In[42]:=
```
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 absissa, 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[42]= UVW`Billiard`

Out[43]=

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 absissa, 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[44]=

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[45]=

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: Symbol Diff2 appears in multiple contexts {UVW`Billiard`, Global`}; definitions in context UVW`Billiard` may shadow or be shadowed by other definitions.

Out[46]=

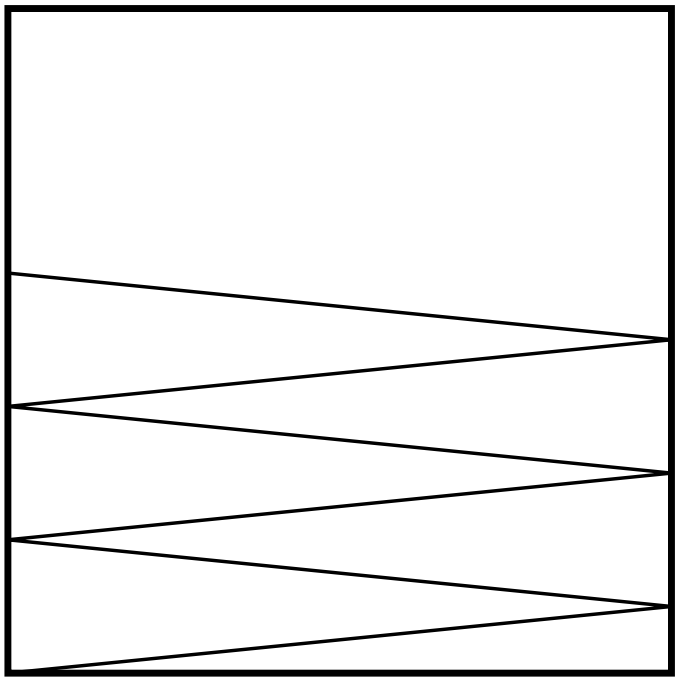
