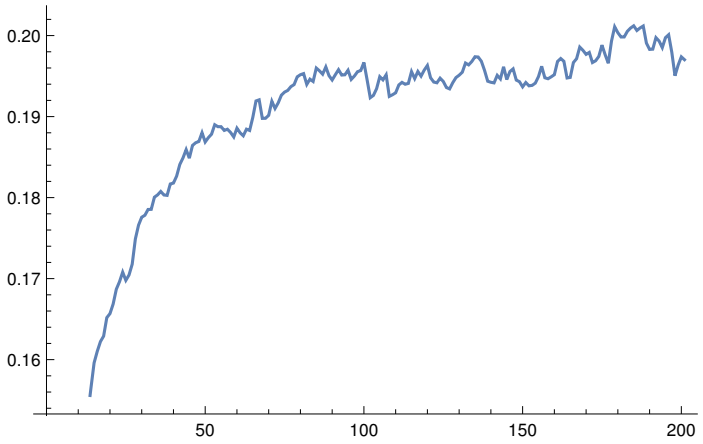
```
Export["/home /jabigelow/summerFellows  /mathematica  /pinMiddle.gif", x];
```

```
ListAnimate [
  Table[ListDensityPlot[data3[[dataIndex, 1, 2, i]]/Max[data3[[10, 1, 2, 1]]],
      PlotRange → All, ColorFunction→ colorMap, ColorFunctionScaling→ False,
      AspectRatio→ ar3], {i, 1, Length[data3[[1, 1, 2]]], 10}]]
```

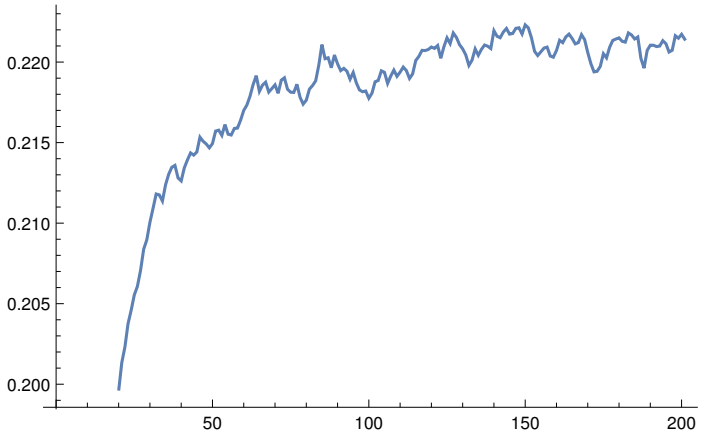


Above is one of the movies we created shown here in *Mathematica.*
Below are graphs of the p character in a particular s sphere and is plotting probablity versus time at one particular distance for the s group.

```
xx = Table[Sum [data[[1, 1, 2, t, i, j]], {i, 1, 10, 1}, {j, 1, 10, 1}]/2000,
      {t, 1, Length[dataSum [[dataIndex, 1, 2]]], 1}];
ListPlot[xx, Joined→True]
```




```
yy = Table[Sum [dataSum [[1, 1, 2, t, i, j]], {i, 1, 10, 1}, {j, 1, 10, 1}]/2000,
      {t, 1, Length[dataSum [[dataIndex, 1, 2]]], 1}];
ListPlot[yy, Joined→True]
```



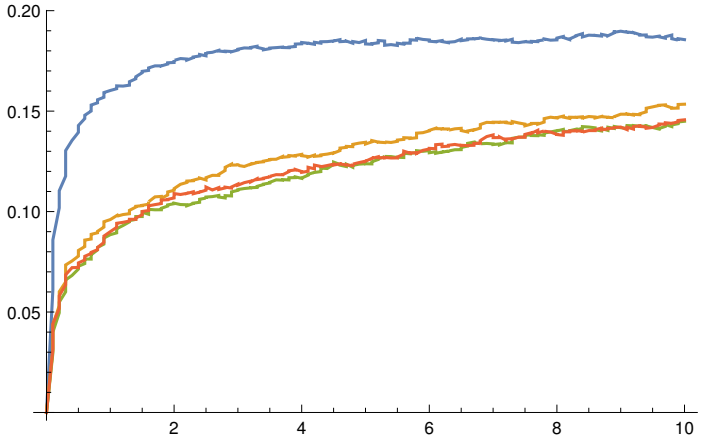

```
(*This matrix  is just here to show the spheres but in the form  of a
    matrix . This is helpful to actually see what the raw data is and can
    be used to help determine  where to find the particular s groups.)
```

```
MatrixForm [data3[[1, 1, 2, 30]]]
```

