

```
In[160]:= BeginPackage["UVW`Billiard`"]
```

```
NextBounce::usage = "
```

```
NextBounce[currentposition,r] computes the next bouncing point of a ball inside a square table with a circular obstacle of radius r. The initial conditions are given in the list currentposition that has four real coordinates, respectively the abscissa, ordinate, incoming angle of the ball, and the current time. The result is returned as another list of four elements, the abscissa and ordinate of the new bouncing point, the new direction of the ball and the current time incremented by the running time between the two bounces."
```

```
Trajectory::usage = "
```

```
Trajectory[shootingangle,tmax,r] draws a square table with a centered circular obstacle of radius r (default 0.5). Draws inside this support the trajectory of a ball starting at the bottom left corner with initial direction shootingangle, up to time tmax. "
```

```
Bundle::usage = "
```

```
Bundle[listofangles,tmax,r] draws a square table with a centered circular obstacle of radius r. Draws inside the trajectories of balls starting at the bottom left corner with initial directions read in listofangles, up to time tmax. "
```

```
Diff2::usage = "
```

```
Diff2[angle,dangle,tmax,r] simulates two trajectories of balls in a square table with a centered circular obstacle with radius r. Both trajectories start from the bottom left corner. They are followed up to time tmax. The shooting angle of the first trajectory is angle, its difference with the second shooting angle is dangle. The function represents three consecutive graphics. The first one is the billiard table with the two trajectories. The second one is the evolution of the absolute difference of angles as a function of time. The third one is the norm of the difference of positions as a function of time. "
```

```
Begin["`Private`"]
```

```
NextBounce[currentposition_List, r_] :=
```

```
Block[{xold, yold, aold, cosaold, sinaold, epsi = 10-6, tnorth, teast, tsouth, twest, tcircle1, tcircle2, delta, beta, trun, xnew, ynew, anew},  
(*Old coordinates-----*)xold = currentposition[[1]];  
yold = currentposition[[2]];  
aold = currentposition[[3]];  
cosaold = Cos[aold];  
sinaold = Sin[aold];  
(*Hitting time of the four edges-----*)tnorth = (1 - yold) / sinaold;  
If[tnorth < epsi, tnorth = Infinity];  
teast = (1 - xold) / cosaold; If[teast < epsi, teast = Infinity];  
tsouth = (-1 - yold) / sinaold; If[tsouth < epsi, tsouth = Infinity];
```

```

twest = (-1 - xold) / cosaold; If[twest < epsi, twest = Infinity];
(*Hitting time of the circular obstacle-----*)
delta = (xold * cosaold + yold * sinaold)^2 - (xold^2 + yold^2 - r^2);
If[delta > 0, (delta = Sqrt[delta];
  tcircle1 = - (xold * cosaold + yold * sinaold) + delta;
  If[tcircle1 < epsi, tcircle1 = Infinity];
  tcircle2 = - (xold * cosaold + yold * sinaold) - delta;
  If[tcircle2 < epsi, tcircle2 = Infinity];), (tcircle1 = Infinity;
  tcircle2 = Infinity;)];
(*Next bounce is at minimal hitting time-----*)
trun = Min[{tnorth, teast, tsouth, twest, tcircle1, tcircle2}];
Switch[Position[{tnorth, teast, tsouth, twest, tcircle1, tcircle2}, trun][[1, 1]], 1,
  (*Next bounce on north edge-----*) (xnew = xold + trun * cosaold;
  ynew = 1.;
  anew = N[2 * Pi] - aold;), 2,
  (*Next bounce on east edge-----*) (ynew = yold + trun * sinaold;
  xnew = 1.;
  anew = N[Pi] - aold;
  If[anew < 0, anew = anew + N[2 * Pi]]);), 3,
  (*Next bounce on south edge-----*) (xnew = xold + trun * cosaold;
  ynew = -1.;
  anew = N[2 * Pi] - aold;), 4,
  (*Next bounce on west edge-----*) (ynew = yold + trun * sinaold;
  xnew = -1.;
  anew = N[Pi] - aold;
  If[anew < 0, anew = anew + N[2 * Pi]]);), 5,
  (*Next bounce on circle-----*) (xnew = xold + trun * cosaold;
  ynew = yold + trun * sinaold;
  anew = Mod[2 * ArcTan[xnew, ynew] - aold, N[2 * Pi]]);), 6,
  (*Next bounce on circle-----*) (xnew = xold + trun * cosaold;
  ynew = yold + trun * sinaold;
  anew = Mod[2 * ArcTan[xnew, ynew] - aold - N[Pi], N[2 * Pi]]););
(*Protection against leaky corners-----*)
If[(Abs[xnew * ynew] > 1 - epsi), If[(aold < N[Pi/2]),
  anew = N[3 * Pi/2] - aold, If[(aold < N[Pi]), anew = N[5 * Pi/2] - aold,
  If[(aold < N[3 * Pi/2]), anew = N[3 * Pi/2] - aold, anew = N[3 * Pi/2] - aold]]];
Return[{xnew, ynew, anew, currentposition[[4]] + trun}];]

```

```

Trajectory[shootingangle_, tmax_, r_] :=
Block[{running = 0., traj = {{-1., -1.}}, next, next = {-1., -1., N[shootingangle], 0.};
While[running < N[tmax], (next = NextBounce[next, r];
  traj = Append[traj, Take[next, 2]];
  running = Last[next];)];
Show[Graphics[{Thickness[0.01], Line[{{-1, -1}, {-1, 1}, {1, 1}, {1, -1}, {-1, -1}}],
  Circle[{0, 0}, r], Thickness[0.005], Line[traj]}], AspectRatio -> 1];]

```

```

Bundle[listofangles_List, tmax_, r_] :=
Block[{nbtraj, running = 0., traj, next, i, g1, g2}, nbtraj = Length[listofangles];
traj = Table[{{-1., -1.}}, {nbtraj}];
Do[{next = {-1., -1., N[listofangles[[i]]], 0.};
running = 0.;
While[running < N[tmax], {next = NextBounce[next, r];
traj[[i]] = Append[traj[[i]], Take[next, 2]];
running = Last[next];}], {i, 1, nbtraj}];
g1 = Graphics[{Thickness[0.01],
Line[{{-1, -1}, {-1, 1}, {1, 1}, {1, -1}, {-1, -1}}, Circle[{0, 0}, r]}];
g2 = Table[Graphics[{Thickness[0.005], Line[traj[[i]]]}], {i, 1, nbtraj}];
Show[Prepend[g2, g1], AspectRatio → 1];

```

```

Diff2[angle_, dangle_, tmax_, r_] :=
Block[{listofangles, running = 0., traj, next, i, g1, g2, angles, diffangles,
aold, diffpos, i1, i2}, listofangles = N[{angle, angle + dangle}];
traj = Table[{{-1., -1., listofangles[[i]], 0.}}, {i, 1, 2}];
Do[{next = {-1., -1., listofangles[[i]], 0.};
running = 0.;
While[running < N[tmax], {next = NextBounce[next, r];
traj[[i]] = Append[traj[[i]], next];
running = Last[next];}], {i, 1, 2}];
(*Computation of differences-----*)
angles = Table[Transpose[Drop[Transpose[traj[[i]]], 2]], {i, 1, 2}];
aold = Abs[dangle];
diffangles = {{0., aold}};
diffpos = {{0., 0.}};
i1 = 2; i2 = 2;
While[{i1 ≤ Length[angles[[1]]]} && {i2 ≤ Length[angles[[2]]]},
{If[angles[[1, i1, 2]] < angles[[2, i2, 2]],
{diffangles = Append[diffangles, {angles[[1, i1, 2]], aold}];
aold = Abs[angles[[1, i1, 1]] - angles[[2, i2 - 1, 1]]];
diffangles = Append[diffangles, {angles[[1, i1, 2]], aold}];
diffpos =
Append[diffpos, {angles[[1, i1, 2]], Sqrt[(traj[[1, i1, 1]] - traj[[2, i2 - 1, 1]] -
(traj[[2, i2, 1]] - traj[[2, i2 - 1, 1]]) * (angles[[1, i1, 2]] - angles[[
2, i2 - 1, 2]]) / (angles[[2, i2, 2]] - angles[[2, i2 - 1, 2]])^2 +
(traj[[1, i1, 2]] - traj[[2, i2 - 1, 2]] - (traj[[2, i2, 2]] -
traj[[2, i2 - 1, 2]]) * (angles[[1, i1, 2]] - angles[[2, i2 - 1, 2]]) /
(angles[[2, i2, 2]] - angles[[2, i2 - 1, 2]])^2}];
i1 = i1 + 1;}, {diffangles = Append[diffangles, {angles[[2, i2, 2]], aold}];
aold = Abs[angles[[1, i1 - 1, 1]] - angles[[2, i2, 1]]];
diffangles = Append[diffangles, {angles[[2, i2, 2]], aold}];
diffpos =
Append[diffpos, {angles[[2, i2, 2]], Sqrt[(traj[[2, i2, 1]] - traj[[1, i1 - 1, 1]] -
(traj[[1, i1, 1]] - traj[[1, i1 - 1, 1]]) * (angles[[2, i2, 2]] - angles[[
1, i1 - 1, 2]]) / (angles[[1, i1, 2]] - angles[[1, i1 - 1, 2]])^2 +
(traj[[2, i2, 2]] - traj[[1, i1 - 1, 2]] - (traj[[1, i1, 2]] -

