7-24-2015

Toward Quantum Analog Computing: Simulating Designer Atomic Systems

Veronica L. Sanford
Ursinus College, vlsanford@gmail.com

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

This summer, I did research with Tom Carroll as my mentor. My collaborators were Jacob Bigelow, Jacob Paul, Matan Peleg, and Hunter Sessa. 3 days a week we stayed at Ursinus College and ran simulations on the supercomputer, Multivac, on campus. 2 times a week we would travel to Bryn Mawr college and conduct experiments using the laser apparatus there. The majority of our summer consisted of writing code to simulate the dipole-dipole interaction of these highly-excited Rydberg atoms. We started the summer by learning about quantum-mechanical systems and trying to understand some of the math behind how these systems interact (using the Schrodinger equation and solving this for the very simple case of a 2 atom system.) Once we had a basic knowledge of this, we had to learn how to use Perl script, C++, and *Mathematica* to write code.

We began by looking at the angular dependence of a system with only 2 atoms. We analyzed how changing the angle between the atoms or changing the direction of the electric field would affect how the atoms exchange energy from the s state to the p state. We then wrote code to replicate some results that were recently reported in a paper which detailed an experiment also looking at this interaction. The simulations were promising; we found that changing the angle between the atoms had a large effect on the exhange of p and s character. The frequency of the oscillations between this character overtime changed drastically with the changing angle.

We did an exercise to test our understanding of *Mathematica* and C++ which required that we created a circle with atoms uniformly distributed throughout it. This would be important later when we created geometries that required a uniform distribution of atoms.

The next thing that we looked at was creating specific geometries of groups of uniformly distributed atoms using C++. We wanted to make different spatial arrangements of atoms that could possibly be duplicated in the lab (in the future.) We created a geometry code which created 1 sphere of p atoms and 2 spheres of s atoms. We changed the spatial arrangement of these atoms to test different parameters in each arrangement. To increase the number of atoms that we could place in each group, we got rid of the angular aspect of our code, which took some work. This would allow for more computing power to be focused on the atoms instead of the angular dependence of the energy exchange. We first tried aligning the 3 groups of atoms in a triangle with the sphere of p atoms on top. We incremented the distance that the middle sphere was from the other 2 spheres to see how this affected the interaction; we found that the interaction between the p sphere and the bottom s sphere became stronger as the middle p sphere became farther and farther from the other 2 groups. This was exactly the result that we were expecting.

The next arrangement that we tried is putting the groups of atoms in a straight line with the p sphere on top and the 2 s spheres below. We incremented the distance that the bottom s sphere was from the other 2 spheres, again to test how the interaction would change. We found something suprising; when the bottom sphere became 30 microns from the other 2 spheres, there was more interaction than at the 20 micron distance and also than at the 40 micron distance. We aren't exactly sure why this happened.

The last arrangement that we tried is putting the p sphere between the 2 s spheres. Although we are still working on this case, we found that the p character spreads simultaneously to the 2 s spheres, when they are equidistant from the p sphere. As we incremented 1 s sphere away from the other 2 spheres, the interaction changed.

/home/vesanford/research/multivac/spNoAngle

-Makefile --changed to get rid of angular dependence
-sp
-spNoAngle.c --was originally sspp.c, but we changed it so that it would no longer consider the angular dependence.
-spNoAngle.o

/home/vesanford/research/multivac/experiment/spNoAngle/
-config.txt --changed to get rid of the angular dependence
-results folder
-data folder
-exec folder
-compilation and submission files for new arrangement runs --compilation code was changed to get rid of the angular dependence

/home/vesanford/research/multivac/experiment/angular
-contains all of the original source code/configuration files from the original tests.

Below is the code that we used for testing how the angular dependence affected a 2-atom system

```
(*define codes for atomic states *)

dneg = 1;
dpos = 2;
pneg = 3;
ppos = 4;
mneg = 5;
mpos = 6;

(*define lists of states*)

phi = {{dneg, dneg}, {dneg, dpos}, {dpos, dneg}, {dpos, dpos}, {pneg,
  mneg}, {pneg, mpos}, {ppos, mneg}, {ppos, mpos}, {mneg,
  pneg}, {mneg, ppos}, {mpos, pneg}, {mpos, ppos}};

(*define the mj values for the possible states*)

mjArray = {0, 0, 0, 0, 0, 0};
mjArray[[dneg]] = -100;
mjArray[[dpos]] = 3 / 2;
mjArray[[pneg]] = -100;
mjArray[[ppos]] = 1 / 2;
mjArray[[mneg]] = -100;
mjArray[[mpos]] = 5 / 2;
```

```
(* The au through hartree variables are constants or conversion \
factors. The rest are variables that will be used in later \
calculations and loops. H is an array that is originally filled with \
zeroes but will later be replaced with values determined from the \
Hamiltonian. k distinguishes the first atom from the second atom. \
θ is the angle between the atoms.  R is the distance between \
the atoms, and auR is this value in atomic units. E is the energy \
element needed for a later matrix, and auEnergy is this value in \
atomic units. μν is needed to calculate the total energy.*)

au = 2.42 * 10^-17;
e = 1.602 * 10^-19;
a0 = 5.291 * 10^-11;
hConstant = 6.626 * 10^-34; (* J.s *)

C3 = 2.39 * 10^9; (* Hz.μm^3 *)
hartree = 4.36 * 10^-18;
H = ConstantArray[0, {12, 12}];
k = 1;
θ = 0; (* radians *)
R = 9.1; (* μm *)

auR = 9.1 * 10^-6 / a0; (* a0 *)
Energy = hConstant * C3 / (R^3); (* J *)

auEnergy = Energy / hartree; (* hartree *)
μν = auEnergy * (auR^3);



(* This module is being used to determine which integral must be used \
for later calculations depending on the difference between the \
states. If the difference is 0, 1, or 2, a different integral must be \
used. If the difference is none of these, the element in the matrix \
remains zero. *)

Angular[θ_, Δmj1_, Δmj2_] :=
 Module[{},
  If[Δmj1 + Δmj2 == 0,
   Return[(1 - (3 * Cos[θ]^2)) / 2]];
  If[Abs[Δmj1 + Δmj2] == 1,
   Return[(3 * Sin[θ] Cos[θ]) / Sqrt[2]]];
  If[Abs[Δmj1 + Δmj2] == 2,
   Return[(3 * Sin[θ]^2) / 2]];
  Return[0];
  ];


(* The following "if" statements determine if the jumps that the \
```

```
atoms make between energies are possible or not. The possible paths \
are checked within the loops. If the path is a possibility, the \
Hamiltonian is set to "u" which is determined through the previously \
described module and other constants. Δmj1 and \
Δmj2 are the differences in states for the jump that \
each atom takes. *)
For[i = 1, i <= 12, i++,
 For[j = i, j <= 12, j++,

  Δmj1 =
   mjArray[[phi[[j, k]]]] - mjArray[[phi[[i, k]]]];
  Δmj2 =
   mjArray[[phi[[j, k + 1]]]] - mjArray[[phi[[i, k + 1]]]];
  u = ((μv) / (auR)^3) Angular[θ, Δmj1, \
Δmj2];

  If[phi[[i, k]] == dneg || phi[[i, k]] == dpos,
   If[phi[[j, k]] == ppos || phi[[j, k]] == pneg,
    If[phi[[i, k + 1]] == dneg || phi[[i, k + 1]] == dpos,
     If[phi[[j, k + 1]] == mpos || phi[[j, k + 1]] == mneg,
      H[[i, j]] = u;
      H[[j, i]] = u;
      ];
     ];
    ];
   ];
  If[phi[[i, k]] == dneg || phi[[i, k]] == dpos,
   If[phi[[j, k]] == mpos || phi[[j, k]] == mneg,
    If[phi[[i, k + 1]] == dneg || phi[[i, k + 1]] == dpos,
     If[phi[[j, k + 1]] == ppos || phi[[j, k + 1]] == pneg,
      H[[i, j]] = u;
      H[[j, i]] = u;
      ];
     ];
    ];
   ];
  If[phi[[i, k]] == mneg || phi[[i, k]] == mpos,
   If[phi[[j, k]] == dpos || phi[[j, k]] == dneg,
    If[phi[[i, k + 1]] == pneg || phi[[i, k + 1]] == ppos,
     If[phi[[j, k + 1]] == dpos || phi[[j, k + 1]] == dneg,
      H[[i, j]] = u;
      H[[j, i]] = u;
      ];
     ];
    ];
   ];
  If[phi[[i, k]] == pneg || phi[[i, k]] == ppos,
   If[phi[[j, k]] == dpos || phi[[j, k]] == dneg,
    If[phi[[i, k + 1]] == mneg || phi[[i, k + 1]] == mpos,
```