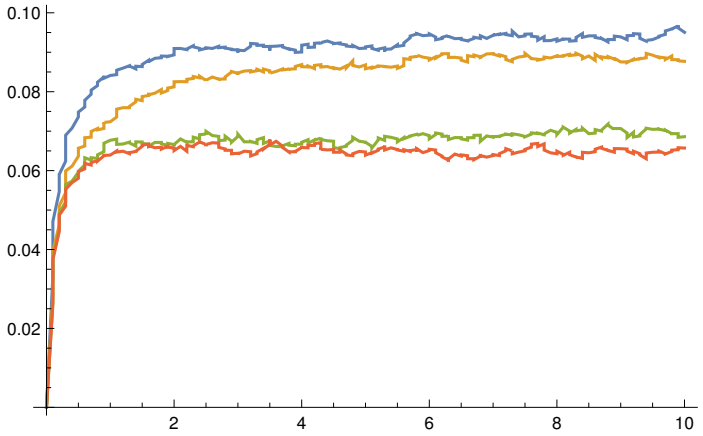
0.	0.	0.531699	0.993434	2.0858	1.55792	1.22694	0.457659	0.	0
0.	2.89261	2.37815	2.95454	5.72698	2.56372	4.68487	4.38629	1.07453	0
0.207387	2.9676	3.75716	5.08774	3.41291	4.04895	4.21129	4.28084	2.66156	0.39
1.19428	3.04759	4.431	7.12861	3.1867	6.17918	6.14816	4.34892	3.61268	0.84
2.90818	5.9414	5.81355	5.42062	3.70271	4.80109	4.09947	3.43461	4.33608	2.4(
1.79588	2.98337	5.92734	6.97535	4.20838	7.4832	6.47538	3.89446	6.13564	3.53
2.86522	4.6808	6.25857	6.04916	7.35362	5.89471	8.43127	4.66671	5.04561	3.38
0.696633	5.75915	5.17971	5.97529	6.15773	7.86624	5.76964	6.74154	3.8119	1.48
0.	2.6415	7.36486	8.24326	5.63153	9.66947	8.24782	5.88745	2.5236	0
0.	0.	0.74253	4.14268	4.47148	8.43068	7.72494	1.05752	0.	0
0.	0.	1.34615	3.32656	8.44476	8.74586	10.5565	1.8795	0.	0
0.	5.58514	16.0604	20.512	21.5321	16.4502	17.2084	14.1749	4.22085	0
4.27706	9.55115	14.0327	28.35	23.6013	21.9918	27.4012	20.3291	14.8901	3.0(
2.90862	24.8075	18.3385	29.6608	24.4061	29.8928	25.0219	30.117	22.7743	4.87
14.3453	22.2733	34.7685	40.7101	22.5419	26.002	31.1374	27.9306	27.6923	9.61
16.7669	28.1902	34.1074	38.1021	40.8172	47.1654	28.4345	27.4743	26.1876	17.(
7.50211	23.708	20.4763	32.7158	33.8322	28.6621	24.932	25.7305	19.0632	9.3
0.62882	20.0852	8.69138	21.5564	36.9249	34.399	21.3115	31.2168	13.4255	4.86
0.	3.71416	22.3031	25.1632	25.6963	29.3716	20.4086	22.1871	1.00925	0
0.	0.	1.60617	6.87743	11.9494	9.21339	8.36878	3.67563	0.	0
0.	0.	0.	0.	0.	0.	0.	0.	0.	0

The graphs below show the p character for the total system as time goes by. Each line is a different distance the second s group is from the other two groups for various simulations.