```

```

      traj[[1, i1 - 1, 2]] * (angles[[2, i2, 2]] - angles[[1, i1 - 1, 2]]) /
      (angles[[1, i1, 2]] - angles[[1, i1 - 1, 2]])^2}}];
    i2 = i2 + 1; ]];
  (*Representation of trajectories-----*) g1 = Graphics[{Thickness[0.01],
    Line[{{-1, -1}, {-1, 1}, {1, 1}, {1, -1}, {-1, -1}}, Circle[{0, 0}, r]}}];
  g2 = Table[Graphics[{Thickness[0.005], Line[
    Transpose[Take[Transpose[traj[[i]]], 2]]]], {i, 1, 2}];
  Show[Prepend[g2, g1], AspectRatio -> 1, ImageSize -> 200]
  (*Representation of angle differences-----*)

  Show[Graphics[{Thickness[0.005], Line[diffangles]}], Axes -> True, Frame -> True,
    AxesLabel -> {"Time", "Difference of angles"}, AspectRatio -> 1, ImageSize -> 200]
  (*Representation of position differences-----*)

  Show[Graphics[{Thickness[0.005], Line[diffpos]}], Axes -> True, Frame -> True,
    AxesLabel -> {"Time", "Difference of positions"}, AspectRatio -> 1, ImageSize -> 200] ];

End[]

EndPackage[]

```

Out[160]= UVW`Billiard`

Out[161]=

NextBounce[currentposition,r] computes the next bouncing point of a ball inside a square table with a circular obstacle of radius r. The initial conditions are given in the list currentposition that has four real coordinates, respectively the abscissa, ordinate, incoming angle of the ball, and the current time. The result is returned as another list of four elements, the abscissa and ordinate of the new bouncing point, the new direction of the ball and the current time incremented by the running time between the two bounces.

Out[162]=

Trajectory[shootingangle,tmax,r] draws a square table with a centered circular obstacle of radius r (default 0.5). Draws inside this support the trajectory of a ball starting at the bottom left corner with initial direction shootingangle, up to time tmax.

Out[163]=

Bundle[listofangles,tmax,r] draws a square table with a centered circular obstacle of radius r. Draws inside the trajectories of balls starting at the bottom left corner with initial directions read in listofangles, up to time tmax.

Out[164]=

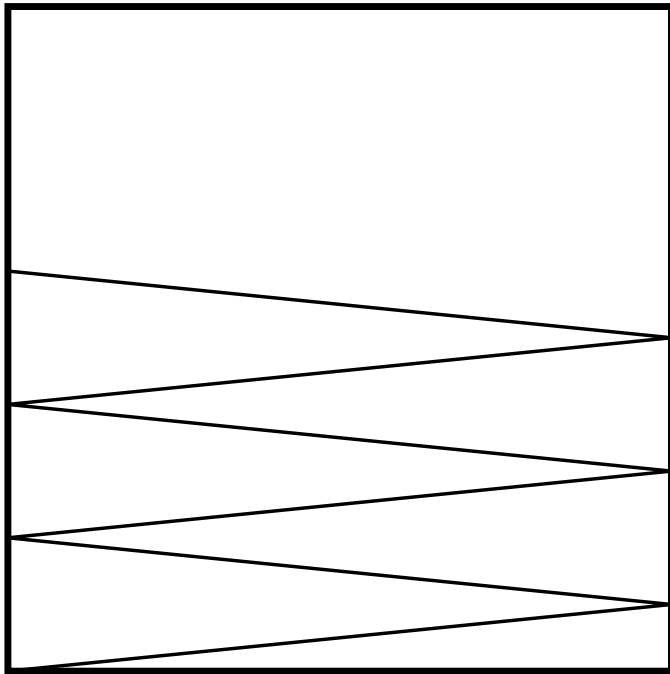
`Diff2[angle,dangle,tmax,r]` simulates two trajectories of balls in a square table with a centered circular obstacle with radius r . Both trajectories start from the bottom left corner. They are followed up to time $tmax$. The shooting angle of the first trajectory is $angle$, its difference with the second shooting angle is $dangle$. The function represents three consecutive graphics. The first one is the billiard table with the two trajectories. The second one is the evolution of the absolute difference of angles as a function of time. The third one is the norm of the difference of positions as a function of time.