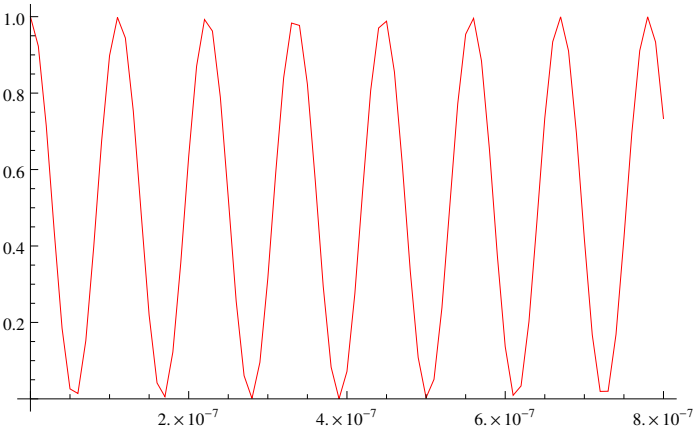
```
    If[phi[[j, k + 1]] == dpos || phi[[j, k + 1]] == dneg,
     H[[i, j]] = u;
     H[[j, i]] = u;
     ];
    ];
   ];
  ];
 ];
```

```
(* The following is used to solve for the time evoultion of the \
interactions of the atoms. The eigenvalues and eigenvectors of the \
Hamiltonian are determined, and the identity matrix is used to \
replace the diagonal elements with the eigenvalues. *)

dataPlot = {};
h = 1;
δ = 10;

evec = Normalize /@ Eigenvectors[N[H]];
eval = Eigenvalues[N[H]];
S = Transpose[evec];
Sdag = ConjugateTranspose[S];
Y = IdentityMatrix[12];

For[w = 1, w <= 12, w++,
 Y[[w, w]] = eval[[w]];
 ];

U[t_] := S.MatrixExp[-I * t * Y / h].Sdag;
initState = {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
probSum = 0;
count = 0;
For[t = 0, t <= 8 * 10^-7, t = t + 10^-8,

 currentState = U[t / au].initState;
 probS =
  Sum[currentState[[qq]] * Conjugate[currentState[[qq]]], {qq, 1, 4}];
 AppendTo[dataPlot, {t, probS}];
 ];

plot0 = ListPlot[dataPlot, Joined -> True, PlotRange -> All,
 PlotStyle -> Red]


(* Below are the calculated frequencies for different angles. The \
```

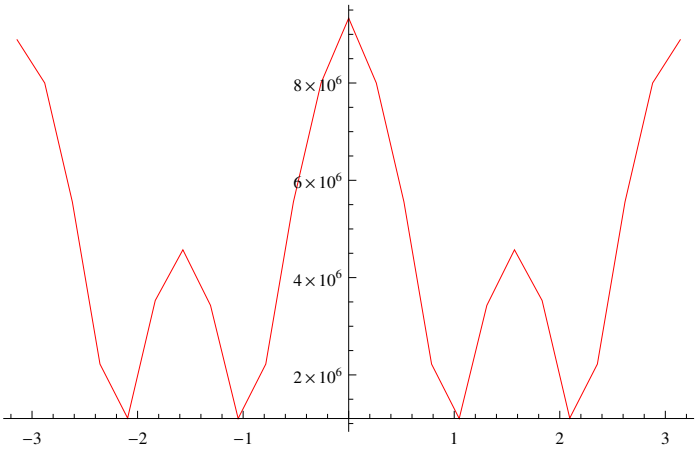
```
first calculated frequency is for 0 radians, and they are calculated \
up to π in steps of π/12. The frequencies were calculated by \
making the probability vs. time plot for each angle, counting the \
number of peaks in a certain time, and dividing these numbers. A list \
was made keeping the angles in order and storing all of their \
frequencies. *)

plot0; (* 7 rotations in 7.5*10^-7s *)
freq0 = 7 / (7.5 * 10^-7);
plotπ12; (* 6 rotations in 7.5*10^-7s *)

freqπ12 = 6 / (7.5 * 10^-7);
plotπ6; (* 10 rotations in 1.8*10^-6s *)

freqπ6 = 10 / (1.8 * 10^-6);
plotπ4; (* 10 rotations in 4.5*10^-6s *)

freqπ4 = 10 / (4.5 * 10^-6);
plotπ3 ; (* 8 rotations in 7.25*10^-6s *)

freqπ3 = 8 / (7.25 * 10^-6);
plot5π12; (* 6 rotations in 1.75*10^-6s *)

freq5π12 = 6 / (1.75 * 10^-6);
plotπ2; (* 8 rotations in 1.75*10^-6s *)

freqπ2 = 8 / (1.75 * 10^-6);
plot7π12; (*6 rotations in 1.7*10^-6s *)

freq7π12 = 6 / (1.7 * 10^-6);
plot2π3; (* 8 rotations in 7.2*10^-6s *)

freq2π3 = 8 / (7.2 * 10^-6);
plot3π4; (* 8 rotations in 3.6*10^-6 *)

freq3π4 = 8 / (3.6 * 10^-6);
plot5π6; (* 10 rotations in 1.8*10^-6s *)

freq5π6 = 10 / (1.8 * 10^-6);
plot11π12; (* 7 rotations in 8.75*10^-7s *)

freq11π12 = 7 / (8.75 * 10^-7);
plotπ; (* 8 rotations in 9*10^-67s *)
freqπ = 8 / (9 * 10^-7);
freqs = {{freq0}, {freqπ12}, {freqπ6}, {freqπ4}, {freq\
π3}, {freq5π12}, {freqπ2}, {freq7π12}, {freq2π3}, \
{freq3π4}, {freq5π6}, {freq11π12}, {freqπ}};
```

```
(* The following creates a plot of frequency of oscillation vs. angle \
between the atoms. Because calculations were only carried-out from 0 \
radians to π radians,a second for-loop was needed to plot the \
symmetric points from π radians to 2π radians. The second \
plot uses the same data but plots it in a polar plot rather than in \
cartesian coordinates. *)

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

angle = 0;
For[ b = 1, b <= Length[freqs], b = b + 1,
 AppendTo[freqplot, {angle, freqs[[b, 1]]}];
 angle = angle + π / 12;
 ]
plotFreq =
 ListPlot[freqplot, Joined -> True,  PlotRange -> All,
  PlotStyle -> Red]
polarListPlot =
 ListPolarPlot[freqplot, Joined -> True, PlotRange -> All,
  PlotStyle -> Purple]
```

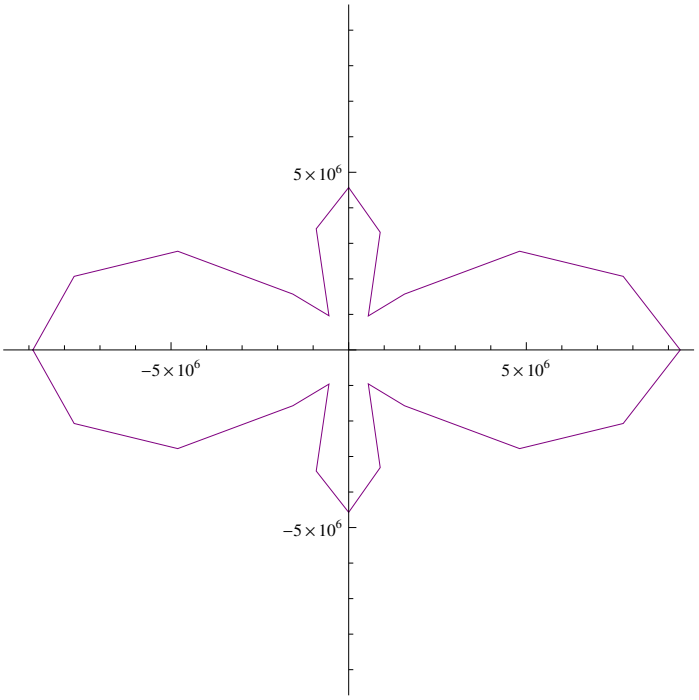
Out[101]=



Out[121]=



Out[122]=

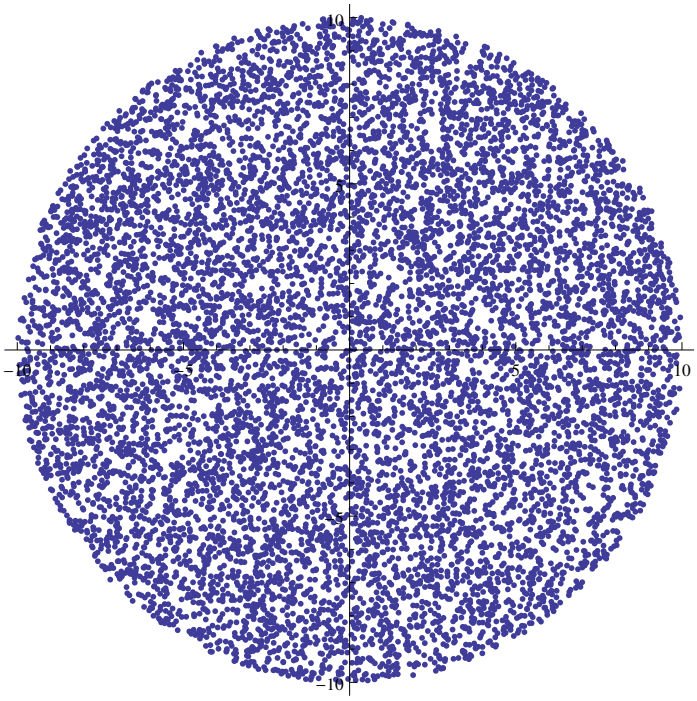


This is code that we used to practice the uniform distribution of atoms in a circle.

```
file = "/home/vesanford/research/multivac/cppExercise/test.dat"

data = BinaryReadList[file, "Real64"];

"/home/vesanford/research/multivac/cppExercise/test.dat"

orderedPairs = Partition[data, 2];

ListPolarPlot[orderedPairs]

(*This is a uniform distribution of atoms over a circle; this is \
important because we need to know how to do this so that our \
simulations are similar to how a group of atoms would be in a lab \
setting. *)
```