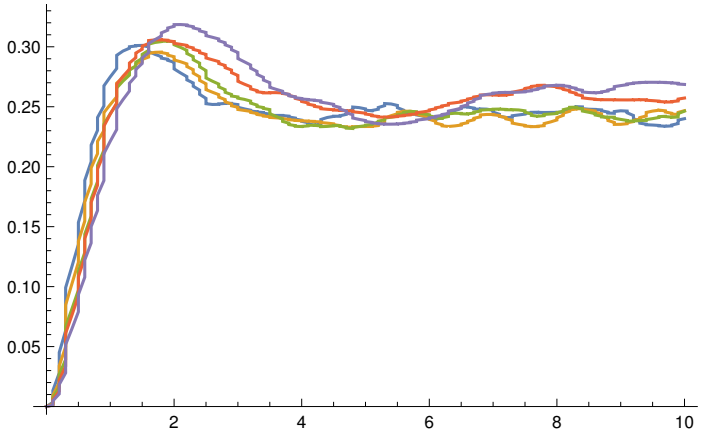
```
ListPlot[Table[{#[[1]], #[[2]]} &/@data[[i, 1, 1]], {i, 1, 4}],
  Joined→ True, PlotRange→ All]
```



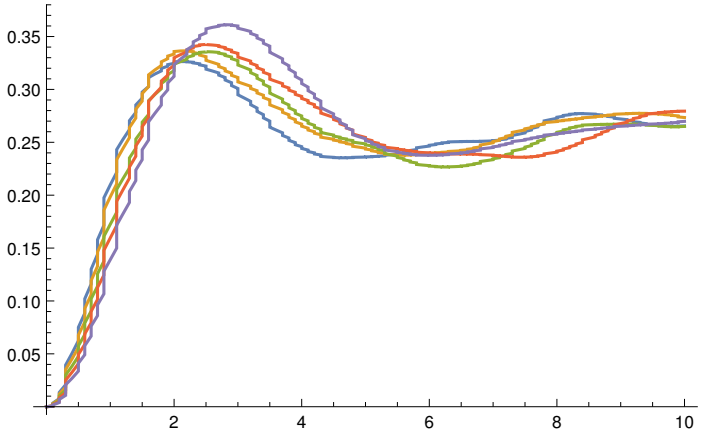
```
ListPlot[Table[{#[[1]], #[[2]]} & /@ data2[[i, 1, 1]], {i, 1, 4}],
  Joined → True, PlotRange → All]
```




```
ListPlot[Table[{#[[1]], #[[2]]} & /@ data3[[i, 1, 1]], {i, 11, 15}],
  Joined → True, PlotRange → All]
```




```
ListPlot[Table[{#[[1]], #[[2]]} & /@ data3[[i, 1, 1]], {i, 16, 20}],
  Joined → True, PlotRange → All]
```