Out[165]= UVW`Billiard`Private`

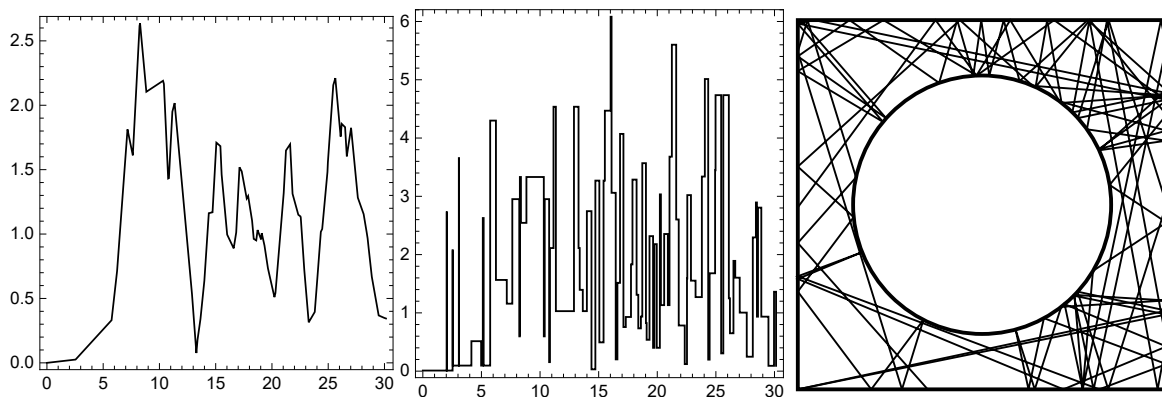
Out[170]= UVW`Billiard`Private`

In[172]:= **Trajectory[.1, 12, 0]**

Out[172]=

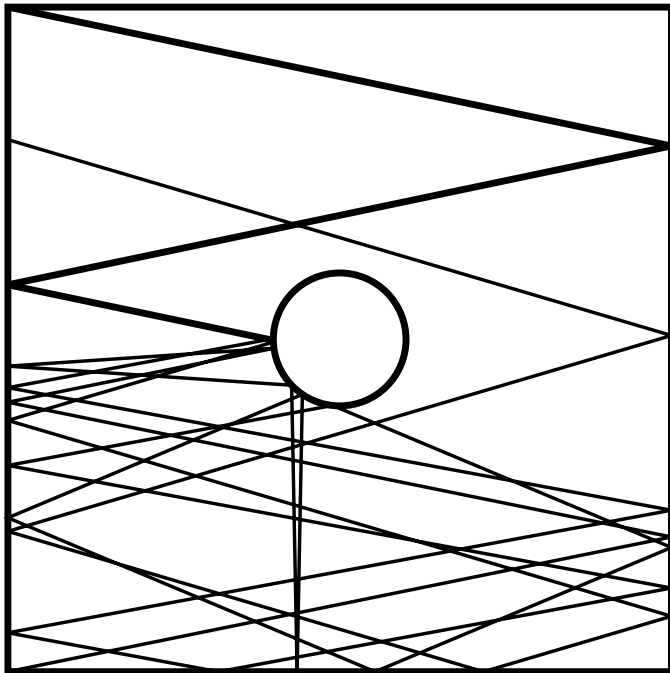
In[173]:= **Diff2[.2, .01, 30, .7]**

Out[173]=



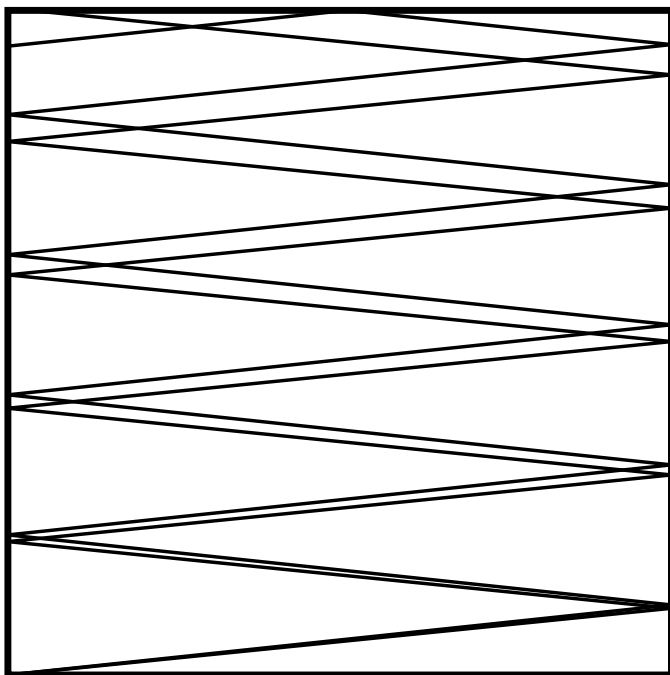
In[174]:= **Trajectory**[.2, 40, .2]

Out[174]=



In[175]:= **Bundle**[{.1, .105}, 20, 0]

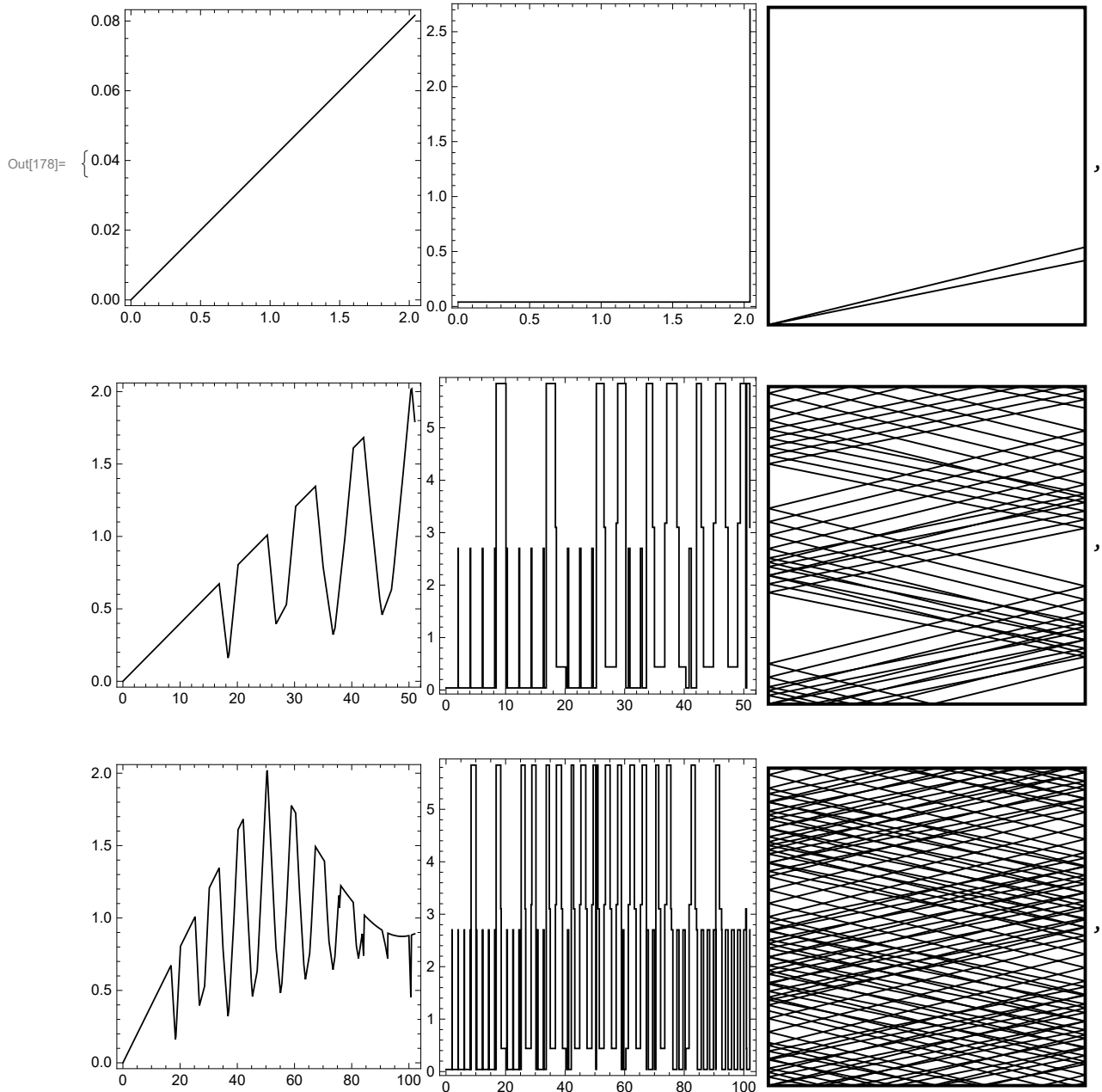
Out[175]=

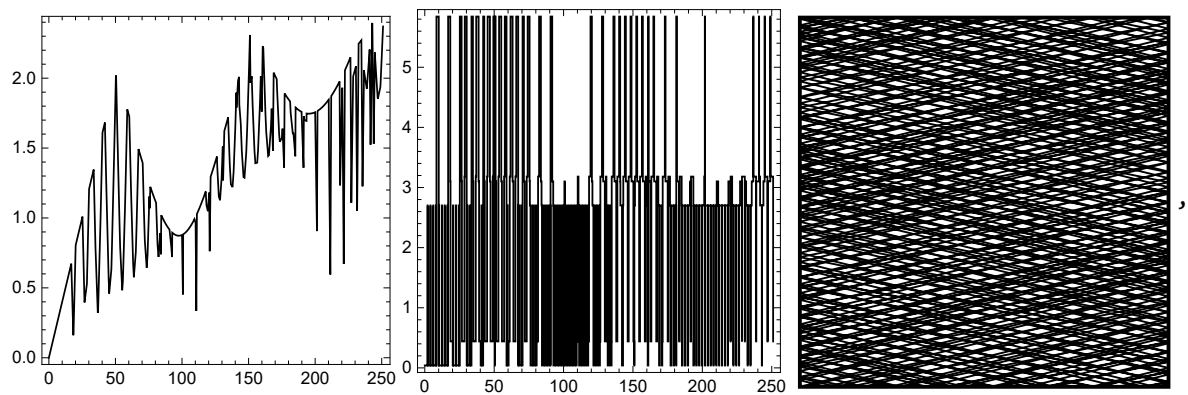
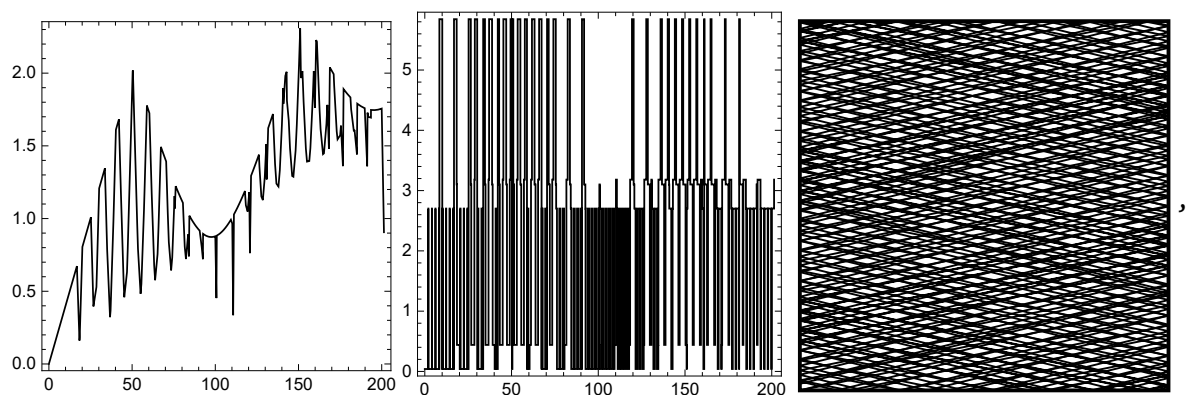
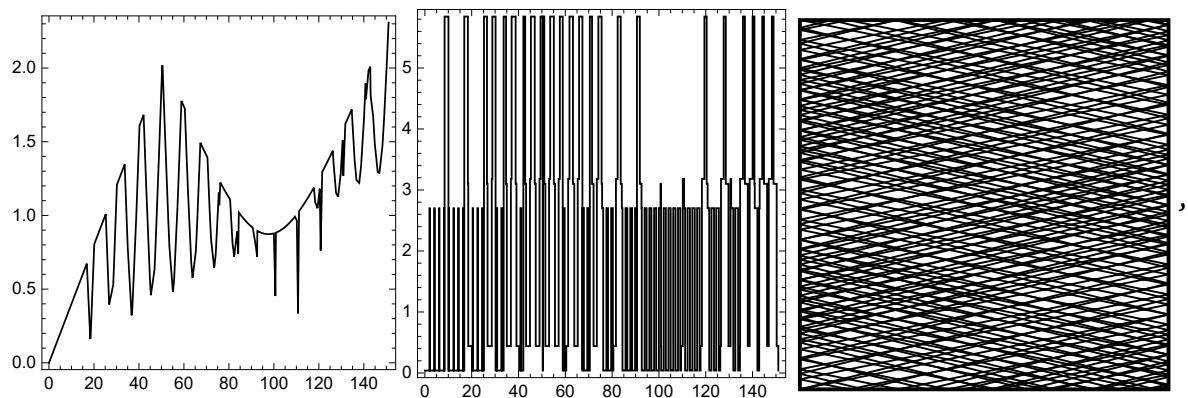