Diff2[angle,dangle,tmax,r] simulates two trajectories of balls in a square table with a centered circular obstacle with radius r. Both trajectories start from the bottom left corner. They are followed up to time tmax. The shooting angle of the first trajectory is angle, its difference with the second 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[47]= UVW`Billiard`Private`

Out[52]= UVW`Billiard`Private`

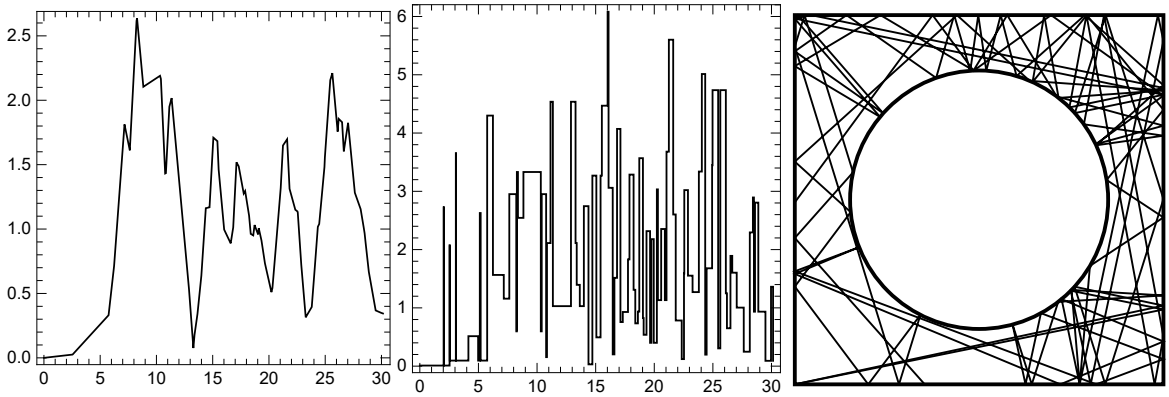
In[54]:= **Trajectory[.1, 12 , 0]**

Out[54]=



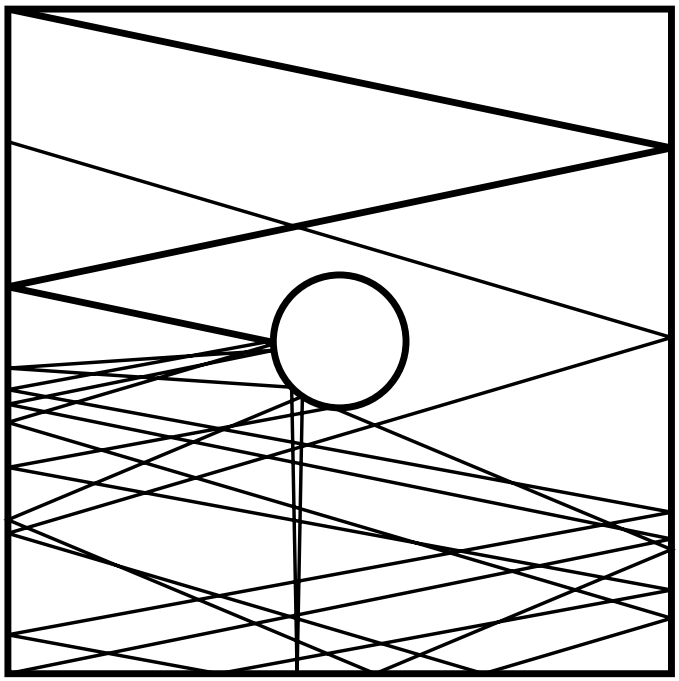
In[55]:= **Diff2[.2, .01, 30, .7]**