Out[59]= /home/vesanford/research/multivac/cppExercise/test.dat

Out[62]=



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*. I load so many different sets of data so that I can see all of the graphs (shown below) at the same time, to compare how different parameters change how the energy exchange occurs,

In[10]:=
```
simulationName = "overlappingSpheres";
data = {};
SetDirectory[
  "/home/vesanford/research/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];
 ];

simulationName = "movingSpheres";
data2 = {};
SetDirectory[
  "/home/vesanford/research/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];
 ];

simulationName = "spreadSpheres";
data3 = {};
SetDirectory[
  "/home/vesanford/research/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];
 ];

simulationName = "spreadSpheres3atoms";
data4 = {};
SetDirectory[
  "/home/vesanford/research/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/vesanford/research/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 = "separated433";
data6 = {};
SetDirectory[
  "/home/vesanford/research/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 = "separated544";
data7 = {};
SetDirectory[
  "/home/vesanford/research/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/vesanford/research/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 = "DIFFERENT";
data9 = {};
SetDirectory[
  "/home/vesanford/research/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];
 ];
```

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[data]
```

```
{11, 2}
```

```
{11, 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., 6}, {5, 10, 34, 0, 0, 0, 0}, {708}, {0, 0, 0, 0, 2}}
```

```
{{6, 5, 2, 0.001`, 3.`, 5}, {1, 2, 10, 0, 0, 0, 0}, {708}, {0, 0, 0, 2, 2}}
```

```
{{6, 5, 2, 0.001, 3., 5}, {1, 2, 10, 0, 0, 0, 0}, {708}, {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}}
```

```
{{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.

```
data3[[1, 1, 1]]
```

```
{{0., 3.39076×10^-31, 1.}, {0.1, 0.038099, 0.923802}, {0.1, 0.0532454, 0.893509},
 {0.2, 0.0637041, 0.872592}, {0.2, 0.0696447, 0.860711}, {0.3, 0.0746708, 0.850658},
 {0.3, 0.0783773, 0.843245}, {0.3, 0.0815794, 0.836841}, {0.4, 0.0831222, 0.833756},
 {0.5, 0.0856038, 0.828792}, {0.5, 0.0865953, 0.826809}, {0.6, 0.0889525, 0.822095},
 {0.6, 0.0910198, 0.81796}, {0.7, 0.0924724, 0.815055}, {0.7, 0.0924886, 0.815023},
 {0.8, 0.0941582, 0.811684}, {0.8, 0.0943865, 0.811227}, {0.9, 0.0949506, 0.810099},
 {0.9, 0.0946281, 0.810744}, {0.9, 0.0948984, 0.810203}, {1., 0.096353, 0.807294},
 {1.1, 0.0965514, 0.806897}, {1.1, 0.0971324, 0.805735}, {1.1, 0.0984357, 0.803129},
 {1.2, 0.0989263, 0.802147}, {1.3, 0.0989112, 0.802178}, {1.3, 0.0992004, 0.801599},
 {1.4, 0.100152, 0.799696}, {1.4, 0.100856, 0.798288}, {1.5, 0.1012, 0.7976},
 {1.5, 0.102121, 0.795758}, {1.6, 0.102502, 0.794997}, {1.6, 0.101946, 0.796109},
 {1.6, 0.10226, 0.79548}, {1.7, 0.103128, 0.793745}, {1.8, 0.102853, 0.794294},
 {1.8, 0.103818, 0.792365}, {1.9, 0.103151, 0.793698}, {1.9, 0.102039, 0.795921},
 {2., 0.101574, 0.796853}, {2., 0.102063, 0.795874}, {2., 0.102042, 0.795915},
 {2.1, 0.101844, 0.796311}, {2.2, 0.101942, 0.796116}, {2.2, 0.101338, 0.797324},
 {2.3, 0.101457, 0.797086}, {2.3, 0.101843, 0.796315}, {2.4, 0.100697, 0.798605},
 {2.4, 0.0998189, 0.800362}, {2.5, 0.100256, 0.799488}, {2.5, 0.100756, 0.798487},
 {2.5, 0.100844, 0.798312}, {2.6, 0.10111, 0.79778}, {2.7, 0.101117, 0.797765},
 {2.7, 0.101674, 0.796652}, {2.8, 0.100904, 0.798192}, {2.8, 0.101199, 0.797602},
 {2.9, 0.101299, 0.797401}, {2.9, 0.102538, 0.794924}, {3., 0.102342, 0.795317},
 {3., 0.102833, 0.794333}, {3., 0.103014, 0.793971}, {3.1, 0.103062, 0.793876},
 {3.2, 0.102874, 0.794253}, {3.2, 0.102333, 0.795334}, {3.3, 0.10305, 0.7939},
 {3.3, 0.103155, 0.793691}, {3.4, 0.102902, 0.794195}, {3.4, 0.103154, 0.793691},
 {3.5, 0.103512, 0.792975}, {3.5, 0.103625, 0.792749}, {3.5, 0.103655, 0.79269},
 {3.6, 0.10323, 0.793541}, {3.7, 0.1022, 0.7956}, {3.7, 0.103058, 0.793884},
 {3.8, 0.104292, 0.791417}, {3.8, 0.104759, 0.790481}, {3.9, 0.104388, 0.791223},
 {3.9, 0.102973, 0.794055}, {4., 0.102817, 0.794367}, {4., 0.102806, 0.794388},
 {4.1, 0.103226, 0.793548}, {4.1, 0.104203, 0.791593}, {4.2, 0.104615, 0.790771},
 {4.2, 0.105002, 0.789995}, {4.3, 0.104225, 0.79155}, {4.3, 0.10323, 0.793539},
 {4.3, 0.10239, 0.79522}, {4.4, 0.104101, 0.791799}, {4.5, 0.104116, 0.791768},
 {4.5, 0.104091, 0.791818}, {4.6, 0.103567, 0.792866}, {4.6, 0.10321, 0.793581},
 {4.7, 0.103782, 0.792435}, {4.7, 0.1042, 0.791601}, {4.8, 0.103347, 0.793306},
 {4.8, 0.103784, 0.792431}, {4.8, 0.102326, 0.795347}, {4.9, 0.101869, 0.796262},
 {5., 0.101232, 0.797536}, {5., 0.101918, 0.796165}, {5.1, 0.101702, 0.796596},
 {5.1, 0.101423, 0.797153}, {5.2, 0.101248, 0.797504}, {5.2, 0.101371, 0.797259},
 {5.3, 0.10096, 0.79808}, {5.3, 0.10023, 0.799539}, {5.3, 0.101163, 0.797673},
 {5.4, 0.100228, 0.799544}, {5.5, 0.100425, 0.79915}, {5.5, 0.102013, 0.795974},
 {5.6, 0.102578, 0.794843}, {5.6, 0.102265, 0.79547}, {5.7, 0.102695, 0.794609},
 {5.7, 0.102845, 0.79431}, {5.8, 0.103704, 0.792592}, {5.8, 0.103616, 0.792768},
 {5.8, 0.103341, 0.793318}, {5.9, 0.103613, 0.792773}, {6., 0.103298, 0.793404},
```

