The Standard Map

Based on a notebook by T. Timberlake (Berry College, GA).
See also the article T.Timberlake, Am. J. Phys. 72 (8) Aug 2004,
available at http://facultyweb.berry.edu/ttimberlake/cc/

Mathematica basics

The most important *Mathematica* functions for studying one dimensional maps are **`Nest`**, and **`NestList`**. **`Nest`** calculates the result of iterating a function a certain number of times, using a certain initial value for the variable. **`NestList`** calculates each result in the iteration process and forms all of these results into a list. The examples below illustrate how these functions can be used with the Standard Map.

First, we must define the function we wish to iterate.

```
In[1]:= K = 1.5;
        f[{θ_, r_}] := N[{Mod[θ + r - K Sin[θ], 2 π, -π], Mod[r - K Sin[θ], 2 π, -π]}];
```

Now suppose we wanted to find the result of iterating this function four times using an initial point {0.3,0.7}. In other words, we want to find f(f(f(f({0.3,0.7})))). In *Mathematica* we can simply use the **`Nest`** function to generate this result.

```
Nest[f, {0.3, 0.7}, 4]
```

```
{-0.33414, 0.212161}
```

If we wish to see a list of all of the first four iterates of f, using an initial point {0.3,0.7}, we use **`NestList`** instead.

```
NestList[f, {0.3, 0.7}, 4]
```

```
{{0.3, 0.7}, {0.55672, 0.25672}, {0.0208333, -0.535886},
 {-0.546301, -0.567134}, {-0.33414, 0.212161}}
```

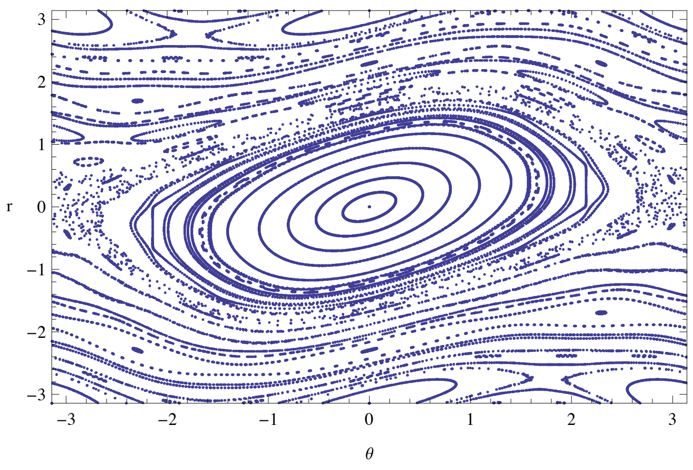
Note that this list contains five elements, the first element being our initial point (or the 0^{th} of the map).

Surfaces of Section

We can generate Poincare surfaces of section by applying NestList to several different initial points in phase space. In the example below each initial point is used to generate 400 other points through iteration of the map with K=0.8. The initial points are chosen to have θ=0 or θ=π/2, with 20 different values for r distributed evenly from 0 to 1.

```
In[12]:= K = 0.8; n = 400; θ1 = 0; θ2 = π / 2 ; rmax = 2 π; nr = 30;
         MapData = Flatten[Table[Join[NestList[f, {θ1, i rmax / nr}, n],
               NestList[f, {θ2, i rmax / nr}, n]], {i, 0, nr}], 1];
         ListPlot[MapData, PlotStyle → PointSize[0.002], PlotRange → π {{-1, 1}, {-1, 1}},
          Frame → True, RotateLabel → False, FrameLabel → {"θ", "r", "", ""}, Axes → False]
```

Out[14]=



Area-Preservation (Evolution of a Box of Initial Conditions)