Out[55]=



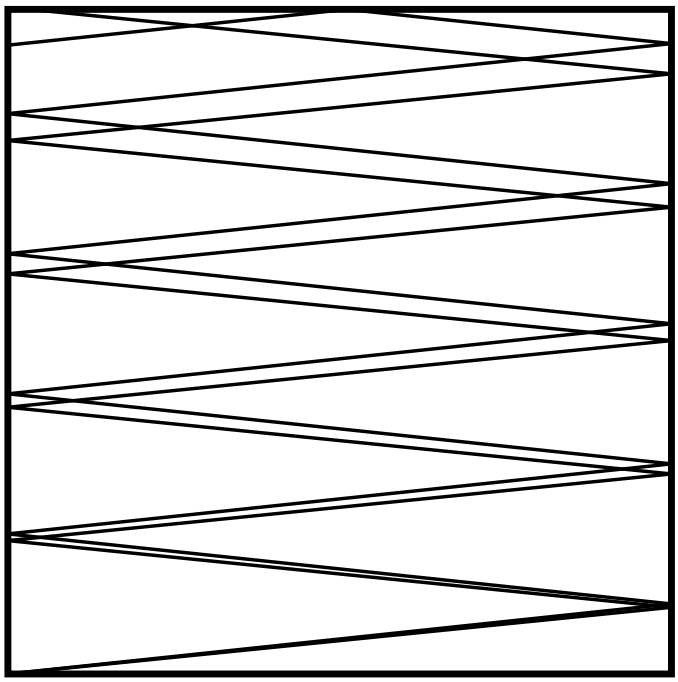
In[56]:=
```
Trajectory[.2, 40, .2]
```

Out[56]=



In[57]:=
```
Bundle[{.1, .105}, 20, 0]
```

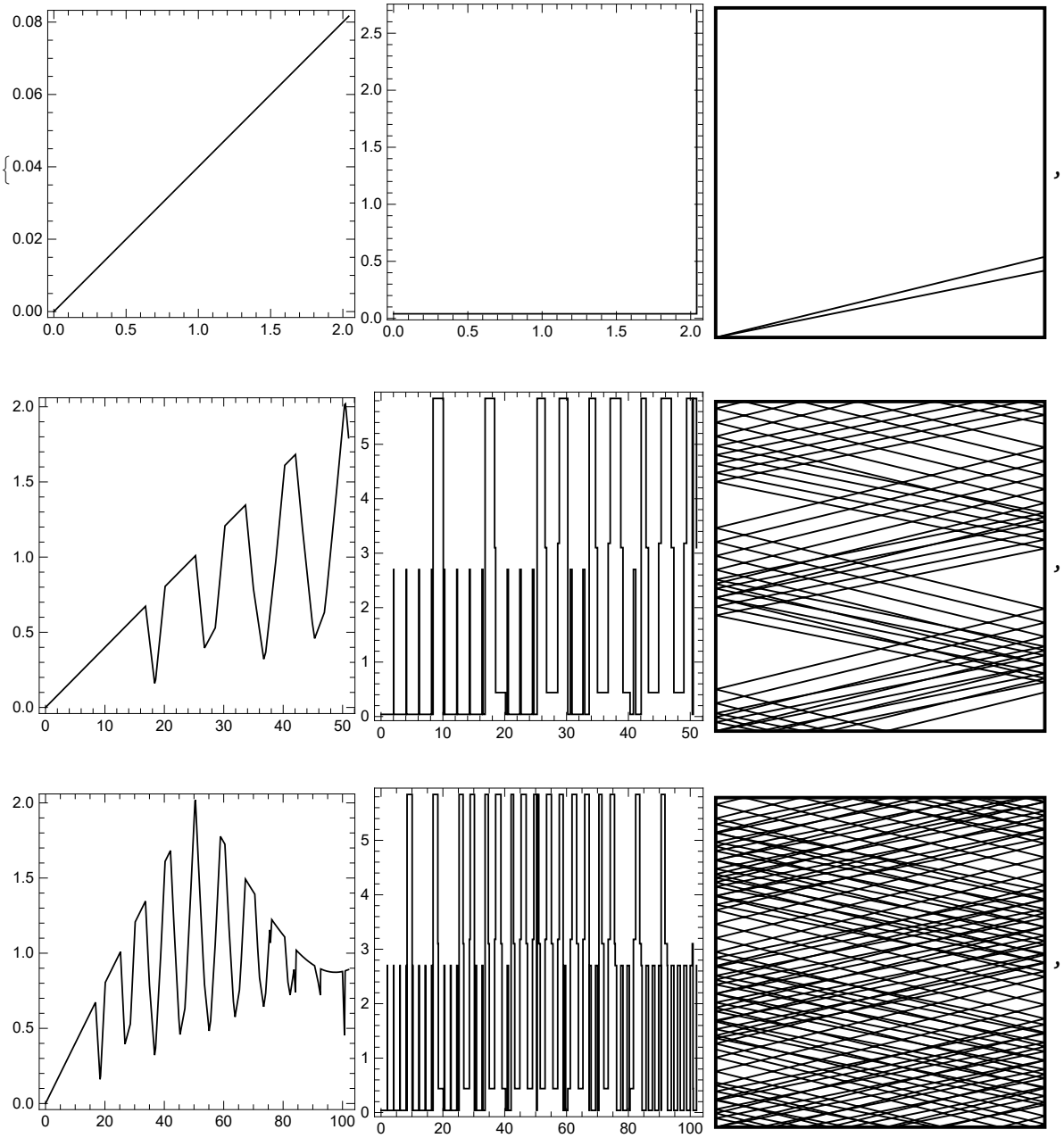
Out[57]=

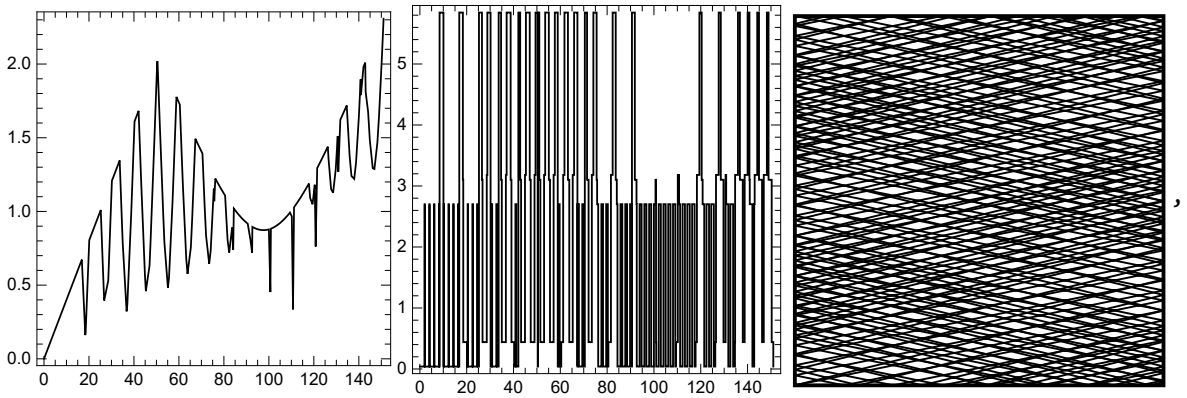


In[58]:=
```
(* I will analyze spacing between two objects using Diff2 by varying tmax *)
graphs = {};
For[i = 1, i < 500, i = i + 50, AppendTo[graphs, Diff2[.2, .04, i, 0]]];
```

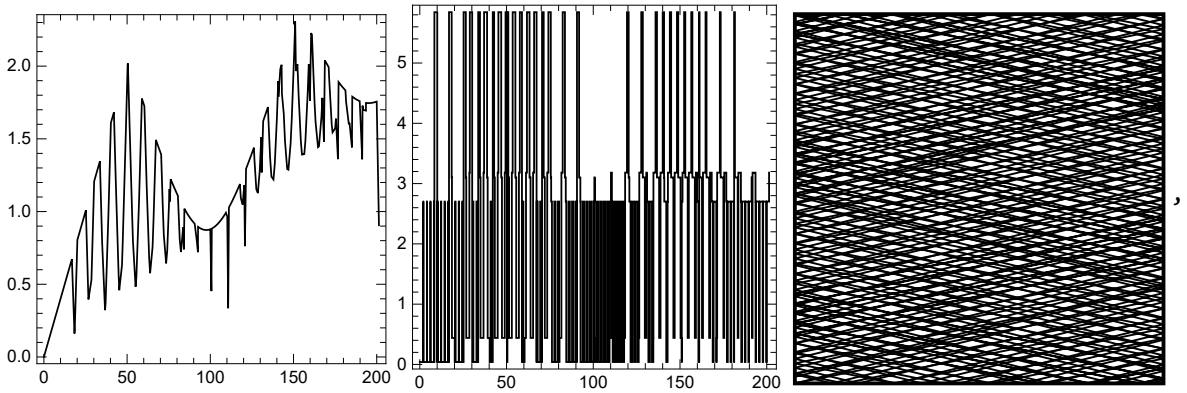
In[60]:= **graphs**

Out[60]=

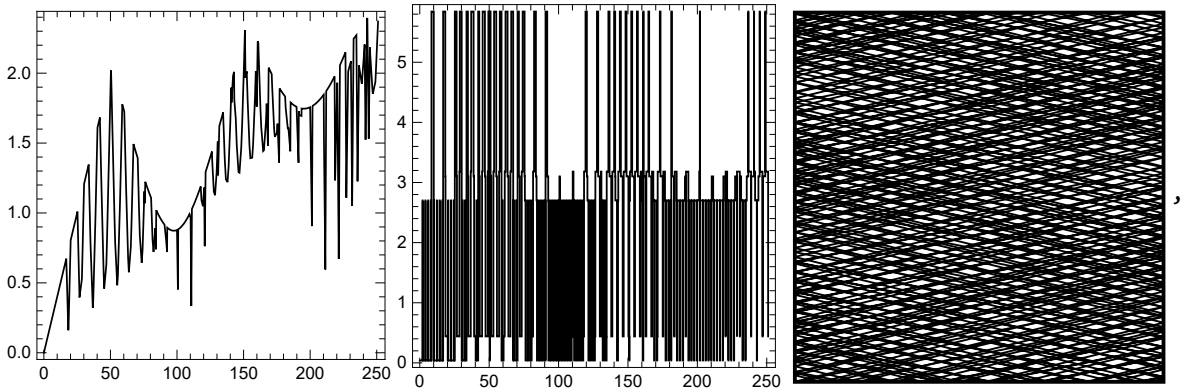



0.0
0.5
1.0
1.5
2.0
0 20 40 60 80 100 120 140