{6., 0.102508, 0.794983}, {6.1, 0.104226, 0.791548}, {6.1, 0.102424, 0.795151},
{6.2, 0.102099, 0.795801}, {6.2, 0.10174, 0.79652}, {6.3, 0.101507, 0.796985},
{6.3, 0.102406, 0.795189}, {6.3, 0.102388, 0.795224}, {6.4, 0.103531, 0.792939},
{6.5, 0.103315, 0.79337}, {6.5, 0.103282, 0.793437}, {6.6, 0.103276, 0.793448},
{6.6, 0.104013, 0.791975}, {6.7, 0.10385, 0.792299}, {6.7, 0.104181, 0.791639},
{6.8, 0.103913, 0.792174}, {6.8, 0.103574, 0.792852}, {6.8, 0.10421, 0.79158},
{6.9, 0.104302, 0.791396}, {7., 0.10534, 0.789321}, {7., 0.104991, 0.790017},
{7.1, 0.10432, 0.79136}, {7.1, 0.104458, 0.791083}, {7.2, 0.103563, 0.792874},
{7.2, 0.103482, 0.793037}, {7.3, 0.103892, 0.792215}, {7.3, 0.102994, 0.794012},
{7.3, 0.102477, 0.795046}, {7.4, 0.10265, 0.7947}, {7.5, 0.103602, 0.792797},
{7.5, 0.104176, 0.791647}, {7.6, 0.104105, 0.79179}, {7.6, 0.103878, 0.792243},
{7.7, 0.103274, 0.793451}, {7.7, 0.103549, 0.792902}, {7.8, 0.103716, 0.792567},
{7.8, 0.10312, 0.793761}, {7.8, 0.102797, 0.794407}, {7.9, 0.103235, 0.79353},
{8., 0.103706, 0.792588}, {8., 0.104479, 0.791042}, {8.1, 0.104537, 0.790927},
{8.1, 0.104108, 0.791784}, {8.2, 0.104949, 0.790103}, {8.2, 0.104048, 0.791903},
{8.3, 0.103525, 0.792951}, {8.3, 0.104001, 0.791999}, {8.4, 0.104685, 0.790631},
{8.4, 0.10386, 0.792279}, {8.4, 0.103691, 0.792617}, {8.5, 0.103412, 0.793176},
{8.6, 0.102666, 0.794669}, {8.6, 0.101922, 0.796157}, {8.7, 0.10252, 0.79496},
{8.7, 0.102398, 0.795204}, {8.8, 0.101527, 0.796946}, {8.8, 0.101408, 0.797185},
{8.9, 0.102262, 0.795476}, {8.9, 0.102837, 0.794326}, {8.9, 0.102153, 0.795694},
{9., 0.102573, 0.794853}, {9.1, 0.102412, 0.795176}, {9.1, 0.103012, 0.793977},
{9.2, 0.103492, 0.793016}, {9.2, 0.103869, 0.792263}, {9.3, 0.104832, 0.790337},
{9.3, 0.103585, 0.792831}, {9.4, 0.10362, 0.792759}, {9.4, 0.10468, 0.790641},
{9.4, 0.105525, 0.788949}, {9.5, 0.105196, 0.789609}, {9.6, 0.104236, 0.791528},
{9.6, 0.104231, 0.791539}, {9.7, 0.103908, 0.792185}, {9.7, 0.104062, 0.791875},
{9.8, 0.103638, 0.792725}, {9.8, 0.103547, 0.792907}, {9.9, 0.102941, 0.794118},
{9.9, 0.10287, 0.79426}, {9.9, 0.104782, 0.790437}, {10., 0.104665, 0.79067}}

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[data3[[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.

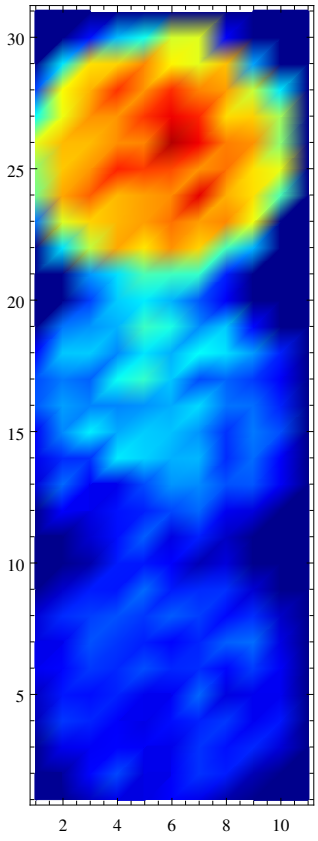
Try executing the following cell for different times by changing the fourth index in the first occurence of data. Note that we divide by the maximum value of *data[[1,1,2,1]]* to normalize all the data the same way. (Why choose that time?)

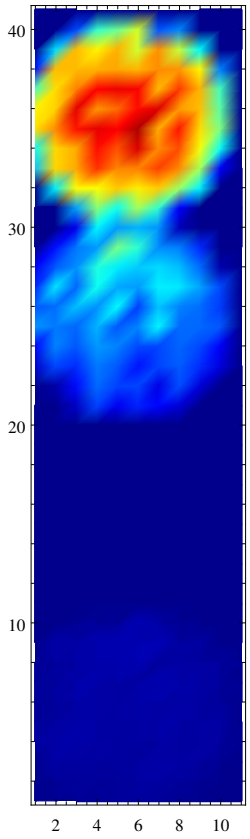
```
Length[data3[[1, 1, 2, 50, 1]]]
```

```
11
```

In[46]:=
```
index = 2;
ListDensityPlot[data6[[index, 1, 2, 100]] / Max[data6[[1, 1, 2, 1]]],
 PlotRange → All, ColorFunction → colorMap, ColorFunctionScaling → False,
 AspectRatio → Length[data6[[index, 1, 2, 1]]] / Length[data6[[index, 1, 2, 1, 1]]]]
```

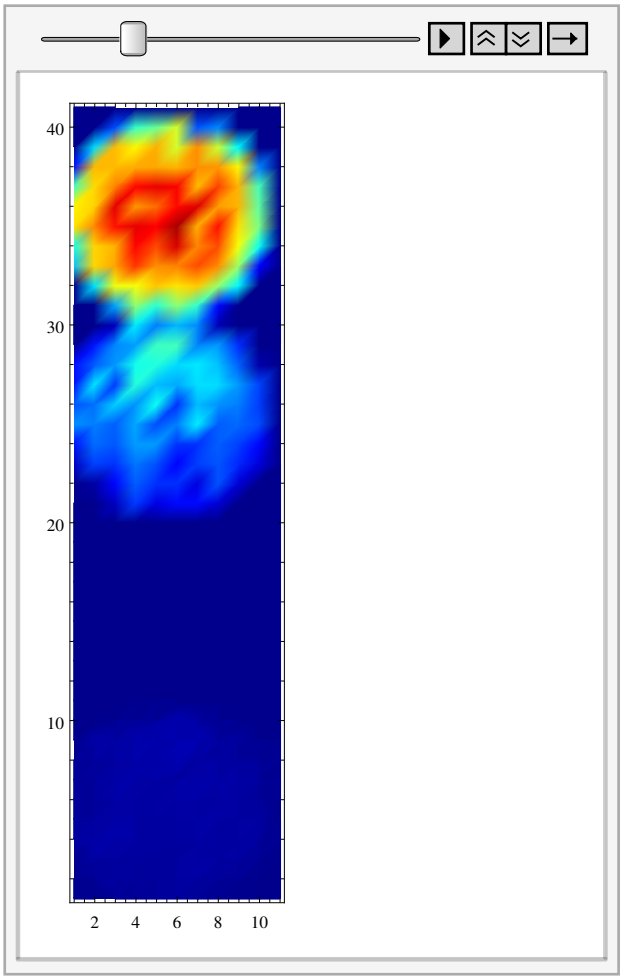
Out[47]=

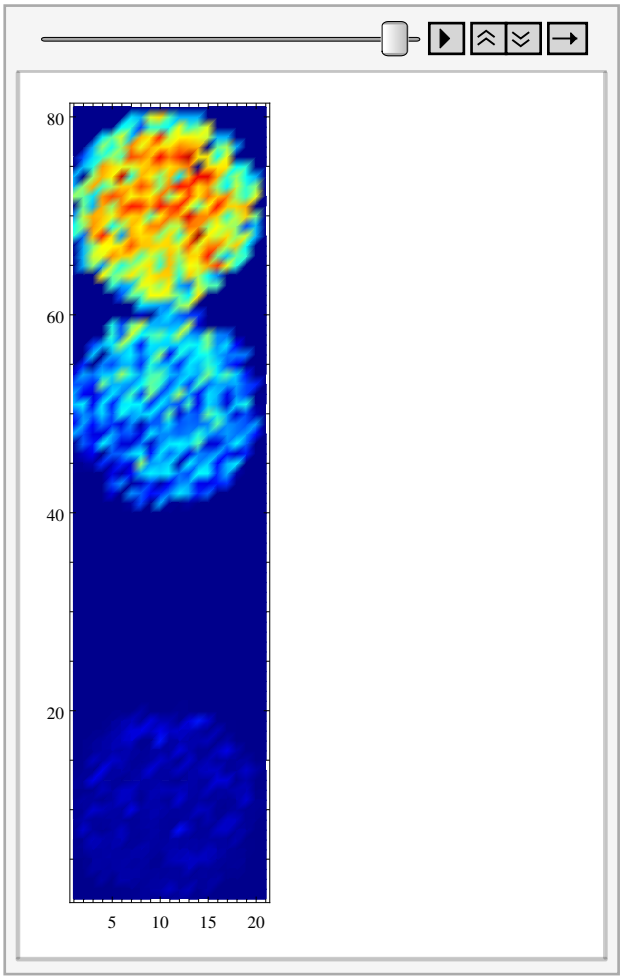




We can also make a movie of the data to watch the energy exchange happen. This might not be interesting if the evolution of your energy exchange is too rapid. You could try a lower density or smaller dipole moment to slow things down so you can see some evolution. You can figure out what all of the parameters to this command mean!

```
ListAnimate[
 Table[ListDensityPlot[data6[[index, 1, 2, i]] / Max[data6[[index, 1, 2, 1]]],
   PlotRange → All, ColorFunction → colorMap, ColorFunctionScaling → False,
   AspectRatio → Length[data6[[index, 1, 2, 1]]] / Length[data6[[index, 1, 2, 1, 1]]]],
  {i, 1, Length[data6[[1, 1, 2]]] / 5, 1}]]
```






```
(* When the electric fields are in the y and z directions,
The interactions of the p atoms only occur with the group of s atoms
 further away from the p atoms, but straight below them. However,
when the electric field is in the x direction, the interaction
 occurs with the group of s atoms that is closer to the p atoms. *)

(* This is a plot which shows 3 groups of atoms
 in a vertical line. The group of p atoms is on top,
and there are 2 groups of s atoms below. A loop is set up so that the
 bottom group of s atoms moves down in steps. This way, the difference in
 interaction depending on the distance between the groups can be analyzed. *)
```

