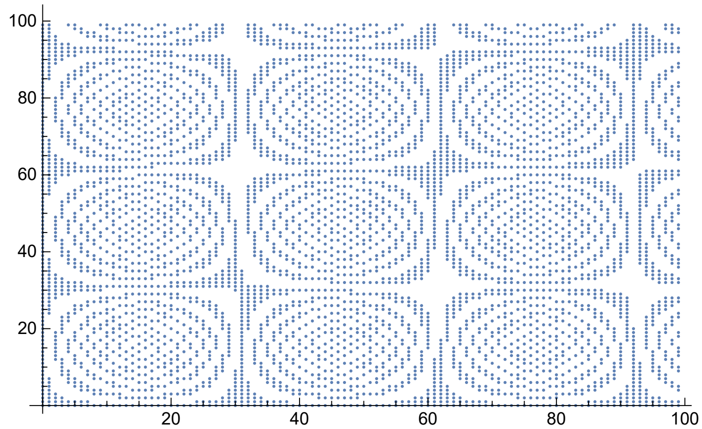
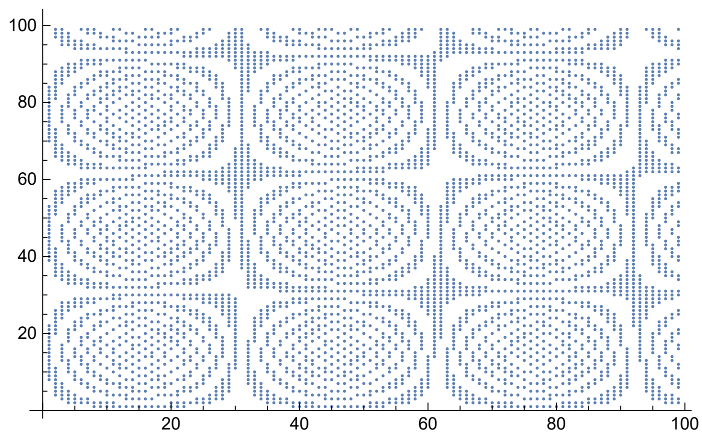
```
(* Adam Beck *)
(* Coin flip, problem 3 *)

(* Assumption: Coin radius is 1, height is 1 *)

(* I will first define several functions to help calculate the coin flip simulation *)

(* Calculates total time to go back to initial height *)

timeFunction[v_] := 2 v / 9.8
timeFunction[4.5]

0.918367

(* Angular velocity is in radians *)
(* converts angular velocity to degrees per second *)
angularToDegrees[a_] := N[a * (180 / Pi)]

(* Function to find where the coin is roated once
 it falls back into initial height. Imagine a unit circle. *)
(* The rightmost part of the coin, facing heads up,
is at degree 0. It is rotated countreclockwise when flipped *)
finalDegrees[timetofunction_, degrees_] := timetofunction * degrees

(* Reduces the result from finalDegrees into
 a value from 0 to 360 degrees on a unit circle *)
reducetobounds[finaldegrees_] := finaldegrees - (360 * Floor[finaldegrees / 360])

(* Takes a reduced degree and finds if it will land heads or tails. 1 = heads,
0 = tails. Assumes that the coin is starting heads up. *)
headsortails[reducedDegree_] := If[(reducedDegree > 270 || reducedDegree < 90), 1, 0]

(* This is where I am going to introduce some error *)

(* To land on the side, the coin needs to rotate exactly 90 or 270 degrees *)
(* However this will never happen as the
 precision of my calculation always has a decimal *)
(* Instead of rounding to the nearest n-th degree, I am going to take a ratio *)
(* If reducedDegree/90 or reducedDegree/270 is between .99 and 1.01,
it lands on its side. This seems reasonable *)
(* As a real coin has some thickness on its side,
allowing it to land on its side without rotating exactly 90 or 270 degrees *)
(* 1 = lands on side and 0 = no *)
```

```
side[reducedDegree_] := If[((reducedDegree / 90 ≥ .99 && reducedDegree / 90 ≤ 1.01) ||
      (reducedDegree / 270 ≥ .99 && reducedDegree / 270 ≤ 1.01)), 1, 0];

(* Implement a function to use all the above functions. Takes in a velocity
  and an angular momentum. Returns 1 = heads, 0 = tails, 2 = side *)
coinFlip[v_, w_] := (
  time = timeFunction[v];
      degrees = angularToDegrees[w];
      totalDegrees = finalDegrees[time, degrees];
      actualDegree = reducetobounds[totalDegrees];
      If[side[actualDegree] == 1, Return[2], Return[headsortails[actualDegree]]])

(* Create a list for the corresponding velocity and angular
 velocity that makes the coin land heads, tails, or on its side *)
headsListv = {};
headsListw = {};
tailsListv = {};
tailsListw = {};
sideListv = {};
sideListw = {};


(* Simulate all integer trials of the coin flip of
 velocity and angular velocity from 0 to 99 (inclusive) *)
a = 0;
b = 0;

While[a < 100,
  While[b < 100,
   result = coinFlip[a, b];
   If[result == 1, (AppendTo[headsListv, a]; AppendTo[headsListw, b])];
   If[result == 0, (AppendTo[tailsListv, a]; AppendTo[tailsListw, b])];
   If[result == 2, (AppendTo[sideListv, a]; AppendTo[sideListw, b])];
   b = b + 1
  ];
  b = 0;
  a = a + 1
 ];

(* Transpose the trials into a table in order to plot it *)
headsTable = Transpose[{headsListv, headsListw}];
tailsTable = Transpose[{tailsListv, tailsListw}];
sidesTable = Transpose[{sideListv, sideListw}];
```