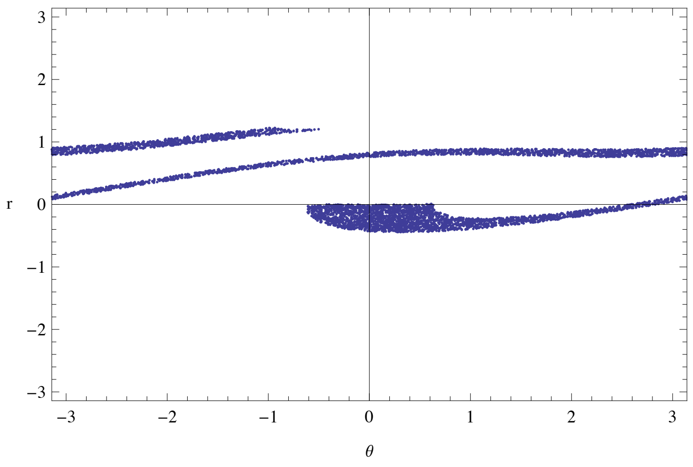
To illustrate the area-preserving nature of the Standard Map we start with a large number of initial points scattered randomly throughout some region. By applying the map to all of these points we find that this region is mapped to a new region that has a different shape but the same area. Repeated application of this procedure further distorts the shape of the region, but the area remains constant.

In the example below we start with 5000 points randomly distributed in ther rectangular region 0.4<θ<0.6 and 0<r<0.12. The map is iterated 4 times (`niter=4`) for these points and the resulting set of points is plotted.

```
In[15]:= K = 0.1; θmin = -0.2 π; θmax = 0.2 π; rmin = 0.0 π; rmax = 0.4 π; n = 5000; niter = 2;
         MapData2[niter_] := Table[Nest[f,
             {Random[Real, {θmin, θmax}], Random[Real, {rmin, rmax}]}, niter], {i, 0, n}];
         areaplot[niter_] := ListPlot[MapData2[niter], PlotStyle → PointSize[0.002],
           PlotRange → π {{-1, 1}, {-1, 1}}, Frame → True,
           RotateLabel → False, FrameLabel → {"θ", "r", "", ""}]
```

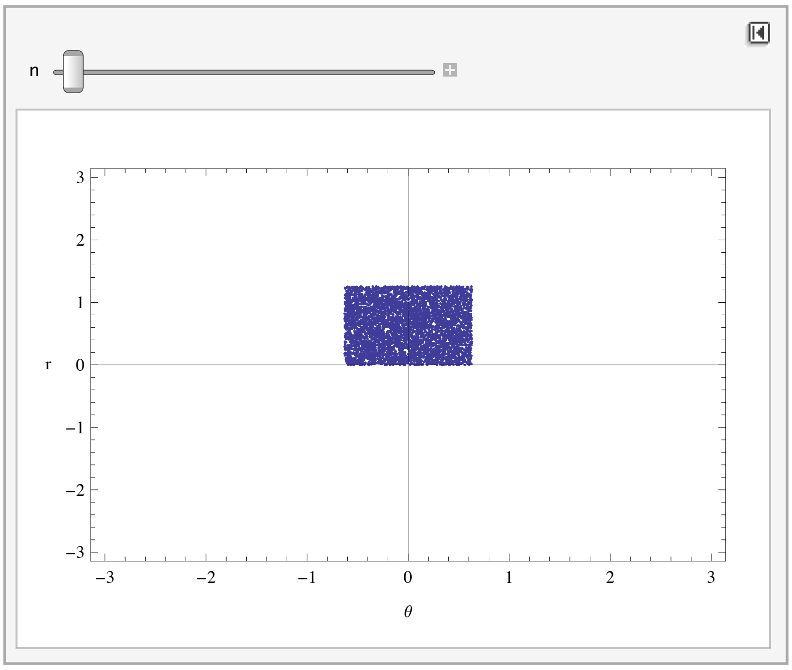
In[19]:=
```
areaplot[10]
```

Out[19]=



Animation

```
Manipulate[areaplot[n], {n, 0, 10, 1}, AppearanceElements → "ResetButton"]
```