0 1 2 3 4 5
0 20 40 60 80 100 120 140

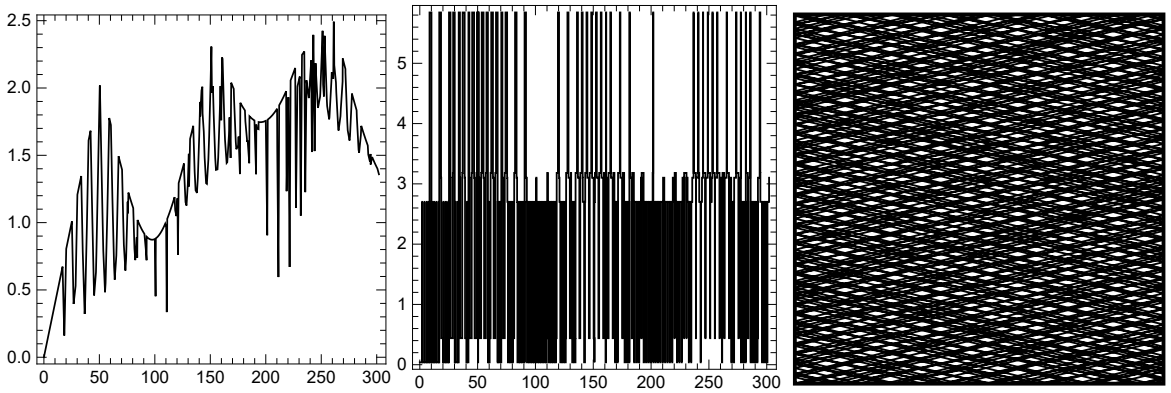
,


0.0
0.5
1.0
1.5
2.0
0 50 100 150 200
0 1 2 3 4 5
0 50 100 150 200

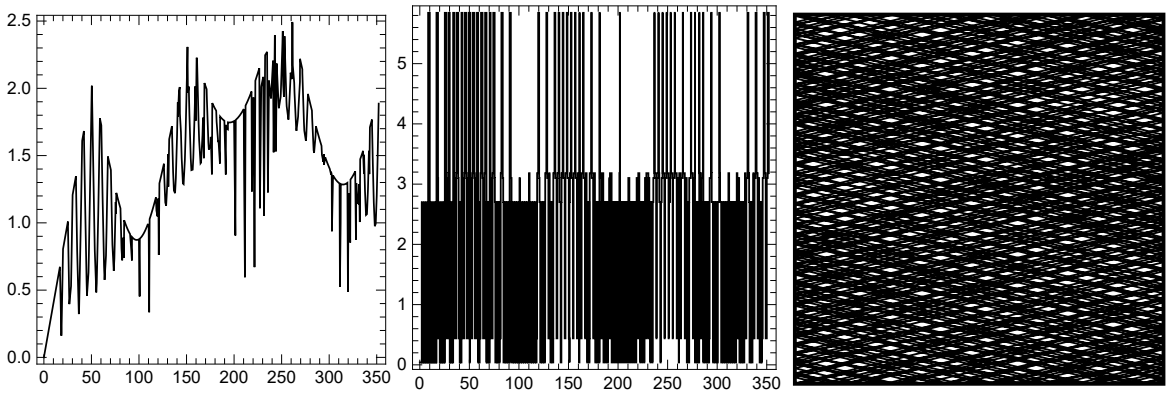
,


0.0
0.5
1.0
1.5
2.0
0 50 100 150 200 250
0 1 2 3 4 5
0 50 100 150 200 250

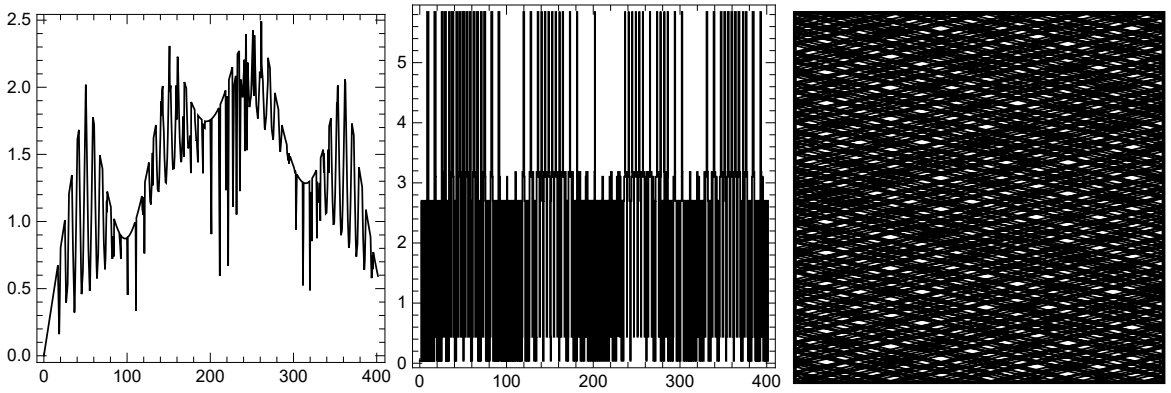
,


0.0
0.5
1.0
1.5
2.0
2.5
0
50
100
150
200
250
300
0
1
2
3
4
5

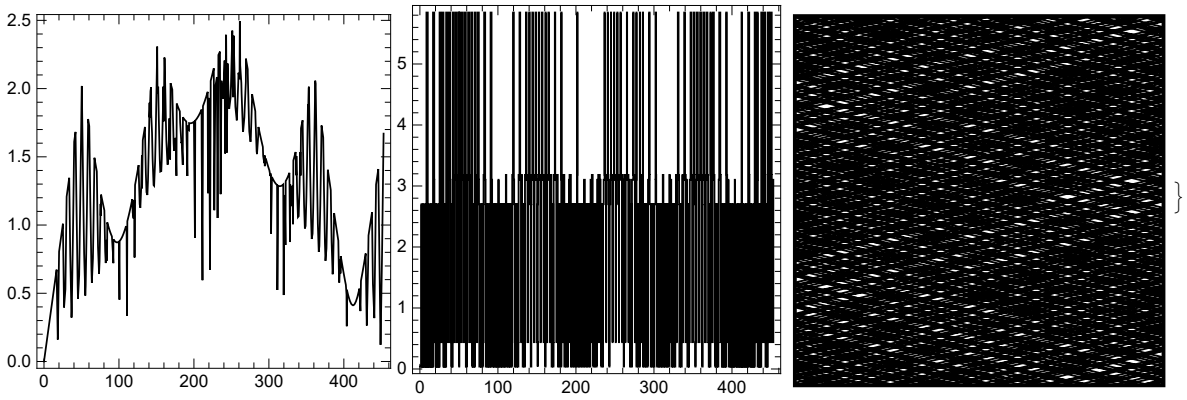
,


0.0
0.5
1.0
1.5
2.0
2.5
0
50
100
150
200
250
300
350
0
1
2
3
4
5

,


0.0
0.5
1.0
1.5
2.0
2.5
0
100
200
300
400
0
1
2
3
4
5

,



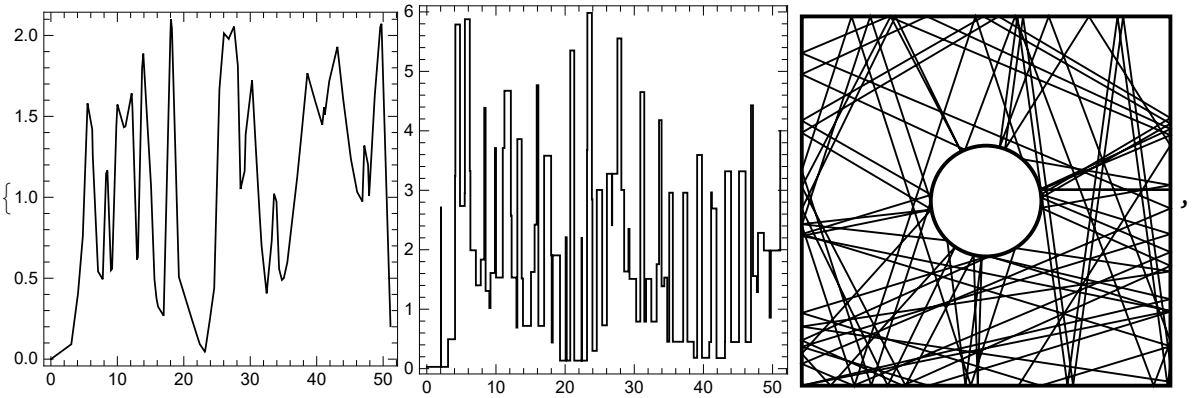