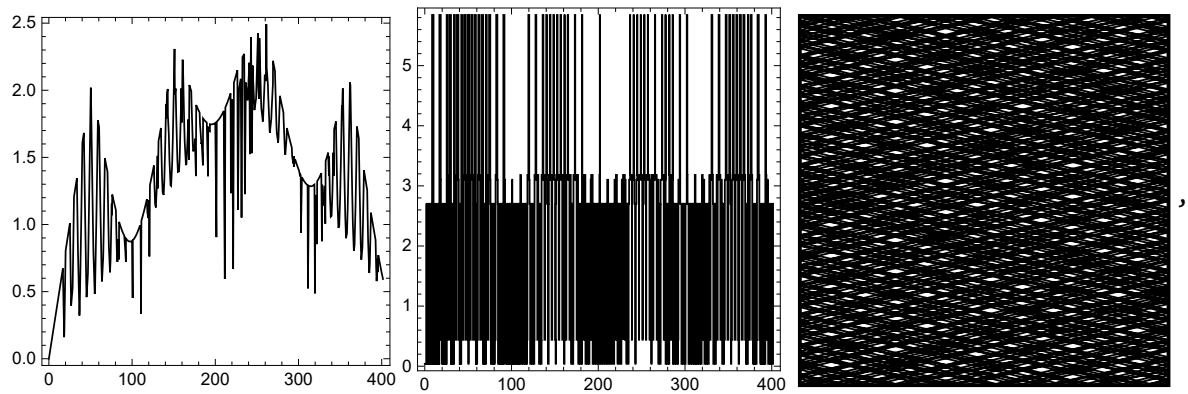
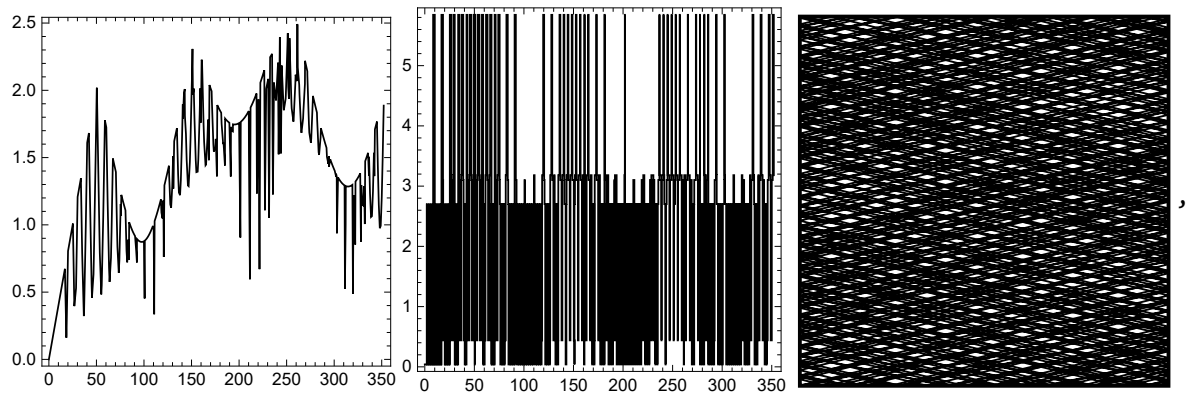
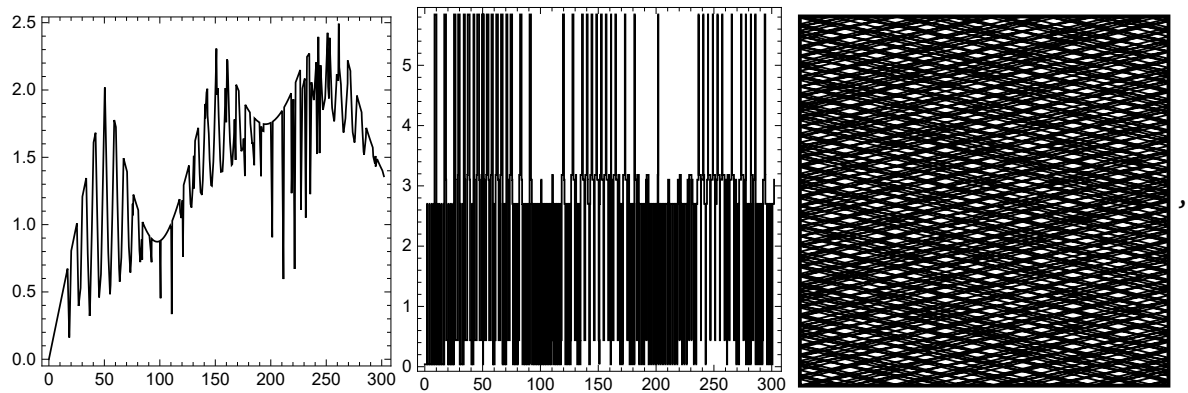
In[176]:=

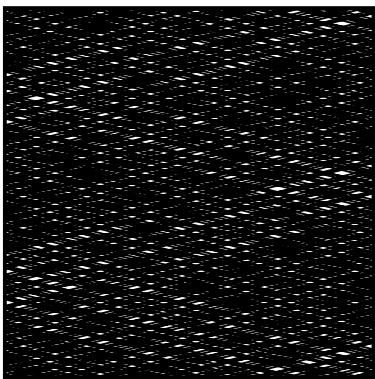
```
(* I will analyze spacing between two objects using Diff2 by varying tmax *)
graphs = {};(* Make a list to store the graphs from Diff2 *)

(* This loop appends the results of a Diff2 function call with constant angle,
dangle, and no
object in the middle. Tmax varies from 1 to 500, increasing by 50 *)
For[i = 1, i < 500, i = i + 50, AppendTo[graphs, Diff2[.2, .04, i, 0]]];
```

In[178]:= **graphs**







```
(* As time increases, it appears that the difference in distance
between the two objects is more random (the first graph). My goal
is to try and make this first graph as random as possible in
order for it to somehow be used to generate a random number.
So far, I'm concluding that increasing time will
increase the randomness of this first graph *)
```

```
(* I will analyze varying the size of the object,
while also varying tmax. I will use a 3 sizes, and 3 times *)
graphs2 = {}
```