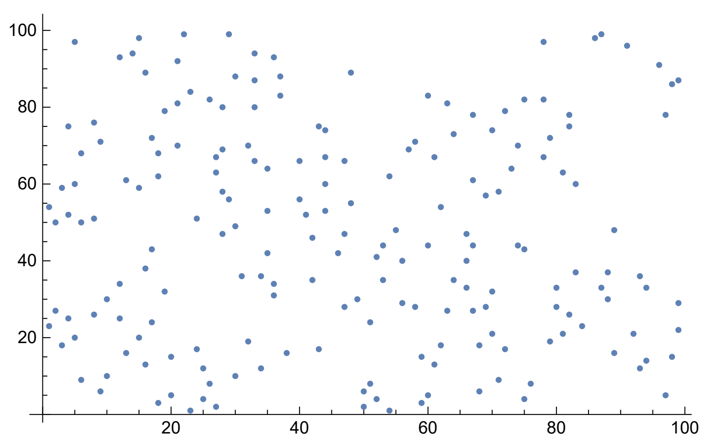
```
(* Plot the phase diagram for heads, tails, and side *)
ListPlot[headsTable]
```



```
ListPlot[tailsTable]
```



```
ListPlot[sidesTable]
```



```
(* There are several observations I can make from these phase diagrams *)
(* Most obviously, the sidesTable plot looks entirely random. There is no "pattern" *)
(* However, there is a distinct pattern when
 plotting velocity and angular veloctiy for heads and tails *)
(* The plot seems to have circles of increasing radii,
and intuitively this makes sense *)

(* Imagine a unit circle where the x axis is velocity and y axis is angular velocity *)
(* Take (0,0) and (1,0) and (0,1) lands on heads *)
(* To land on heads again if velocity was increased from 0 to,
say, .2, we would need a smaller angular veloctiy
 to make up for the increased velocity (the coin flips higher,
  so it has to spin a little less. The inverse of this statement is true too.*)
(* This point could be a point like (.2,.7) and if apply this
 pattern to all x values [-1,1] the plot will form a circle *)

(* Results for maximum v and w values: *)
(* 50: .502 heads, .4796 tails, .0184 side *)
(* 100: .4964 heads, .4842 tails, .0194 side *)
(* 150: .4934 heads, .4856 tails, .020 side *)
(* Conclusion: As v and w increase,
the ratio of heads and tails approaches 50/50 via Squeeze theorem.*)
(* A larger data set could not be computed as the program runs for quite some time! *)

(* I will now find the ratios for heads, tails, and sides for this trial *)
totalLength = Length[headsListv] + Length[tailsListv] + Length[sideListv]
```

10 000

```
headsratio = N[Length[headsListv] / totalLength]
tailsratio = N[Length[tailsListv] / totalLength]
sideratio = N[Length[sideListv] / totalLength]
```