In[61]:=
```
(* I will analyze varying the size of the object,
while also varying tmax. I will use a 3 sizes, and 3 times *)
graphs2 = {}
AppendTo[graphs2, Diff2[.2, .03, 50, .3]];
```

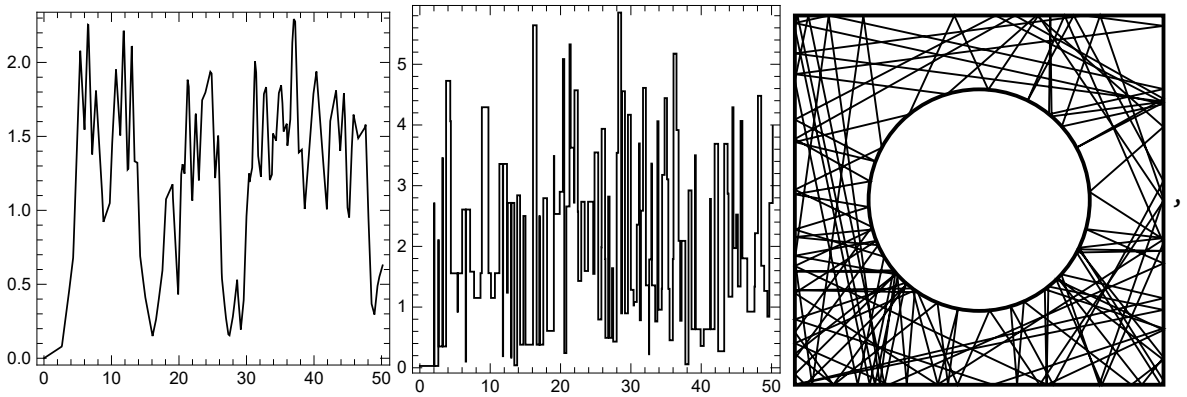
Out[61]= { }

In[63]:=
```
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]];
```

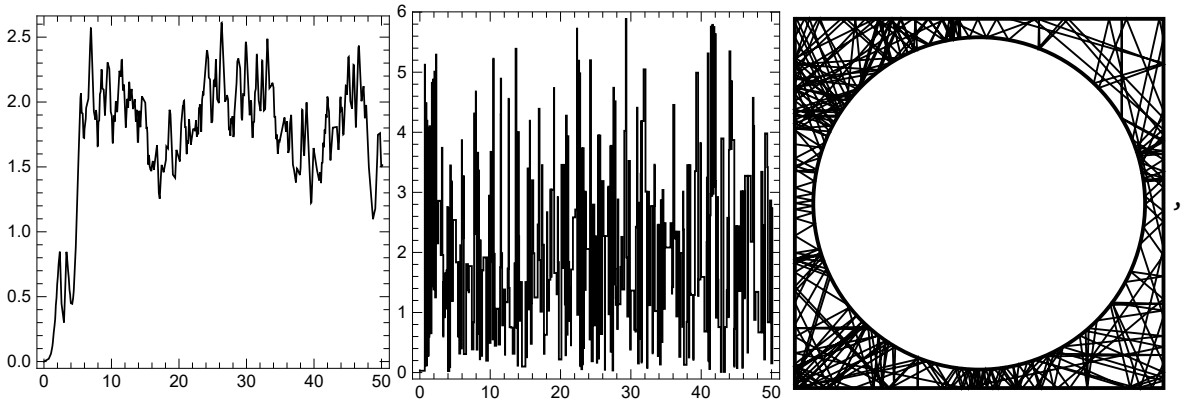
In[71]:=
```
graphs2
```

Out[71]=

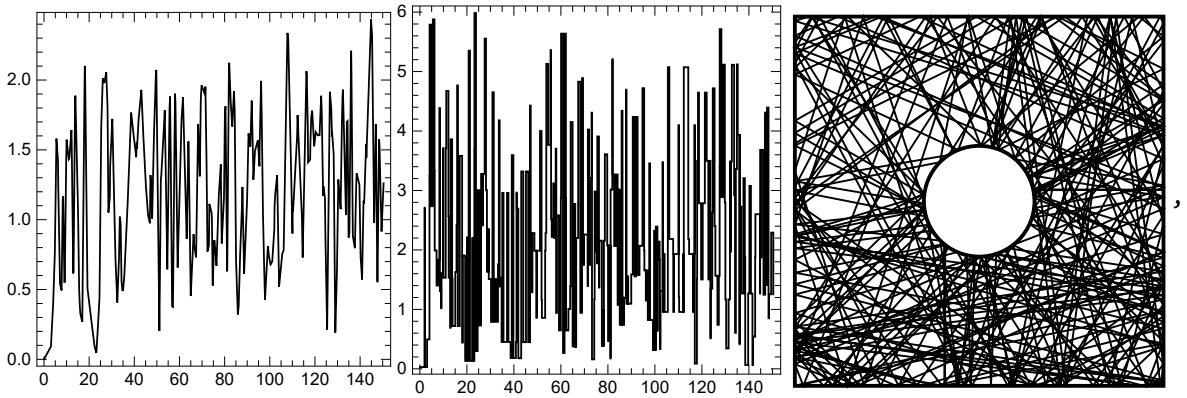



0.0
0.5
1.0
1.5
2.0
0
10
20
30
40
50
0
1
2
3
4
5

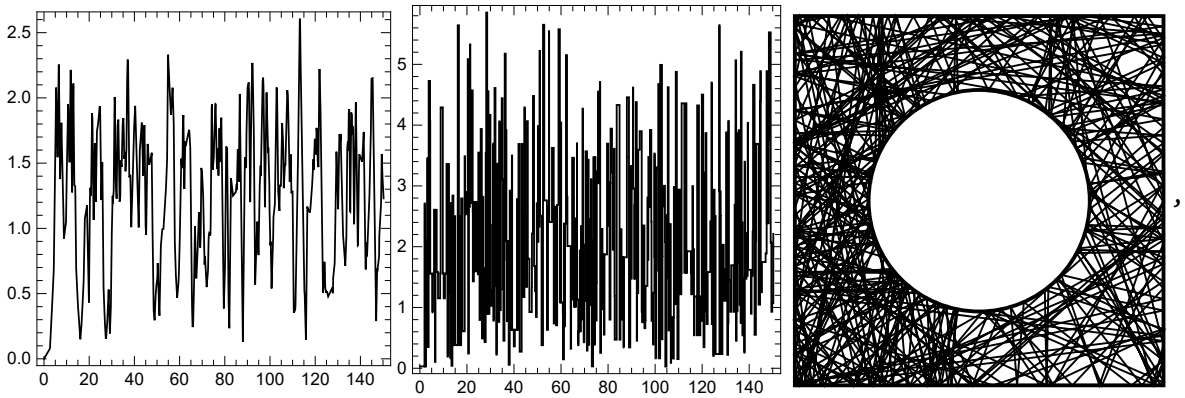
,


0.0
0.5
1.0
1.5
2.0
2.5
0
10
20
30
40
50
0