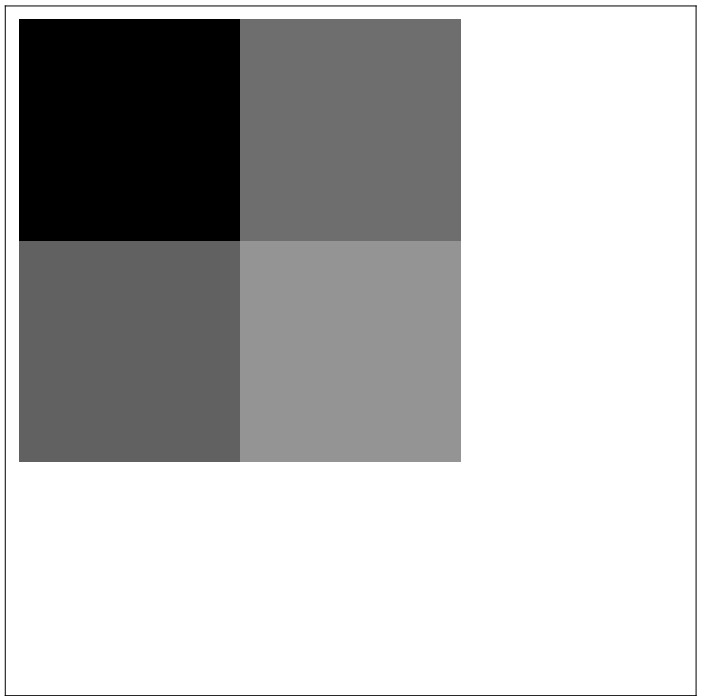
```
MatrixForm[data[[4, 1, 2, 1]]]
```

$$\begin{pmatrix} 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 6.43337\times10^{-32} & 0. \\ 0. & 0. & 0. & 7.07564\times10^{-33} & 4.95695\times10^{-33} & 1.37323\times10 \\ 0. & 0. & 0. & 9.60561\times10^{-32} & 7.60095\times10^{-33} & 6.47024\times10 \\ 0. & 0. & 4.4211\times10^{-30} & 3.57884\times10^{-32} & 2.56726\times10^{-31} & 7.64154\times10 \\ 0. & 9.07763\times10^{-33} & 3.35262\times10^{-31} & 1.66796\times10^{-31} & 5.93959\times10^{-32} & 1.58229\times10 \end{pmatrix}$$

0.	1.61475×10^{-32}	9.07889×10^{-33}	2.84139×10^{-31}	1.18115×10^{-31}	1.34473×10
7.29747×10^{-32}	1.05305×10^{-31}	1.2194×10^{-31}	5.43678×10^{-34}	2.0755×10^{-31}	1.18123×10
2.56442×10^{-32}	4.21371×10^{-31}	1.42038×10^{-31}	2.37456×10^{-31}	1.60391×10^{-31}	1.47323×10
3.5142×10^{-32}	2.06619×10^{-31}	4.03087×10^{-31}	7.63169×10^{-31}	2.95645×10^{-32}	1.30198×10
2.00755×10^{-31}	3.42701×10^{-31}	1.66428×10^{-31}	6.06008×10^{-32}	3.90449×10^{-32}	2.52019×10
1.69089×10^{-32}	4.45957×10^{-32}	1.4928×10^{-31}	4.43438×10^{-31}	1.58587×10^{-31}	1.88385×10
3.16232×10^{-32}	1.36877×10^{-31}	1.18588×10^{-31}	3.23112×10^{-31}	1.59239×10^{-32}	6.44372×10
1.13493×10^{-32}	7.2834×10^{-31}	2.20821×10^{-31}	3.53025×10^{-31}	6.91368×10^{-32}	2.21004×10
0.	3.05236×10^{-34}	8.14305×10^{-32}	2.05947×10^{-31}	1.31378×10^{-31}	4.22177×10
0.	0.	1.5054×10^{-32}	4.87276×10^{-31}	6.85608×10^{-32}	6.43595×10
0.	0.	7.08471×10^{-33}	1.28176×10^{-31}	7.62061×10^{-32}	1.71351×10
0.	0.	0.	0.	1.18526×10^{-31}	9.80056×10
0.	0.	0.	0.	1.43885×10^{-32}	3.73583×10
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.

0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	$4.50667\times10^{}$
0.	0.	0.	7.71014×10^{-31}	6.2086×10^{-31}	$1.17232\times10^{}$
0.	0.	0.	3.14307×10^{-31}	8.87054×10^{-32}	$6.72435\times10^{}$
0.	0.	2.18195×10^{-30}	3.97411×10^{-31}	1.32153×10^{-30}	$1.15723\times10^{}$
0.	1.53152×10^{-30}	6.14415×10^{-31}	1.25768×10^{-30}	1.17199×10^{-30}	$8.23966\times10^{}$
4.51763×10^{-31}	9.53353×10^{-32}	1.06466×10^{-30}	1.38528×10^{-30}	9.13513×10^{-31}	$3.38838\times10^{}$
5.8553×10^{-31}	1.31041×10^{-31}	2.00548×10^{-30}	1.44403×10^{-30}	1.18177×10^{-30}	$7.85826\times10^{}$
7.10899×10^{-31}	6.36994×10^{-31}	1.05965×10^{-30}	6.5138×10^{-31}	3.88262×10^{-31}	$1.4793\times10^{-}$
9.3591×10^{-31}	0.	1.79971×10^{-30}	3.50899×10^{-30}	1.56928×10^{-30}	$2.51401\times10^{}$
6.66565×10^{-32}	8.01181×10^{-31}	1.06043×10^{-30}	1.55607×10^{-30}	1.01081×10^{-30}	4.224×10^{-3}
0.	1.09639×10^{-30}	3.06902×10^{-31}	3.11381×10^{-31}	2.6956×10^{-30}	$5.1912\times10^{-}$
2.27575×10^{-31}	3.28593×10^{-30}	9.0242×10^{-31}	1.20857×10^{-30}	1.93442×10^{-30}	$1.34689\times10^{}$
0.	0.	3.57967×10^{-30}	7.18137×10^{-31}	2.28041×10^{-30}	$1.56582\times10^{}$
0.	2.09505×10^{-31}	1.25565×10^{-30}	7.34016×10^{-31}	4.03993×10^{-31}	$3.4124\times10^{-}$
0.	0.	1.71943×10^{-31}	2.29775×10^{-30}	1.56174×10^{-30}	$1.23807\times10^{}$
0.	0.	1.98242×10^{-31}	1.32148×10^{-30}	1.06061×10^{-30}	$1.83228\times10^{}$
0.	0.	0.	5.03667×10^{-31}	1.0699×10^{-30}	$5.84061\times10^{}$
0.	0.	0.	0.	0.	2.882×10^{-3}
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	4.
0.	0.	0.	1.	1.	3.
0.	0.	1.	4.	5.	3.
0.	1.	8.	3.	7.	7.
0.	0.	4.	4.	8.	5.
0.	2.	5.	5.	6.	8.
2.	3.	8.	6.	1.	5.
3.	8.	8.	1.	5.	2.
3.	1.	5.	4.	6.	4.
2.	3.	5.	3.	8.	11.
2.	6.	2.	5.	2.	7.
0.	4.	5.	5.	6.	4.
0.	2.	3.	6.	6.	9.
0.	0.	8.	4.	3.	5.
0.	1.	4.	1.	3.	5.
0.	0.	1.	0.	2.	9.
0.	0.	0.	0.	2.	3.
0.	0.	0.	0.	2.	2.
0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.

```
ArrayPlot[data[[1, 1, 2, 10]]]
```



data[[1, 1, 1]]

A very large output was generated. Here is a sample of it:

{{0., 2.20667 × 10^-32, 1.}, {0., 0.127112, 0.49155},
 <<1996>>, {2., 0.166865, 0.332541}, {2., 0.160611, 0.357557}}

Show Less | Show More | Show Full Output | Set Size Limit...

data[[2, 2]]

{{5, 4, 2, 0.001, 3., 5}, {1, 0, 10, 0, 0, 0, 0}, {708}, {0, 0, 0, 0, 2}}

Length[data[[1, 1, 1]]]

2000

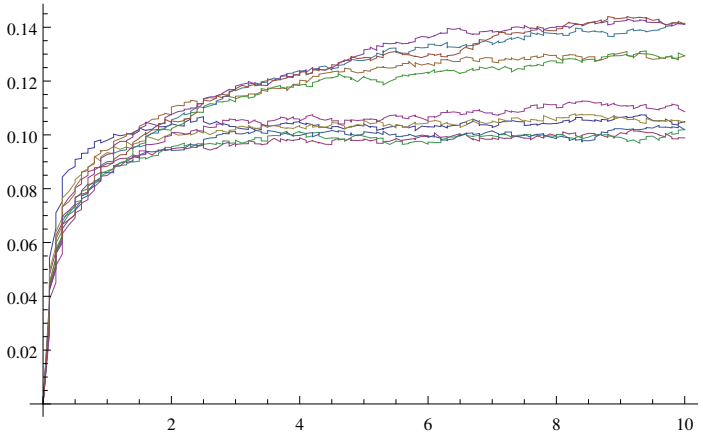
data[[1, 1, 1]]

{{0., 2.47294 × 10^-31, 1.}, {0.1, 0.00530353, 0.989393}, {0.1, 0.0125649, 0.97487},
 {0.2, 0.0174696, 0.965061}, {0.2, 0.0217579, 0.956484}, {0.3, 0.0262154, 0.947569},
 {0.3, 0.0292921, 0.941416}, {0.3, 0.0316486, 0.936703}, {0.4, 0.0338227, 0.932355},
 {0.5, 0.0358532, 0.928294}, {0.5, 0.0377852, 0.92443}, {0.6, 0.0398552, 0.92029},
 {0.6, 0.0411667, 0.917667}, {0.7, 0.0424099, 0.91518}, {0.7, 0.0438772, 0.912246},
 {0.8, 0.0453912, 0.909218}, {0.8, 0.0474007, 0.905199}, {0.9, 0.0492203, 0.90156},
 {0.9, 0.0505032, 0.898994}, {0.9, 0.0516373, 0.896725}, {1., 0.0530291, 0.893942},
 {1.1, 0.0540404, 0.891919}, {1.1, 0.0548551, 0.89029}, {1.1, 0.0558005, 0.888399},
 {1.2, 0.0561462, 0.887708}, {1.3, 0.0570098, 0.88598}, {1.3, 0.0578257, 0.884349},
 {1.4, 0.0577322, 0.884536}, {1.4, 0.0581009, 0.883798}, {1.5, 0.0592299, 0.88154},
 {1.5, 0.0605616, 0.878877}, {1.6, 0.0611798, 0.87764}, {1.6, 0.0615833, 0.876833},
 {1.6, 0.0628362, 0.874328}, {1.7, 0.0640423, 0.871915}, {1.8, 0.0648221, 0.870356},

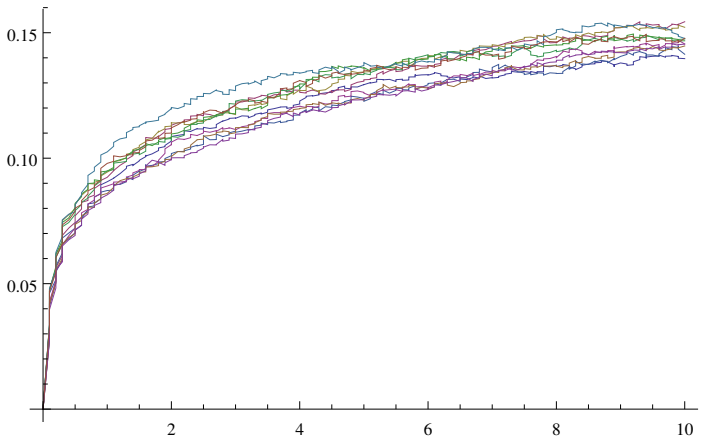
{1.8, 0.0653654, 0.869269}, {1.9, 0.0659649, 0.86807}, {1.9, 0.0663965, 0.867207},
{2., 0.0669058, 0.866188}, {2., 0.0679589, 0.864082}, {2., 0.068313, 0.863374},
{2.1, 0.0681056, 0.863789}, {2.2, 0.0688575, 0.862285}, {2.2, 0.069656, 0.860688},
{2.3, 0.0697838, 0.860432}, {2.3, 0.0694546, 0.861091}, {2.4, 0.0698513, 0.860297},
{2.4, 0.0705325, 0.858935}, {2.5, 0.0708071, 0.858386}, {2.5, 0.071302, 0.857396},
{2.5, 0.0719528, 0.856094}, {2.6, 0.0727181, 0.854564}, {2.7, 0.0727929, 0.854414},
{2.7, 0.0728013, 0.854397}, {2.8, 0.073542, 0.852916}, {2.8, 0.0739452, 0.85211},
{2.9, 0.0742462, 0.851508}, {2.9, 0.0747085, 0.850583}, {3., 0.0750523, 0.849895},
{3., 0.0751891, 0.849622}, {3., 0.0758627, 0.848275}, {3.1, 0.0762678, 0.847464},
{3.2, 0.0763112, 0.847378}, {3.2, 0.0768395, 0.846321}, {3.3, 0.077696, 0.844608},
{3.3, 0.0776048, 0.84479}, {3.4, 0.0780118, 0.843976}, {3.4, 0.0791796, 0.841641},
{3.5, 0.0796843, 0.840631}, {3.5, 0.0795579, 0.840884}, {3.5, 0.0798821, 0.840236},
{3.6, 0.0804743, 0.839051}, {3.7, 0.0804614, 0.839077}, {3.7, 0.0804987, 0.839003},
{3.8, 0.0806155, 0.838769}, {3.8, 0.0808402, 0.83832}, {3.9, 0.0813344, 0.837331},
{3.9, 0.0815675, 0.836865}, {4., 0.0820475, 0.835905}, {4., 0.0823168, 0.835366},
{4.1, 0.0822816, 0.835437}, {4.1, 0.0827217, 0.834557}, {4.2, 0.0836981, 0.832604},
{4.2, 0.0838818, 0.832236}, {4.3, 0.0836446, 0.832711}, {4.3, 0.0840772, 0.831846},
{4.3, 0.0847619, 0.830476}, {4.4, 0.0847506, 0.830499}, {4.5, 0.0849493, 0.830101},
{4.5, 0.0855766, 0.828847}, {4.6, 0.0853847, 0.829231}, {4.6, 0.0852484, 0.829503},
{4.7, 0.0857905, 0.828419}, {4.7, 0.086291, 0.827418}, {4.8, 0.0863938, 0.827212},
{4.8, 0.0867471, 0.826506}, {4.8, 0.0871002, 0.8258}, {4.9, 0.0872325, 0.825535},
{5., 0.0875651, 0.82487}, {5., 0.0881388, 0.823722}, {5.1, 0.0886414, 0.822717},
{5.1, 0.0887439, 0.822512}, {5.2, 0.0885618, 0.822876}, {5.2, 0.0882103, 0.823579},
{5.3, 0.0880826, 0.823835}, {5.3, 0.0883187, 0.823363}, {5.3, 0.0887476, 0.822505},
{5.4, 0.0887353, 0.822529}, {5.5, 0.0887492, 0.822502}, {5.5, 0.0895143, 0.820971},
{5.6, 0.0904735, 0.819053}, {5.6, 0.0901678, 0.819664}, {5.7, 0.0897363, 0.820527},
{5.7, 0.0906746, 0.818651}, {5.8, 0.0917055, 0.816589}, {5.8, 0.0914837, 0.817033},
{5.8, 0.0912225, 0.817555}, {5.9, 0.0916034, 0.816793}, {6., 0.0923394, 0.815321},
{6., 0.0924317, 0.815137}, {6.1, 0.0926826, 0.814635}, {6.1, 0.0931624, 0.813675},
{6.2, 0.0933643, 0.813271}, {6.2, 0.0935385, 0.812923}, {6.3, 0.0944343, 0.811131},
{6.3, 0.0950583, 0.809883}, {6.3, 0.095026, 0.809948}, {6.4, 0.0951358, 0.809728},
{6.5, 0.0958361, 0.808328}, {6.5, 0.0962492, 0.807502}, {6.6, 0.0963446, 0.807311},
{6.6, 0.0969898, 0.80602}, {6.7, 0.0977599, 0.80448}, {6.7, 0.0976113, 0.804777},
{6.8, 0.0979447, 0.804111}, {6.8, 0.098566, 0.802868}, {6.8, 0.0985338, 0.802932},
{6.9, 0.0983771, 0.803246}, {7., 0.0986852, 0.80263}, {7., 0.098745, 0.80251},
{7.1, 0.0991641, 0.801672}, {7.1, 0.0999704, 0.800059}, {7.2, 0.100602, 0.798796},
{7.2, 0.100244, 0.799511}, {7.3, 0.100018, 0.799965}, {7.3, 0.100281, 0.799438},
{7.3, 0.100145, 0.79971}, {7.4, 0.0995708, 0.800858}, {7.5, 0.0995589, 0.800882},
{7.5, 0.0997818, 0.800436}, {7.6, 0.0994762, 0.801048}, {7.6, 0.099465, 0.80107},
{7.7, 0.0995467, 0.800907}, {7.7, 0.099875, 0.80025}, {7.8, 0.0999517, 0.800097},
{7.8, 0.100038, 0.799925}, {7.8, 0.100152, 0.799695}, {7.9, 0.100079, 0.799842},
{8., 0.100229, 0.799542}, {8., 0.100815, 0.79837}, {8.1, 0.101093, 0.797813},
{8.1, 0.100326, 0.799348}, {8.2, 0.0999253, 0.800149}, {8.2, 0.100278, 0.799443},
{8.3, 0.100595, 0.79881}, {8.3, 0.100189, 0.799623}, {8.4, 0.0996904, 0.800619},
{8.4, 0.100081, 0.799839}, {8.4, 0.101047, 0.797906}, {8.5, 0.101366, 0.797269},
{8.6, 0.101431, 0.797138}, {8.6, 0.101744, 0.796512}, {8.7, 0.102326, 0.795348},
{8.7, 0.103225, 0.79355}, {8.8, 0.103287, 0.793427}, {8.8, 0.102888, 0.794223},
{8.9, 0.102769, 0.794463}, {8.9, 0.103096, 0.793809}, {8.9, 0.102667, 0.794666},
{9., 0.101961, 0.796079}, {9.1, 0.102608, 0.794784}, {9.1, 0.103809, 0.792381},

```
 {9.2, 0.104169, 0.791663}, {9.2, 0.104437, 0.791126}, {9.3, 0.104874, 0.790251},
 {9.3, 0.104922, 0.790156}, {9.4, 0.105034, 0.789931}, {9.4, 0.105459, 0.789082},
 {9.4, 0.105576, 0.788847}, {9.5, 0.105886, 0.788229}, {9.6, 0.10624, 0.78752},
 {9.6, 0.106504, 0.786993}, {9.7, 0.106394, 0.787213}, {9.7, 0.105663, 0.788674},
 {9.8, 0.105343, 0.789314}, {9.8, 0.105566, 0.788868}, {9.9, 0.105812, 0.788376},
 {9.9, 0.106181, 0.787638}, {9.9, 0.106506, 0.786988}, {10., 0.106231, 0.787538}}
```