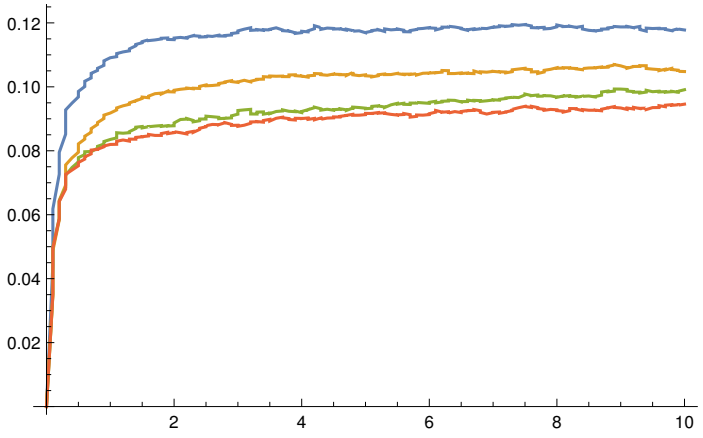
Here we again are just loading more data from different simulations so we can more easily look at the results from the various simulations

```
simulationName  = "spreadSpheres";
data4 = {};
SetDirectory[
    "/home /jabigelow/summerFellows  /simulationData /" <> simulationName  <> "/"];
For[i = 0, i ≤ Length[FileNames []] - 1, i++,
    filename = simulationName  <> "_" <> ToString[i] <> ".txt";
    istrm = OpenRead[filename ];
    a = Read[istrm ];
    Close[istrm ];
    data4 = Append[data4, a];
  ];
simulationName  = "separated344";
data5 = {};
SetDirectory[
    "/home /jabigelow/summerFellows  /simulationData /" <> simulationName  <> "/"];
For[i = 0, i ≤ Length[FileNames []] - 1, i++,
    filename = simulationName  <> "_" <> ToString[i] <> ".txt";
    istrm = OpenRead[filename ];
    a = Read[istrm ];
    Close[istrm ];
    data5 = Append[data5, a];
  ];
simulationName  = "separated544";
data6 = {};
SetDirectory[
    "/home /jabigelow/summerFellows  /simulationData /" <> simulationName  <> "/"];
For[i = 0, i ≤ Length[FileNames []] - 1, i++,
    filename = simulationName  <> "_" <> ToString[i] <> ".txt";
    istrm = OpenRead[filename ];
    a = Read[istrm ];
    Close[istrm ];
    data6 = Append[data6, a];
  ];
simulationName  = "line455";
data7 = {};
SetDirectory[
    "/home /jabigelow/summerFellows  /simulationData /" <> simulationName  <> "/"];
For[i = 0, i ≤ Length[FileNames []] - 1, i++,
    filename = simulationName  <> "_" <> ToString[i] <> ".txt";
    istrm = OpenRead[filename ];
    a = Read[istrm ];
    Close[istrm ];
    data7 = Append[data7, a];
  ];
```

```
simulationName  = "separated533";
data8 = {};
SetDirectory[
    "/home /jabigelow/summerFellows  /simulationData /" <> simulationName  <> "/"];
For[i = 0, i ≤ Length[FileNames []] - 1, i++,
    filename = simulationName  <> "_" <> ToString[i] <> ".txt";
    istrm = OpenRead[filename ];
    a = Read[istrm ];
    Close[istrm ];
    data8 = Append[data8, a];
  ];
```

```
simulationName  = "spreadSpheres3atoms ";
data9 = {};
SetDirectory[
    "/home /jabigelow/summerFellows  /simulationData /" <> simulationName  <> "/"];
For[i = 0, i ≤ Length[FileNames []] - 1, i++,
    filename = simulationName  <> "_" <> ToString[i] <> ".txt";
    istrm = OpenRead[filename ];
    a = Read[istrm ];
    Close[istrm ];
    data9 = Append[data9, a];
  ];
```

```
dataSum  = (data + data2 + data3 + data4 + data5 + data6 + data7 + data8 + data9) / 9;
ListPlot[Table[{#[[1]], #[[2]]} & /@ dataSum  [[i, 1, 1]], {i, 1, 4}],
  Joined → True, PlotRange → All]
```