1
2
3
4
5
6

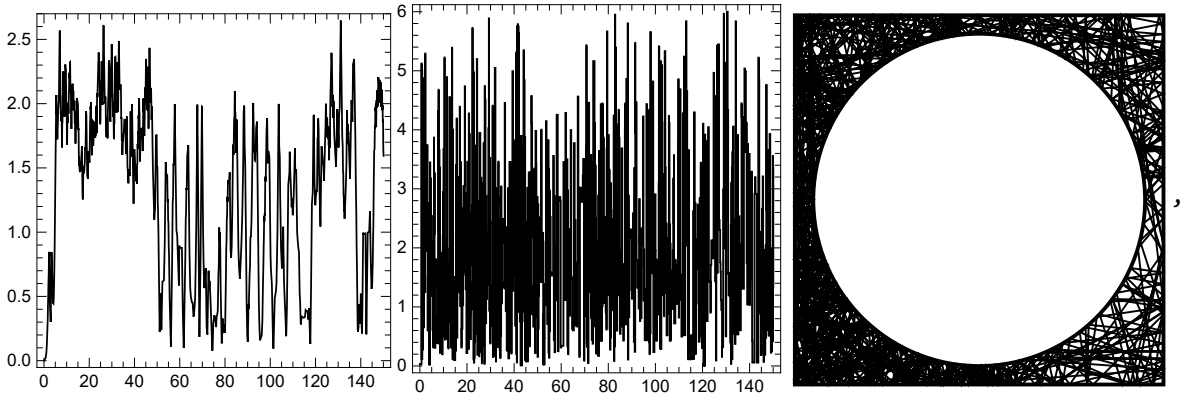
,


0.0
0.5
1.0
1.5
2.0
0
20
40
60
80
100
120
140
0
1
2
3
4
5
6

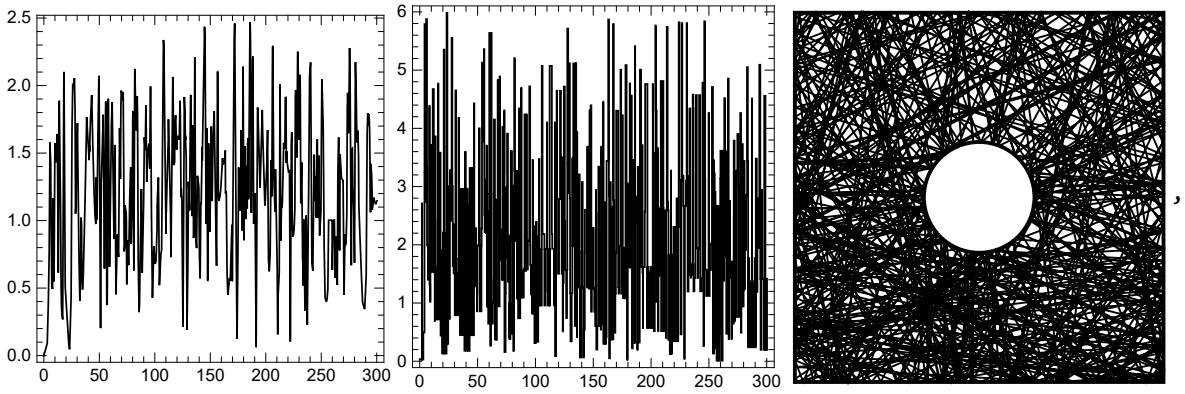
,


0.0
0.5
1.0
1.5
2.0
2.5
0 20 40 60 80 100 120 140
0 1 2 3 4 5
0 20 40 60 80 100 120 140

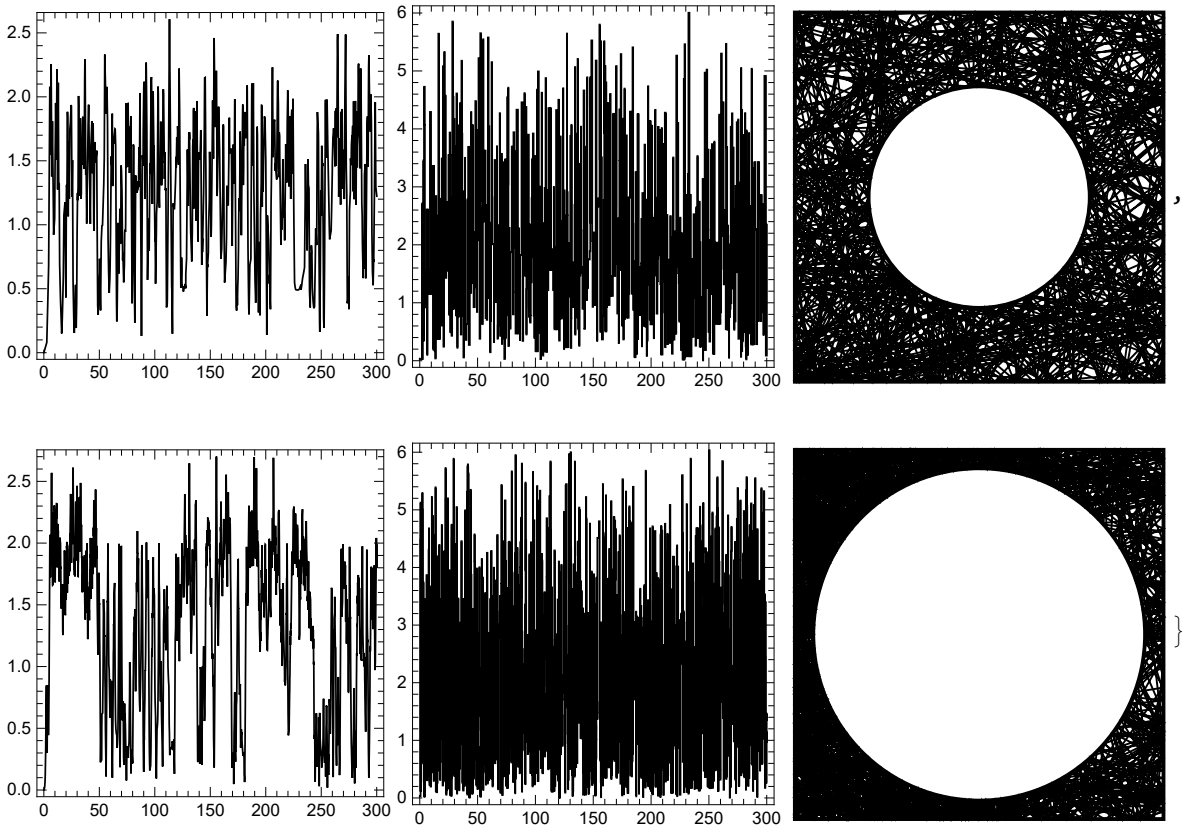
,


0.0
0.5
1.0
1.5
2.0
2.5
0 20 40 60 80 100 120 140
0 1 2 3 4 5 6
0 20 40 60 80 100 120 140

,


0.0
0.5
1.0
1.5
2.0
2.5
0 50 100 150 200 250 300
0 1 2 3 4 5 6
0 50 100 150 200 250 300

,



In[82]:=
```
(* I will now vary angle and time, medium object,
small, medium, and large for both angle and time *)