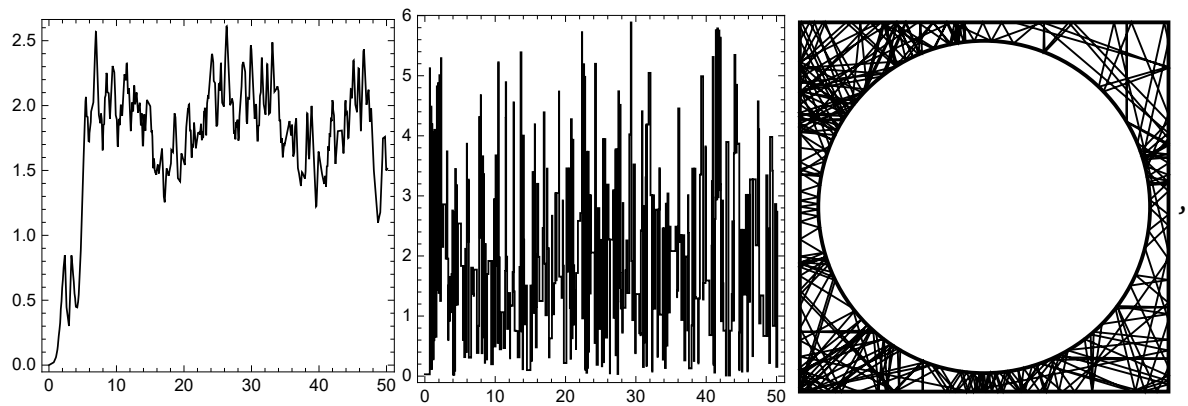
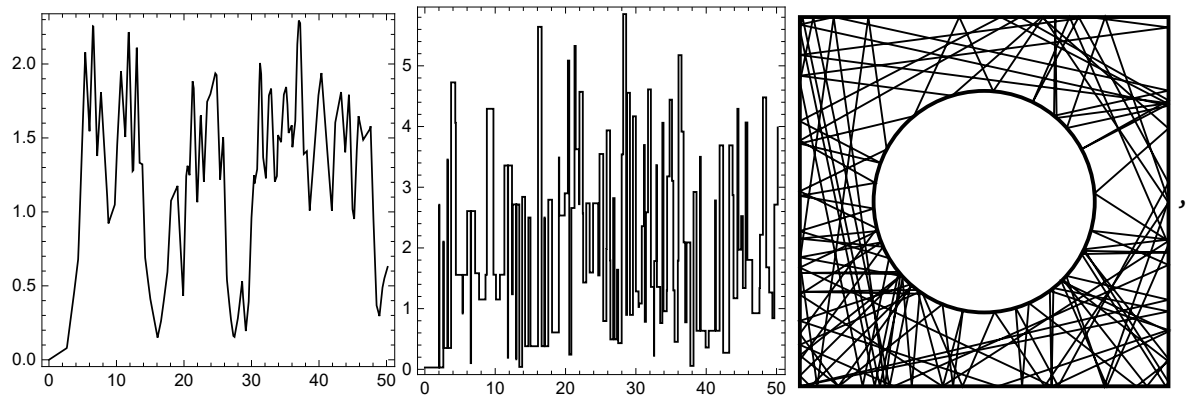
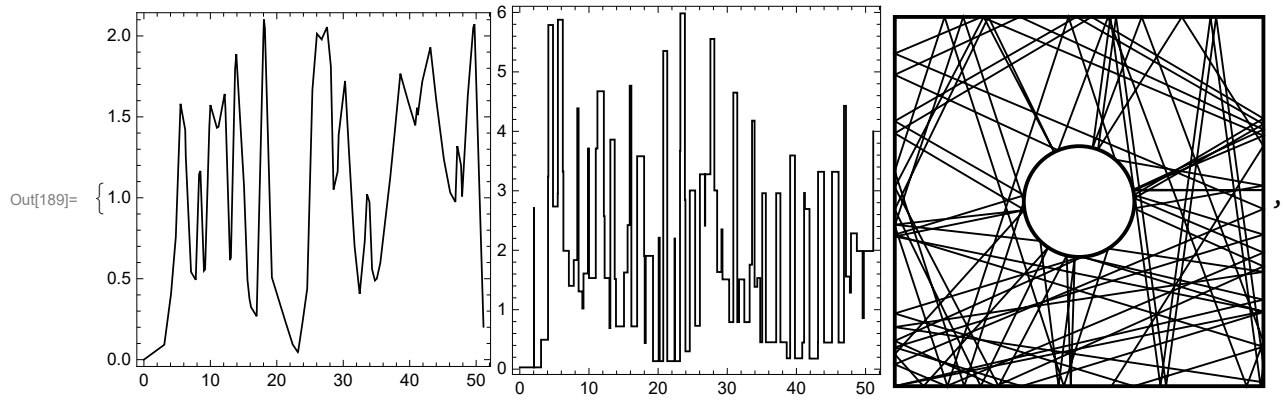
```
(* Append 9 graphs of consisent angle and dangle. I varied tmax as either 50,
100 or 150. And
the side of the boject from .3 .6 and .9. *)
AppendTo[graphs2, Diff2[.2, .03, 50, .3]];
```

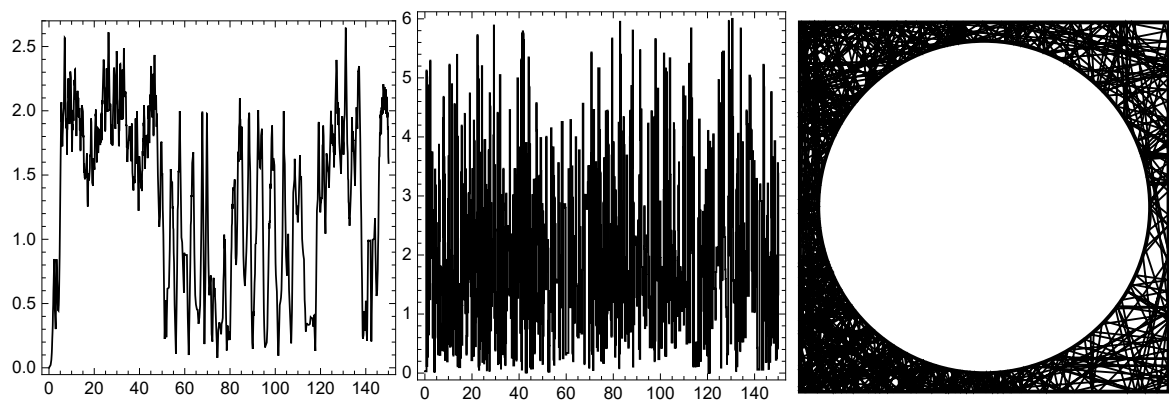
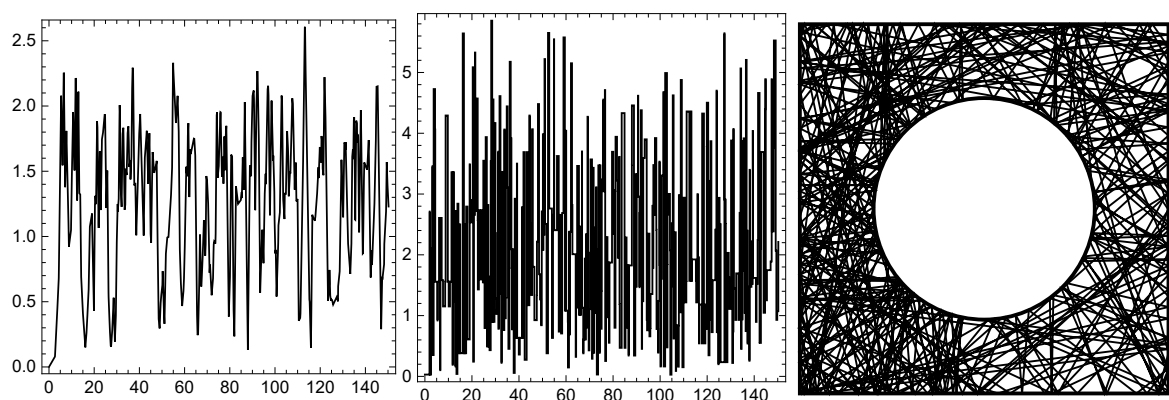
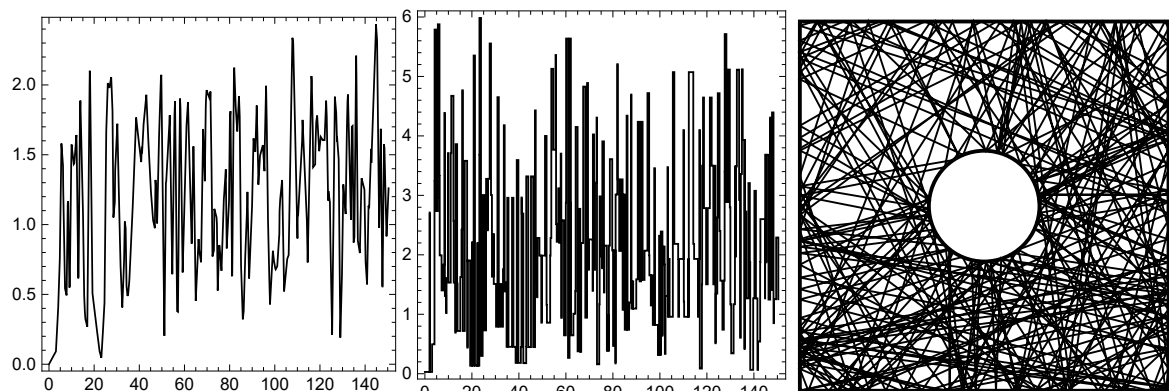
Out[179]= $\{\}$