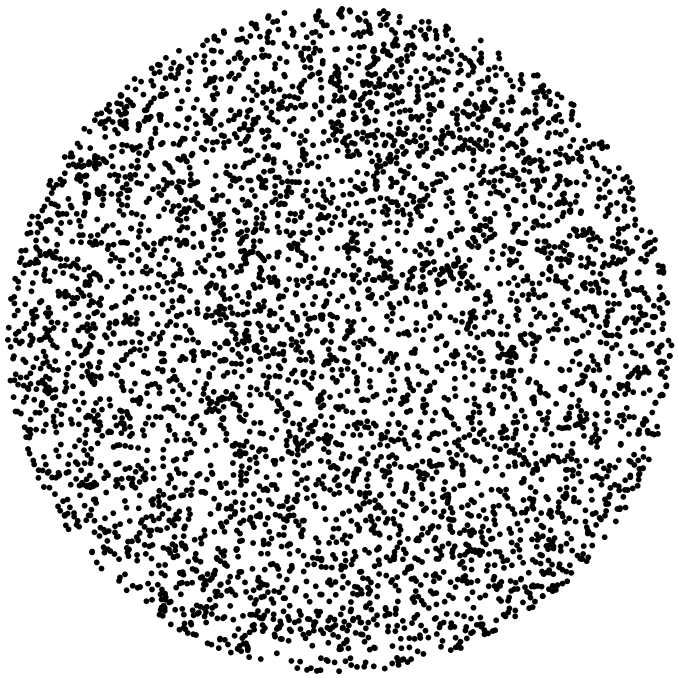
This code was made in order to test to see if we were correctly distributing different atoms among a circle which we then used to make sure we could uniformily randomly distribute atoms among a sphere.

```
file =
    "/home /jabigelow/summerFellows  /multivac /experiment /cppExercise/test.dat";
numbers  = BinaryReadList[file, "Real64"];
x = {};
y = {};
Length[numbers ]
```

```
10000
```

```
graph = {};
size= Length[numbers ];
For [i = 1, i ≤ size, i = i + 2,
    r = numbers [[i]];
    θ = numbers [[i+1]];
    x = r*Cos[θ];
    y = r*Sin[θ];
    AppendTo[graph, Graphics[Point[{x, y}]]];
  ];

Show[graph]
```



```
Show[Graphics[Point[{3, 4}]], Graphics[Point[{2, 7}]]]
```

Also used was the final projcet for our fall 2014 research which we altered in order to look at different effects of a 2 atom system with one p and one s in order to better understand the energy exchange.

This program was written for the final of the fall semester of 2014. Using the matrix H, where δ originally was 0, we composed S using the eigenvectors of H. We then achieved the time operator with matrix multiplication of S eI and Sdag, giving us U[t]. Then we created a loop to graph the increments of t over the evolution.(in 107)

Plot 1 is δ =0.1, plot 2 is δ=5, plot 3 is δ=10

```
deltaPlot = {};
For[δ = -10, δ ≤ 10, δ = δ + 1,
    h = 1;
    u = 1;
    H = ( 0  u
          u  δ );
evec = Normalize /@ Eigenvectors[H];
eval = Eigenvalues[H];
    S = Transpose[evec];
    Sdag = ConjugateTranspose[S];
U[t_] := S. ( e^(-i eval[[1]] t/h)          0
                    0          e^(-i eval[[2]] t/h) ) . Sdag;
    initState = {1, 0};
    probSum  = 0;
    count = 0;
    For[t = 0, t ≤ 10, t = t + .1,
      currentState = U[t].initState;
      probS = currentState[[2]] * Conjugate[currentState[[2]]];
      probSum  += probS ;
      count++;
    ];
    AppendTo[deltaPlot, {δ, probSum  / count}];
  ];
deltaPlot
```

```
{{-10, 0.0189551 + 0. i}, {-9, 0.0236747 + 0. i}, {-8, 0.0289284 + 0. i},
  {-7, 0.0379628 + 0. i}, {-6, 0.0492217 + 0. i}, {-5, 0.0694644 + 0. i},
  {-4, 0.0976727 + 0. i}, {-3, 0.157308 + 0. i}, {-2, 0.250086 + 0. i},
  {-1, 0.406237 + 0. i}, {0, 0.473992 + 0. i}, {1, 0.406237 + 0. i}, {2, 0.250086 + 0. i},
  {3, 0.157308 + 0. i}, {4, 0.0976727 + 0. i}, {5, 0.0694644 + 0. i}, {6, 0.0492217 + 0. i},
  {7, 0.0379628 + 0. i}, {8, 0.0289284 + 0. i}, {9, 0.0236747 + 0. i}, {10, 0.0189551 + 0. i}}
```

```
plot1 = ListPlot[deltaPlot, Joined→True, PlotRange→All]
```