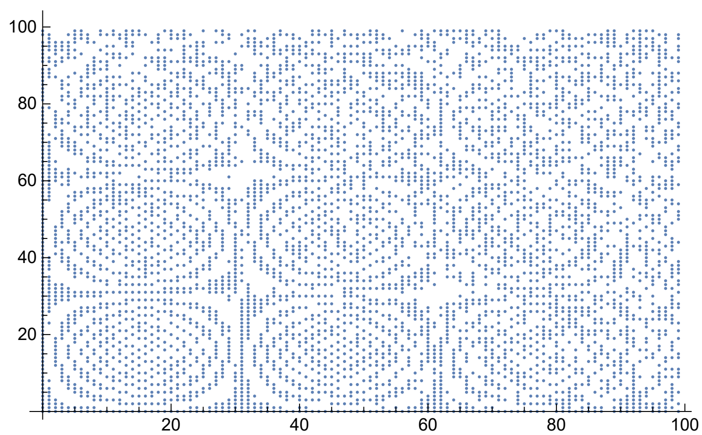
0.4964

0.4842

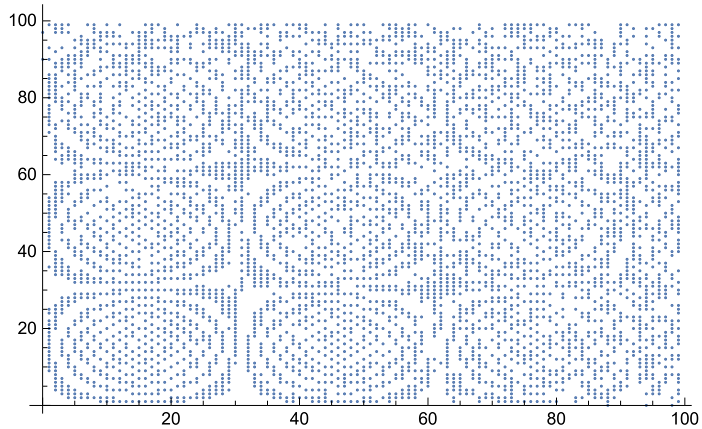
0.0194

```
(* I will now add error into the velocity and angular velocity calculations. First,
reset our lists to be empty *)
headsListv = {};
headsListw = {};
tailsListv = {};
tailsListw = {};
sideListv = {};
sideListw = {};


(* This is a trial where there is a random error to v and w between [-.1 and .1] *)
```

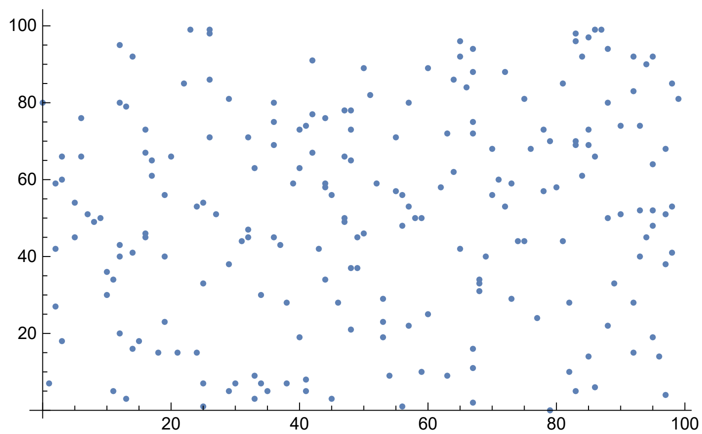
```
a = 0;
b = 0;
While[a < 100,
  While[b < 100,
   vError = RandomReal[{-.1, .1}] + a; (* Error is applied here *)
   wError = RandomReal[{-.1, .1}] + b;
   result = coinFlip[vError, wError];
   If[result == 1, (AppendTo[headsListv, a]; AppendTo[headsListw, b])];
   If[result == 0, (AppendTo[tailsListv, a]; AppendTo[tailsListw, b])];
   If[result == 2, (AppendTo[sideListv, a]; AppendTo[sideListw, b])];
   b = b + 1
  ];
  b = 0;
  a = a + 1
 ];

(* Transpose the data into a Table in order to plot *)

headsTable = Transpose[{headsListv, headsListw}];
tailsTable = Transpose[{tailsListv, tailsListw}];
sidesTable = Transpose[{sideListv, sideListw}];


(* Plot the phase diagram for heads, tails, and side *)
ListPlot[headsTable]
```



```
ListPlot[tailsTable]
```




```
ListPlot[sidesTable]
```




```
(* As we can see, the sidesTable is seemingly random again *)
(* The heads and tails graphs are exhibit the circular patterns again. However,
the patterns only apply to low v and w values. As one can see,
the patterns start to break up and
 become random data as v and w increase. I believe this is
 because the small "error" applied to v and w becomes very large
 over time as the time the coin is flipping increases. So there is no
 pattern anymore for heads and tails, just a seemingly random 50/50 data *)

(*Again, we take the ratio for this test and we see that it is about a 50/50 split *)
totalLength = Length[headsListv] + Length[tailsListv] + Length[sideListv]
```

10 000

```
headsratio = N[Length[headsListv] / totalLength]
tailsratio = N[Length[tailsListv] / totalLength]
sideratio = N[Length[sideListv] / totalLength]
```

```
0.4921
```

```
0.4867
```

```
0.0212
```

```
(* In conclusion, the coin flip simulation shows us
 that given a velocity and angular velocity with some "error",
a coin flip satisfies the equipartition property *)
```