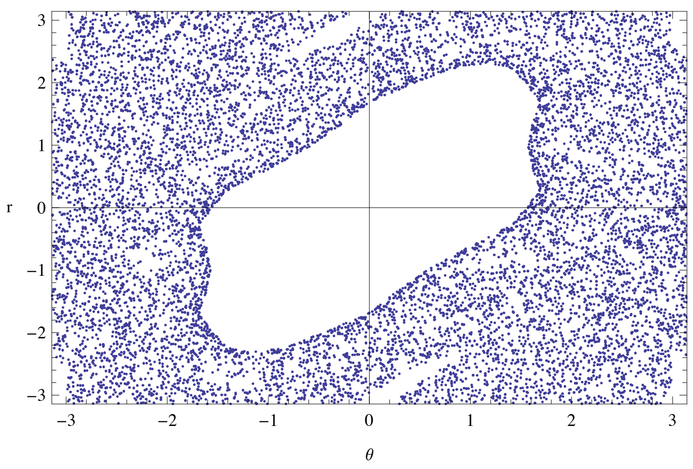


Single Trajectory

Plotting a single trajectory is very simple. The procedure is identical to that used to create the surface of section, but with only a single initial point used.

```
In[41]:= K = 1.98; θ0 = π / 2; r0 = 0.00; n = 10 000;
         MapData = NestList[f, {θ0, r0}, n];
         ListPlot[MapData, PlotStyle → PointSize[0.002], PlotRange → π {{-1, 1}, {-1, 1}},
          Frame → True, RotateLabel → False, FrameLabel → {"θ", "r", "", ""}]
```

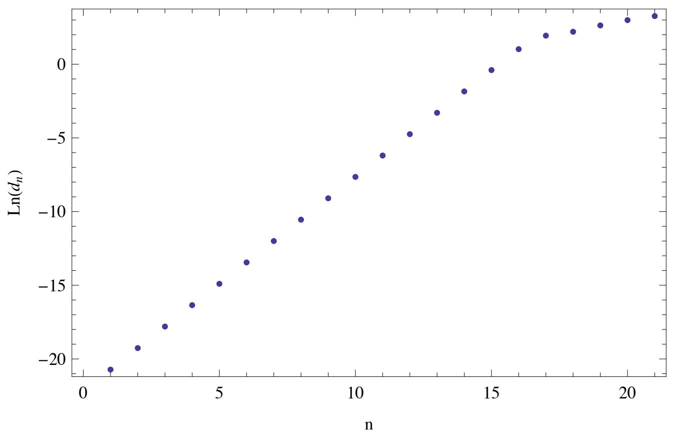
Out[43]=



Exponential Divergence and Lyapunov Exponents

To illustrate exponential divergence of neighboring trajectories we can start with two different, but very close, initial points. The distance between the iterates of these points should increase exponentially with time if the initial points are in a chaotic region of the phase space. Therefore, a plot of the natural log of this distance should increase linearly with the number of iterations. In the example below we create such a plot using initial points {π,0} (which is a fixed point) and a slightly displaced point. The graph shows the log of the distance between the trajectories versus the number of iterations of the map, up to 30 iterations. Note that the function is no longer defined Modulo 1, because this leads to incorrect measures of the distance between the trajectories.

```
K = 2.5; θ1 = π; r1 = 0; θ2 = π + 0.000000001; r2 = 0; n = 20;
f[{θ_, r_}] := N[{θ + r - K Sin[θ], r - K Sin[θ]}];
L1 = NestList[f, {θ2, r2}, n];
DList = Table[Log[Sqrt[(θ1 - L1[[i, 1]])^2 + (r1 - L1[[i, 2]])^2]], {i, n + 1}];
ListPlot[DList, Frame → True, Axes → False,
 FrameLabel → {"n", "Ln(dn)", "", ""}, PlotStyle → PointSize[0.01]]
```



This result can be compared to the positive Lyapunov exponent for this fixed point, which should be equal to the slope of the line in the previous plot. The Lyapunov exponent of the fixed point is found by taking the natural log of the eigenvalues of the Jacobian matrix for the fixed point (see previous section). Using the Jacobian given in the previous section we find:

```
K = 2.5;
Jac = {{1 + K, 1}, {K, 1}};
Log[Eigenvalues[Jac]]
```

```
{1.45057, -1.45057}
```

The slope in the above plot is roughly given by

```
(DList[[15]] - DList[[1]]) / 14
```

```
1.45188
```

So we see that the agreement between the two is good.