graphs2 = {}
```

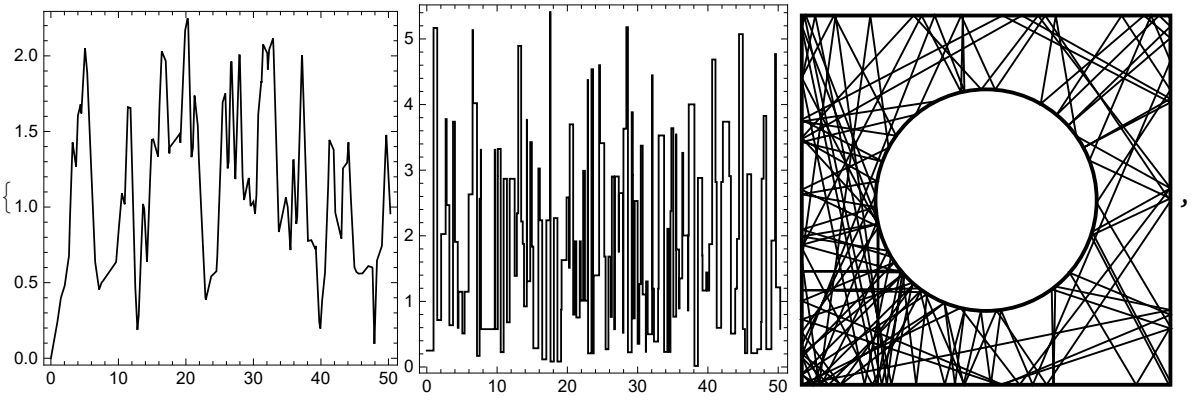
Out[82]= { }

```
time = 50;
```

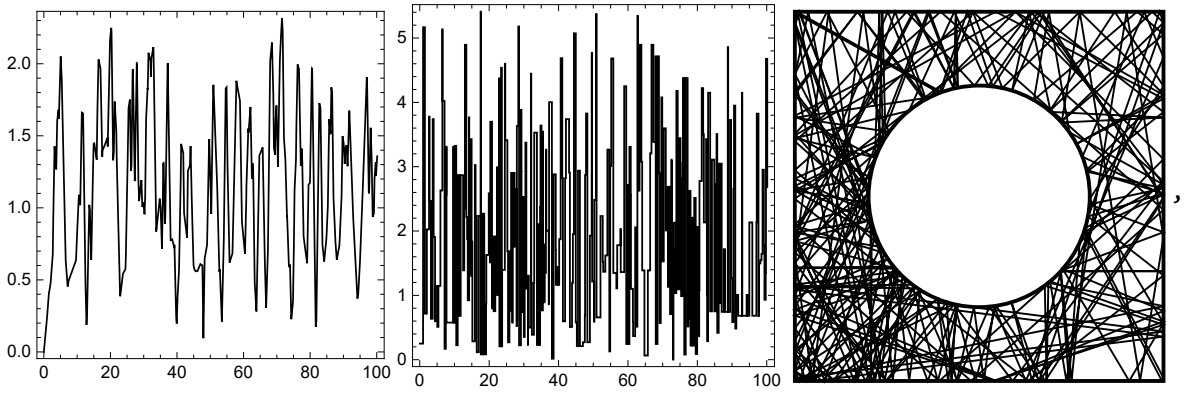
In[83]:=
```
For[i = 1, i < 4, i++,
  AppendTo[graphs2, Diff2[.2, i / 4, time * 1, .6]];
  AppendTo[graphs2, Diff2[.2, i / 4, time * 2, .6]];
  AppendTo[graphs2, Diff2[.2, i / 4, time * 3, .6]];
 ];
```

In[84]:=
```
graphs2
```

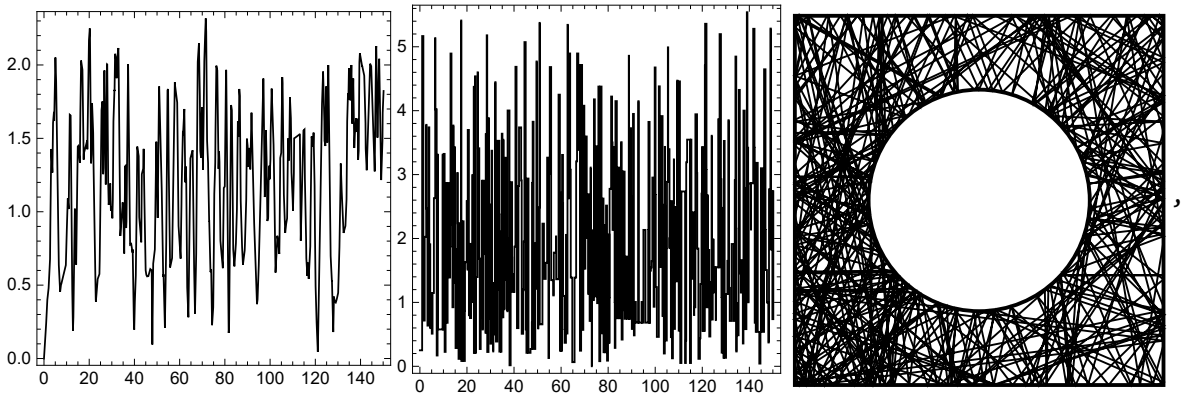
Out[84]= {


2.0
1.5
1.0
0.5
0.0
0 10 20 30 40 50
5
4
3
2
1
0
0 10 20 30 40 50

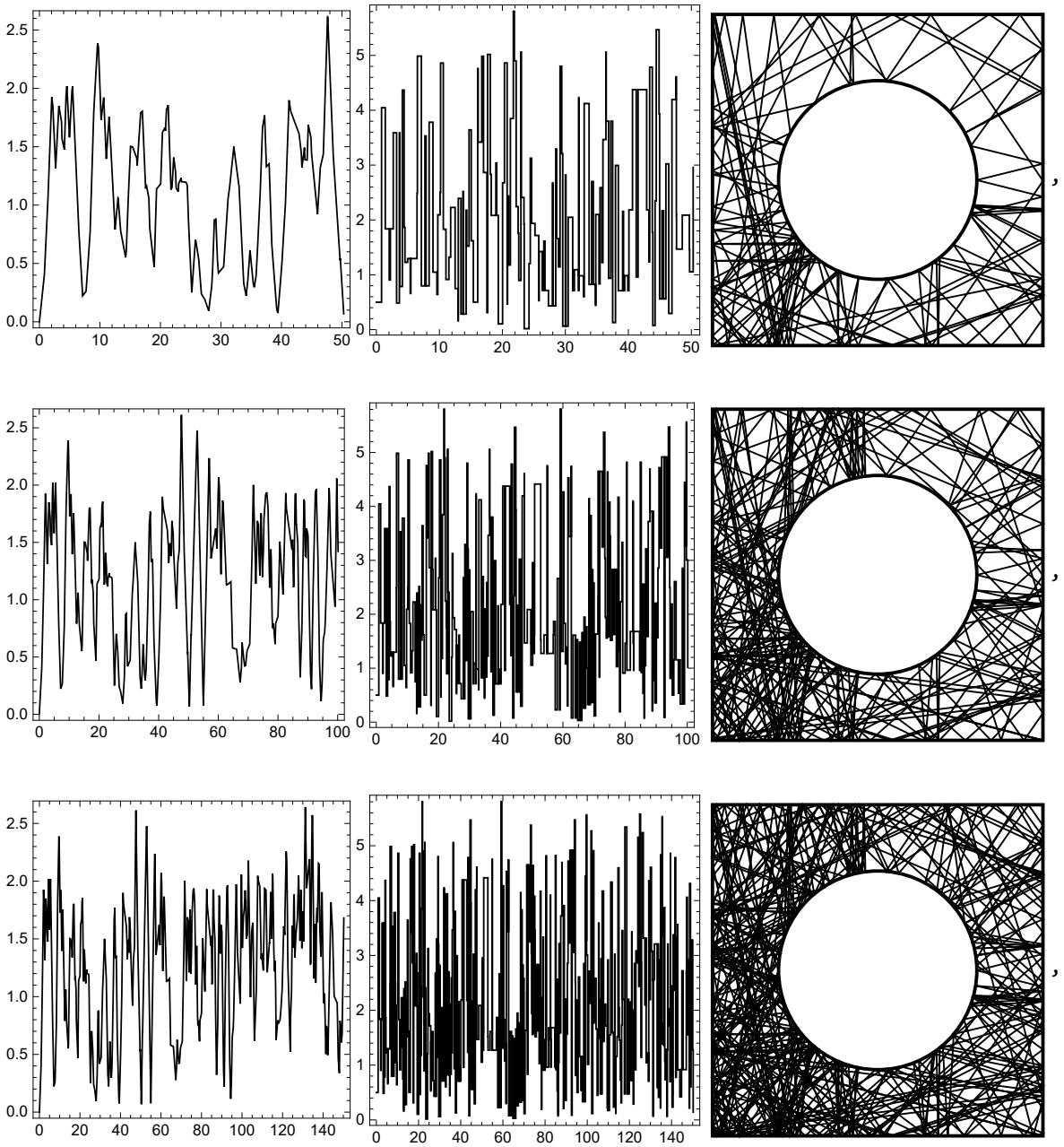
,

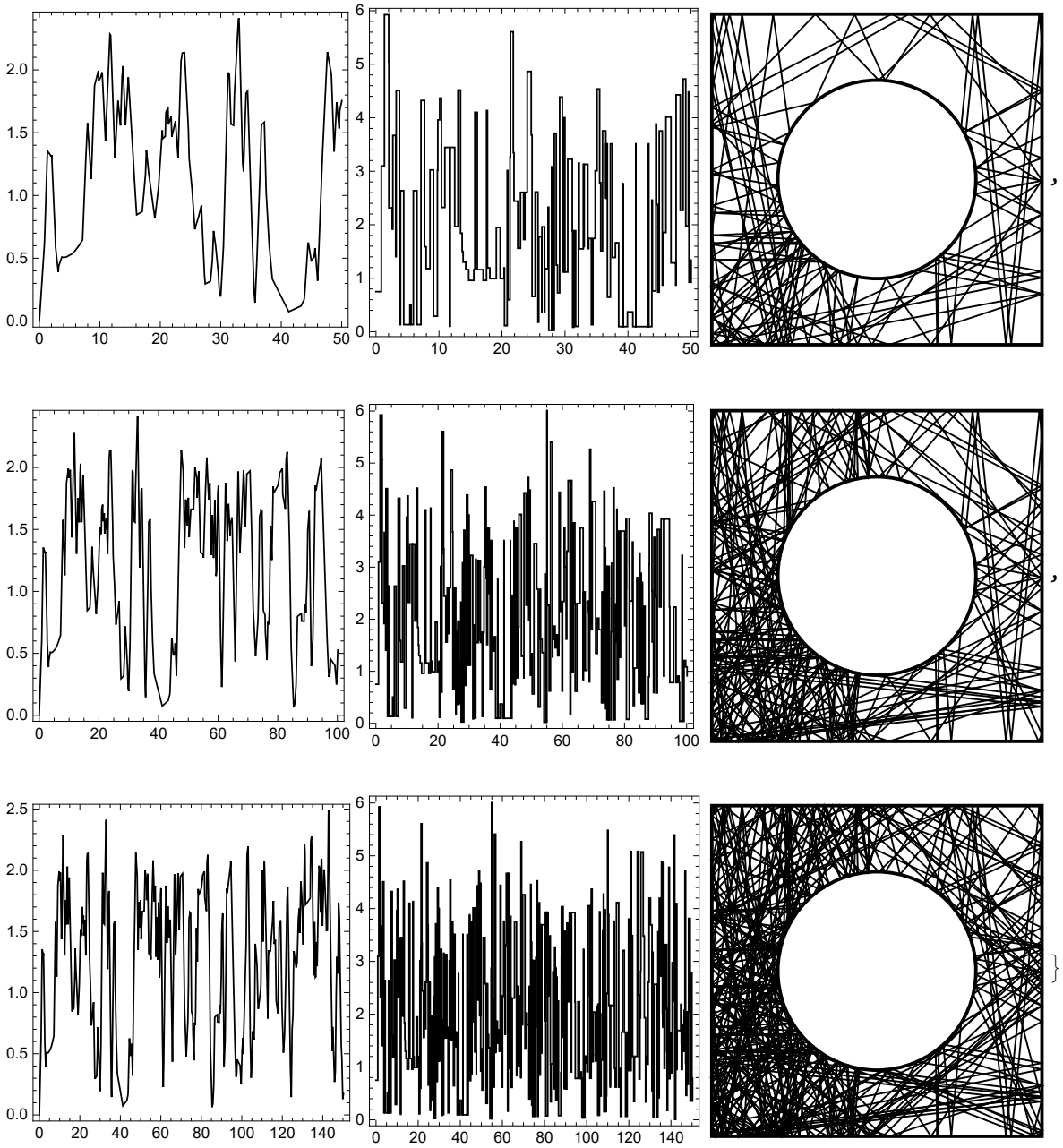

2.0
1.5
1.0
0.5
0.0
0 20 40 60 80 100
5
4
3
2
1
0
0 20 40 60 80 100

,


2.0
1.5
1.0
0.5
0.0
0 20 40 60 80 100 120 140
5
4
3
2
1
0
0 20 40 60 80 100 120 140

,


0.0
0.5
1.0
1.5
2.0
2.5
0
1
2
3
4
5
0
10
20
30
40
50
60
80
100
120
140



```
(* Conclusions so
 far: A greater angle will produce more random results with no object in the middle. *)
(* With an object in the middle, greater time will produce more random results,
and the angle does not matter *)
(* Now we need to study the size of the object to see if that
 matters. Time is varied with object size, angle is held constant *)
```