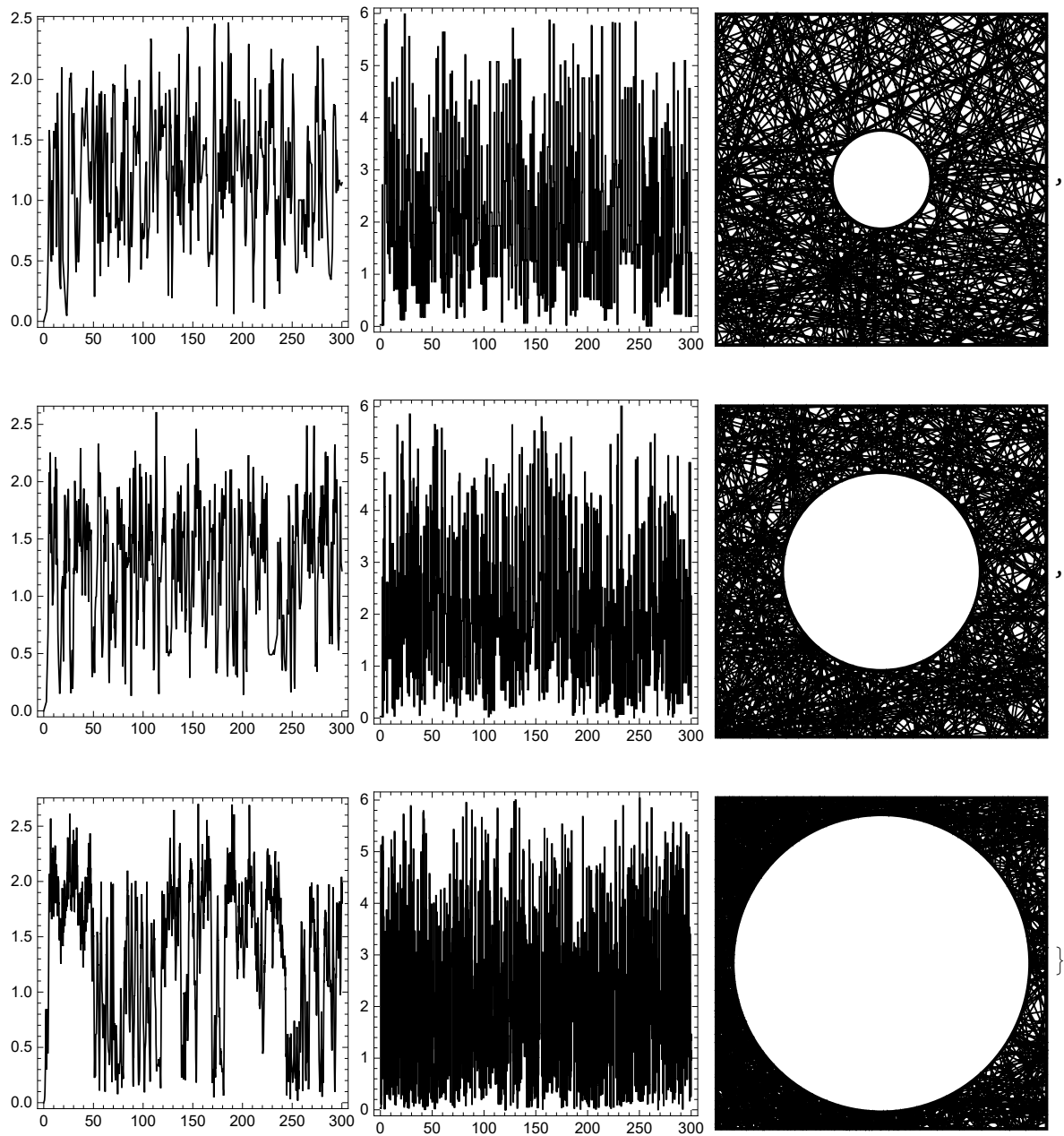
```
AppendTo[graphs2, Diff2[.2, .03, 50, .6]];
AppendTo[graphs2, Diff2[.2, .03, 50, .9]];
AppendTo[graphs2, Diff2[.2, .03, 150, .3]];
AppendTo[graphs2, Diff2[.2, .03, 150, .6]];
AppendTo[graphs2, Diff2[.2, .03, 150, .9]];
AppendTo[graphs2, Diff2[.2, .03, 300, .3]];
AppendTo[graphs2, Diff2[.2, .03, 300, .6]];
AppendTo[graphs2, Diff2[.2, .03, 300, .9]];

```

graphs2







```

In[190]:= (* The rightmost column is the trials using the large object. These trials,
by looking at the first graph,
don't appear to have a random graph. I believe this is because the object is so big,
that the two objects will get "stuck" on
the same side and bounce around. Their distances won't be
far from each other. This is shown because the first graph has large
groupings. *)
(* In my opinion, the trials with a medium object have
a more equal 2nd graph for absolute distances of angles. I will
consider the medium object as an "optimal" object size,
but a small object would not be a bad choice either. *)

```

```

(* I will now vary angle and time, medium object is constant,
small, medium, and large for both angle and time. I expect
whatever result is optimal will have a large tmax,
because I concluded before that tmax yeilded a more random first graph. *)
graphs2 = {}