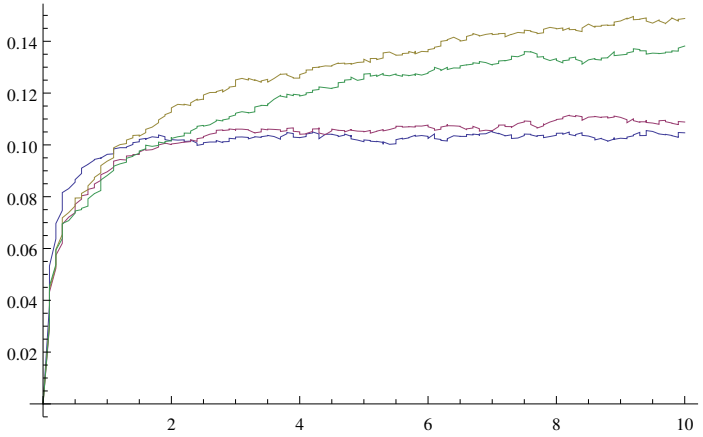
```
ListPlot[Table[{#[[1]], #[[2]]} & /@ data[[i, 1, 1]], {i, 1, 11}],
 Joined → True, PlotRange → All]
(*This is showing when the two s spheres are overlapping out to 30 microns
 away in steps of 2. There becomes more exchange at further distances. *)
```



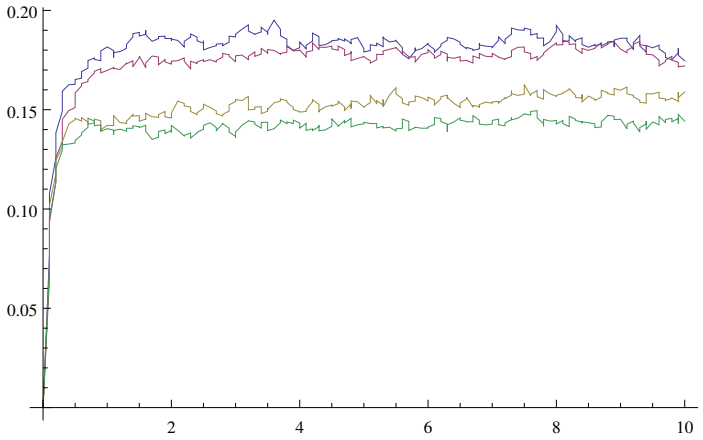

```
ListPlot[Table[{#[[1]], #[[2]]} & /@ data2[[i, 1, 1]], {i, 1, 11}],
 Joined → True, PlotRange → All]
(* This is showing from when the second s sphere is 30
 mircrons away until it is 50 microns away in steps of 2 *)
```



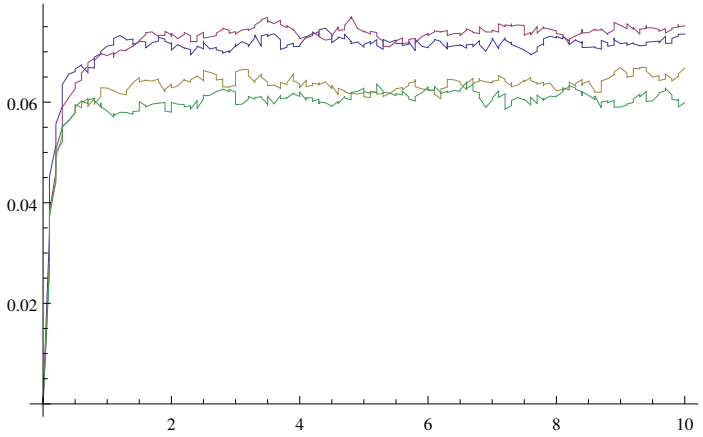
```
(* This is from when the s spheres are overlapping
 until the far one is 40 microns from the p sphere in
 steps of 10 microns. There are 4 atoms in each group. *)
ListPlot[Table[{#[[1]], #[[2]]} & /@ data3[[i, 1, 1]], {i, 1, 4}],
 Joined → True, PlotRange → All]
```



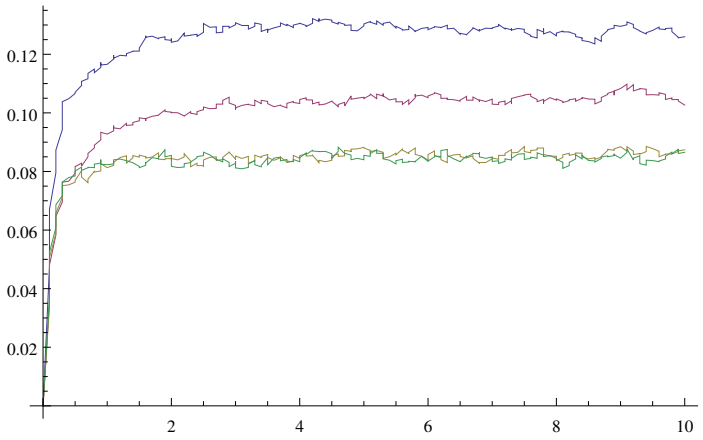

```
(* same as above;;
 From when the s spheres are overlapping until the far one is 40 microns from the
  p sphere in steps of 10 microns for 3 atoms instead of 4 in each group *)
ListPlot[Table[{#[[1]], #[[2]]} & /@ data4[[i, 1, 1]], {i, 1, 4}],
 Joined → True, PlotRange → All]
```



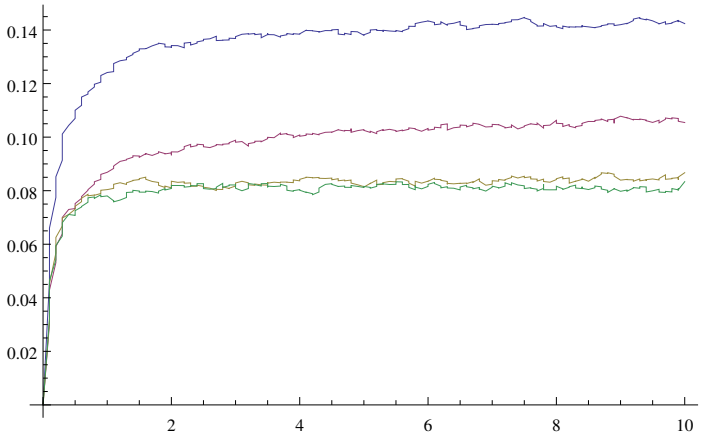
```
(* Same as above;;From when the s spheres are overlapping
  until the far one is 40 microns from the p sphere in steps of 10
  microns for 3 p atoms and 4 s atoms in each of the 2 groups. *)
ListPlot[Table[{#[[1]], #[[2]]} & /@ data5[[i, 1, 1]], {i, 1, 4}],
 Joined → True, PlotRange → All]
```



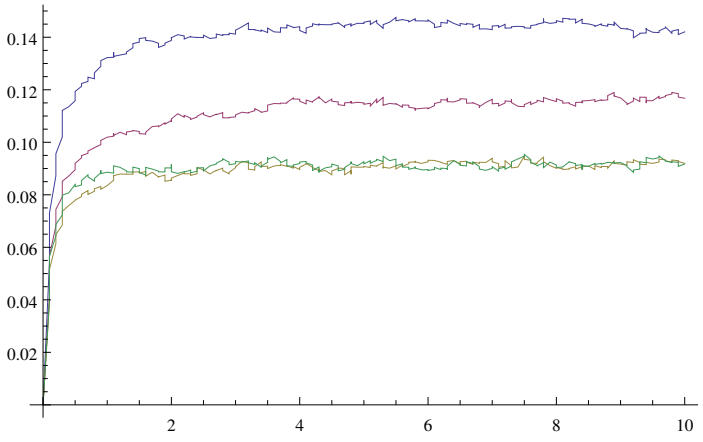

```
(* Same as above;;From when the s spheres are overlapping
  until the far one is 40 microns from the p sphere in steps of 10
  microns for 4 p atoms and 3 s atoms in each of the 2 groups. *)
ListPlot[Table[{#[[1]], #[[2]]} & /@ data6[[i, 1, 1]], {i, 1, 4}],
 Joined → True, PlotRange → All]
```



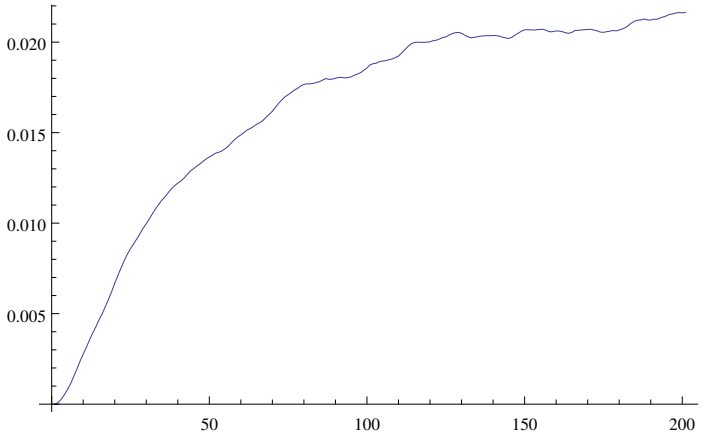
```
(* Same as above;;From when the s spheres are overlapping
  until the far one is 40 microns from the p sphere in steps of 10
  microns for 5 p atoms and 4 s atoms in each of the 2 groups. *)
ListPlot[Table[{#[[1]], #[[2]]} & /@ data7[[i, 1, 1]], {i, 1, 4}],
 Joined → True, PlotRange → All]
```



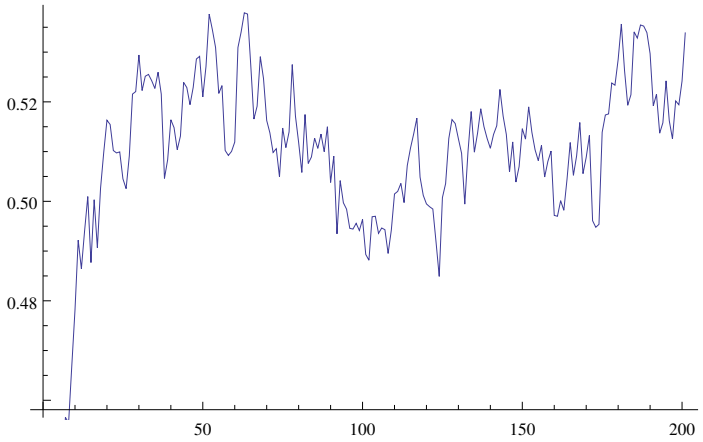

```
(* Same as above;;From when the s spheres are overlapping
  until the far one is 40 microns from the p sphere in steps of 10
  microns for 5 p atoms and 4 s atoms in each of the 2 groups. *)
ListPlot[Table[{#[[1]], #[[2]]} & /@ data8[[i, 1, 1]], {i, 1, 4}],
 Joined → True, PlotRange → All]
```



```
xx = Table[Sum[data5[[index, 1, 2, t, k, j]], {k, 1, 10, 1}, {j, 1, 10, 1}] / 500,
    {t, 1, Length[data5[[index, 1, 2]]], 1}];
ListPlot[xx, Joined → True]
```




```
yy = Table[Sum[data5[[index, 1, 2, t, k, j]],
      {k, (index * 10) - 9, (index * 10), 1}, {j, 1, 10, 1}] / 500,
    {t, 1, Length[data5[[index, 1, 2]]], 1}];
ListPlot[yy, Joined → True]
```



In[50]:= `MatrixForm[data3[[3, 1, 2, 30]]]`

Out[50]//MatrixForm=

0.	0.	0.000736258	0.00788042	0.00998577	0.00948306	0.0239641	9.2
0.	0.0138504	0.058786	0.0233778	0.0420198	0.130857	0.0632798	0.
0.00294683	0.0157795	0.0133778	0.0486925	0.0686295	0.0484099	0.0443745	0.
0.0607304	0.0199434	0.0959368	0.117987	0.0307704	0.0201615	0.0348992	0
0.0501602	0.0801729	0.0648622	0.027333	0.0440821	0.0757391	0.0866121	0
0.0184884	0.0254865	0.100193	0.0444712	0.0517953	0.04141	0.104779	0
0.0596525	0.109649	0.130316	0.123074	0.227894	0.0430955	0.128081	0
0.00231184	0.00653361	0.0767558	0.111806	0.0924875	0.0531674	0.0754054	0.
0.	0.0197534	0.155852	0.171514	0.168868	0.155199	0.0937732	0.
0.	0.	0.00269646	0.0294765	0.0097014	0.114131	0.0320678	0.
0.	0.	0.	0.	0.	0.	0.	
0.	0.	0.	0.	0.	0.	0.	
0.	0.	0.	0.	0.	0.	0.	
0.	0.	0.	0.	0.	0.	0.	
0.	0.	0.	0.	0.	0.	0.	
0.	0.	0.	0.	0.	0.	0.	
0.	0.	0.	0.	0.	0.	0.	
0.	0.	0.	0.	0.	0.	0.	
0.	0.	0.	0.	0.	0.	0.	
0.	0.	0.	0.	0.	0.	0.	
0.	0.	0.633257	0.873954	1.23593	4.73912	1.02687	0
0.	1.06805	2.09058	4.63145	3.38075	3.22598	2.54252	1
1.04854	4.53315	4.83196	3.64785	5.55963	7.52475	4.54382	4
1.1381	3.32743	6.5797	6.01602	4.48729	5.72294	4.43104	6
2.76886	4.25936	6.02429	5.24159	7.3492	8.54018	5.29092	3
3.83449	4.56729	6.11898	5.33444	5.95444	12.2987	8.8662	3
2.02127	4.81361	4.61559	5.10782	9.96149	6.67143	8.27105	5
1.62276	4.81306	10.7753	10.639	9.70781	7.57351	9.7373	6
0.	2.70411	3.64144	8.22705	13.2864	7.3351	7.93267	5
0.	0.	0.561279	5.76429	4.46413	10.0279	3.38185	1
0.	0.	2.05137	7.17049	8.65815	8.4709	5.86431	0
0.	10.9725	15.3004	15.5914	21.928	21.8986	14.1042	1
3.33615	23.3655	26.0558	20.1167	32.5956	29.3768	26.5652	1
7.86749	23.7645	17.2343	27.3863	33.1966	24.0431	24.9223	1
7.24993	18.5456	20.6897	28.4264	28.0678	38.9911	32.2057	3
13.1993	23.1248	31.9227	32.3817	29.392	32.5331	33.7773	2
3.87689	20.4808	29.1018	25.1467	22.9139	35.098	31.3436	2
2.38817	15.631	24.3899	12.82	31.7647	30.74	36.5284	2
0.	2.35319	14.4897	19.4329	38.8653	25.0725	24.5513	1
0.	0.	1.38699	9.16543	12.2558	15.342	7.70265	4
0.	0.	0.	0.	0.	0.	0.	