```

```
Out[190]= {}
```

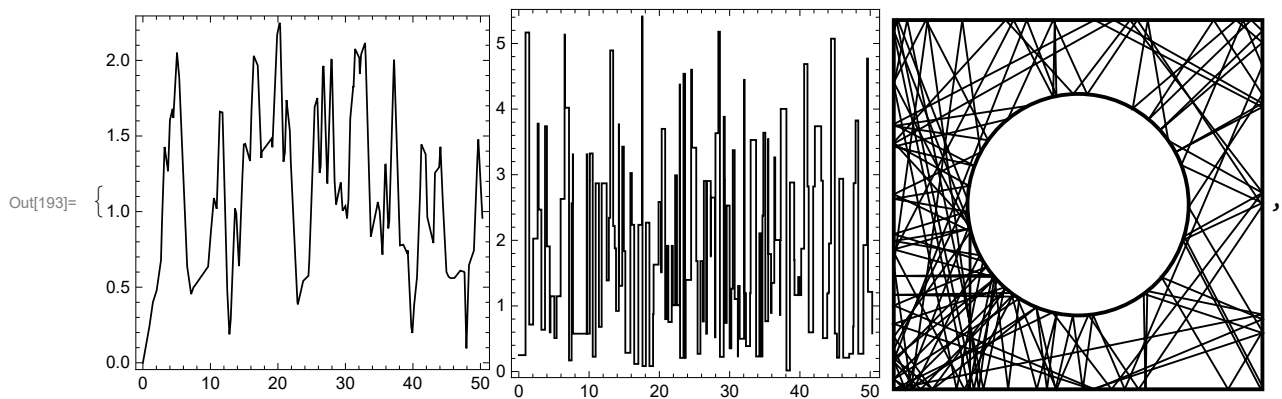
```
In[191]:= time = 50;
```

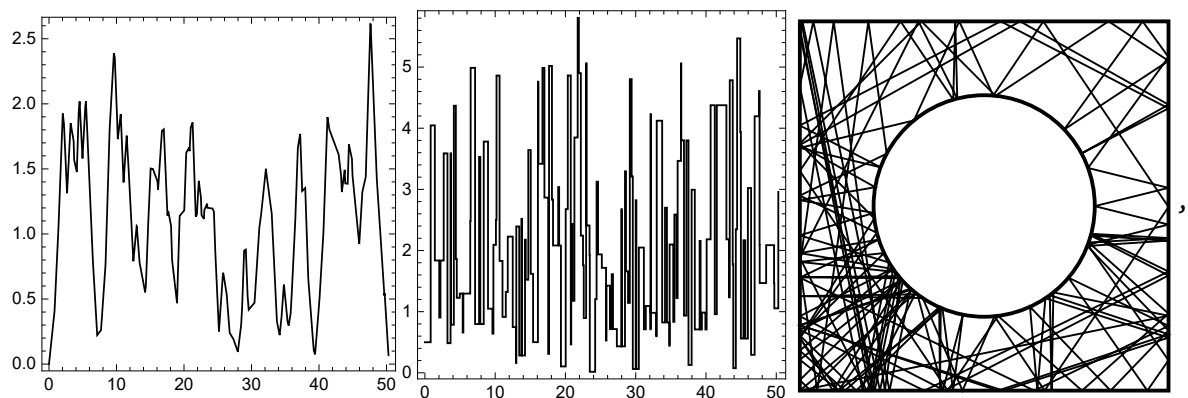
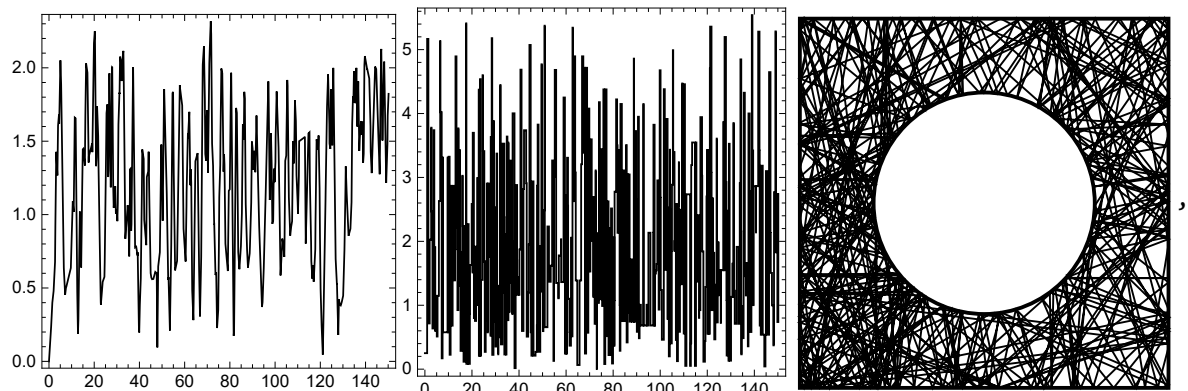
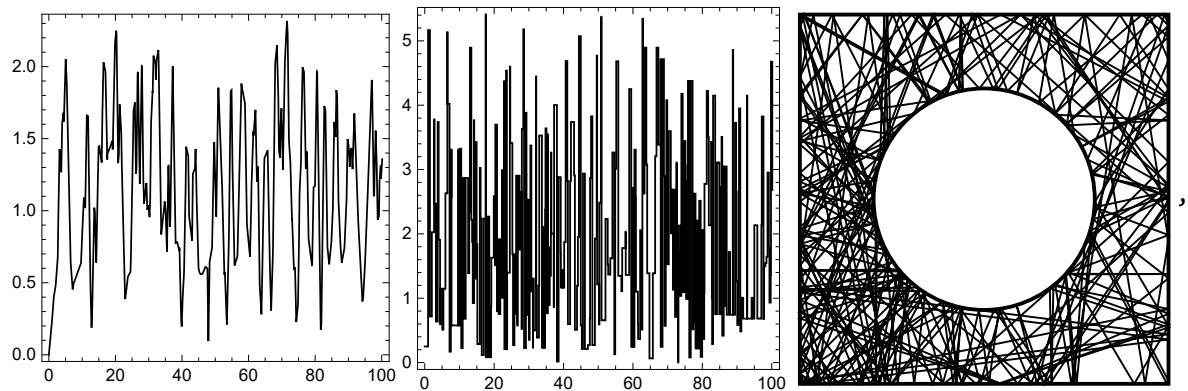
```

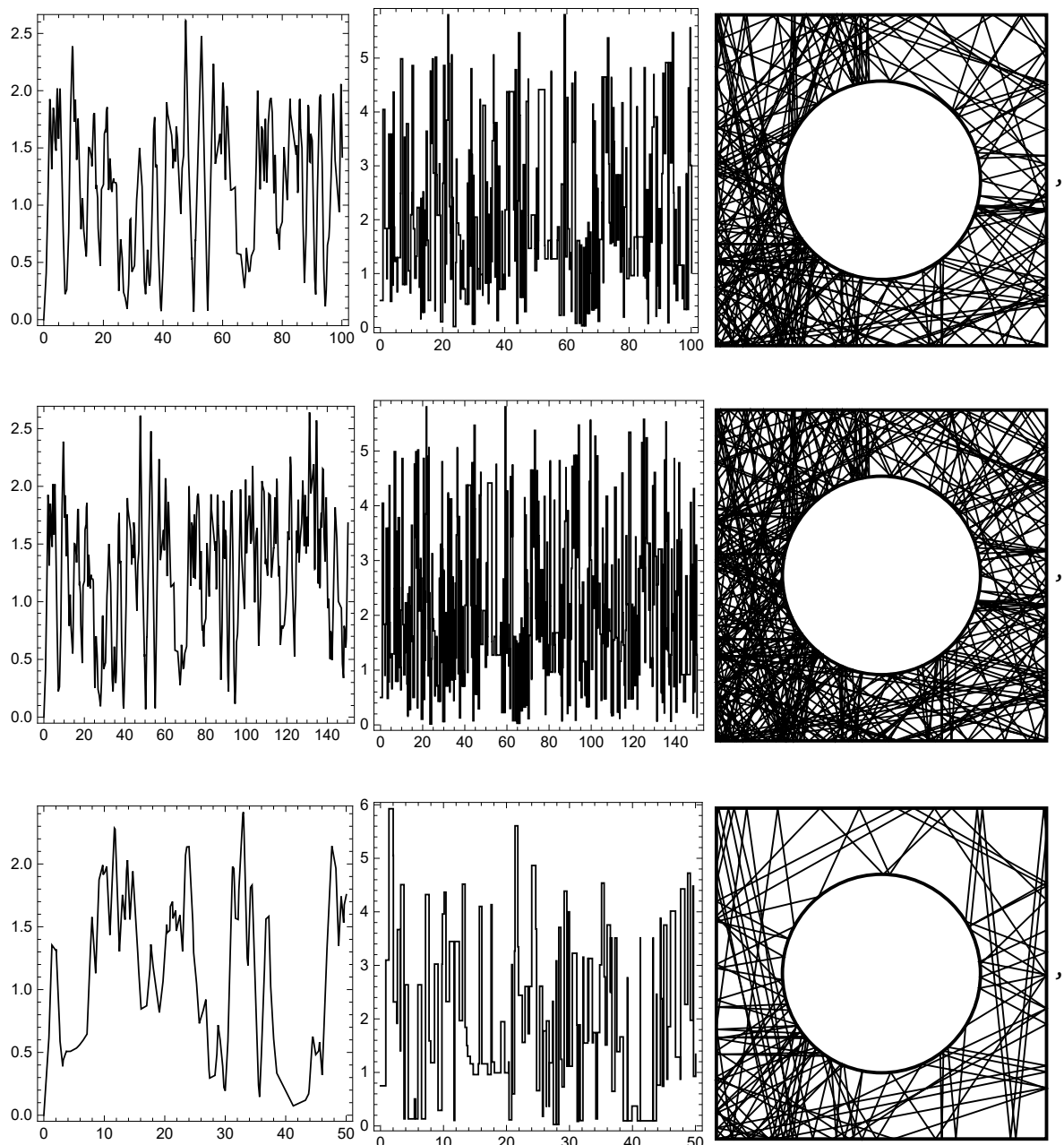
In[192]:= (* Append 9 graphs. 3 different times with 3 different dangle. dangle varies from .25,
.5, .75 while time varies from 50, 100, 150. *)
For[i = 1, i < 4, i++,
AppendTo[graphs2, Diff2[.2, i/4, time * 1, .6]];
AppendTo[graphs2, Diff2[.2, i/4, time * 2, .6]];
AppendTo[graphs2, Diff2[.2, i/4, time * 3, .6]];
];

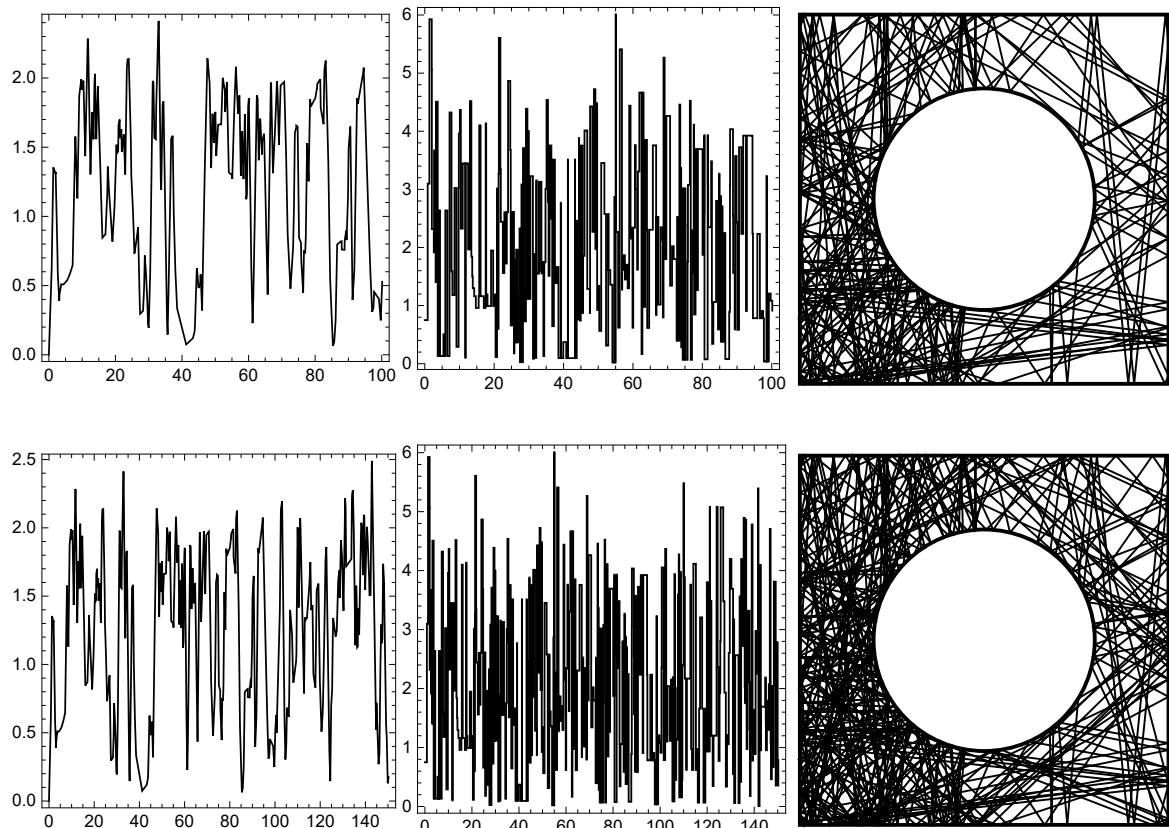
```

```
In[193]:= graphs2
```









```
In[194]:= (* Conclusions so far: A greater angle will produce more
            random results with no object in the middle. This is obvious. *)
            (* From these graphs, we can see that each column has pretty random
            data for its first graph, and a pretty equal spacing
            for the 2nd graph, the absolute difference of angles. Therefore,
            dangle does not matter. *)
            (* Greater time will produce more random results for the first graph,
            as we can see from our output. *)
            (* In summation: With an object in the middle,
            greater time will produce more random results, and the angle does not matter *)

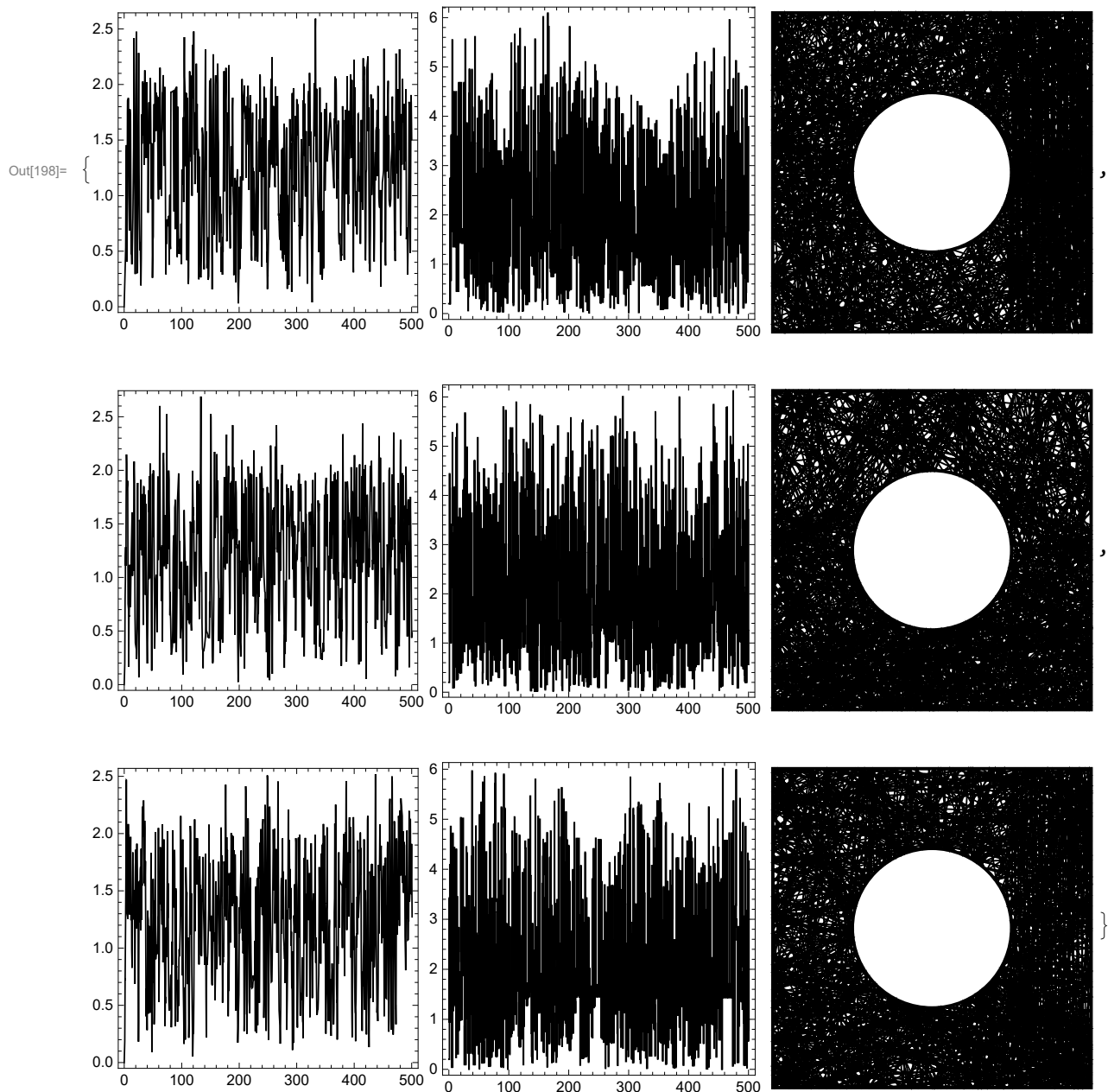
            (* So potentially use this simulation as a random number generator,
            we need a large tmax, a medium or small object in the middle,
            with any dangle. Does initial angle make a difference? *)

            (* I will analyze 3 trials, a small, medium, and large initial angle,
            with the rest of the parameters held constant. *)
```

```
graphs3 = {}
AppendTo[graphs3, Diff2[.1, .2, 500, .5]];
AppendTo[graphs3, Diff2[.4, .2, 500, .5]];
AppendTo[graphs3, Diff2[.7, .2, 500, .5]];
```

Out[194]= {}

In[198]:= **graphs3**



(* Conclusion: The initial angle makes no difference. The 2nd graph of absolute angle differences appears uniform across all graphs.
The first graph appears random across the whole time from 0 to tmax on all graphs *)

(* Conclusion: We need a large tmax, medium or small object in the middle, any dangle, and any initial angle. Obviously dangle cannot be 0 and initial angle cannot be 0 or 1 *)