Logistic

迭代

定义函数f，迭代iter

In[1]:=
```
f[a_ : 2] := Function[x, a x (1 - x)]
```
纯函数

In[2]:=
```
(*x0: 起始点,m: 最大迭代次数*)
iter[a_, m_ : 100, x0_ : 0.2] :=
 NestWhileList[f[a], x0, Count[{##}, Last@{##}] == 1 &, All, m]
```
嵌套列表循环　计数　最后一个　全部

```
iter[2]
```

Out[3]=
```
{0.2, 0.32, 0.4352, 0.491602, 0.499859, 0.5, 0.5, 0.5, 0.5}
```

使用UnsameQ性能差

In[4]:=
```
(*x0: 起始点,m: 最大迭代次数*)
iter2[a_, m_ : 100, x0_ : 0.2] := NestWhileList[f[a], x0, UnsameQ, All, m]
```
嵌套列表循环　不恒等判定　全部

```
iter2[2]
```

Out[5]=
```
{0.2, 0.32, 0.4352, 0.491602, 0.499859, 0.5, 0.5, 0.5, 0.5}
```

对比性能

In[6]:=
```
Grid@Table[First@Timing@h[3, m], {h, {iter, iter2}}, {m, {100, 300, 500, 600}}]
```
格子　表格　第一个　计算时间

Out[6]=
```
   0.    0.015625 0.03125 0.046875
0.03125   0.75    3.4375   5.9375
```

In[7]:=
```
Grid@Table[First@Timing@h[3, m],
```
格子　表格　第一个　计算时间

```
  {h, {NestList[f[#1], 0.2, #2] &, iter}}, {m, {100, 300, 500, 600, 1000, 2000, 3000}}]
```
嵌套列表

Out[7]=
```
0.    0.       0.       0.      0.   0.   0.
0. 0.015625 0.03125 0.046875 0.125 0.5 1.125
```

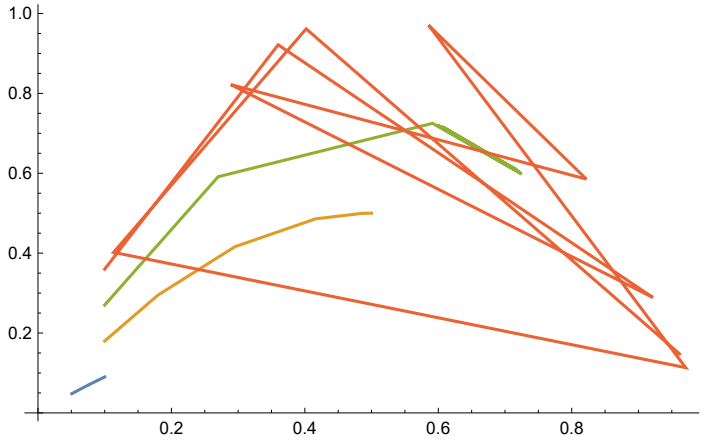
说明使用NestList的性能远高于NestWhileList

可视化迭代

选取不同的a值，从初值0 .1 开始迭代

In[8]:=
```
ListLinePlot[Table[Partition[iter[a, 10, 0.1], 2, 1], {a, 1, 4}]]
```
绘制点集的线条 表格 划分

Out[8]=



加入y=x上的点作为参考

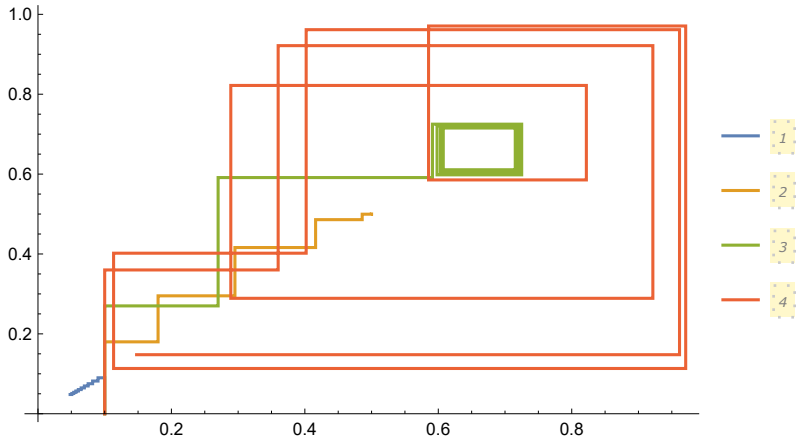
In[20]:=
```
lps[l_] := ReplacePart[{1, 2} → 0]@Partition[Riffle[l, l], 2, 1]
```
替换部分 划分 交互插入

```
lps[iter[2]]
```

Out[21]=
```
{{0.2, 0}, {0.2, 0.32}, {0.32, 0.32}, {0.32, 0.4352},
 {0.4352, 0.4352}, {0.4352, 0.491602}, {0.491602, 0.491602},
 {0.491602, 0.499859}, {0.499859, 0.499859}, {0.499859, 0.5}, {0.5, 0.5},
 {0.5, 0.5}, {0.5, 0.5}, {0.5, 0.5}, {0.5, 0.5}, {0.5, 0.5}, {0.5, 0.5}}
```

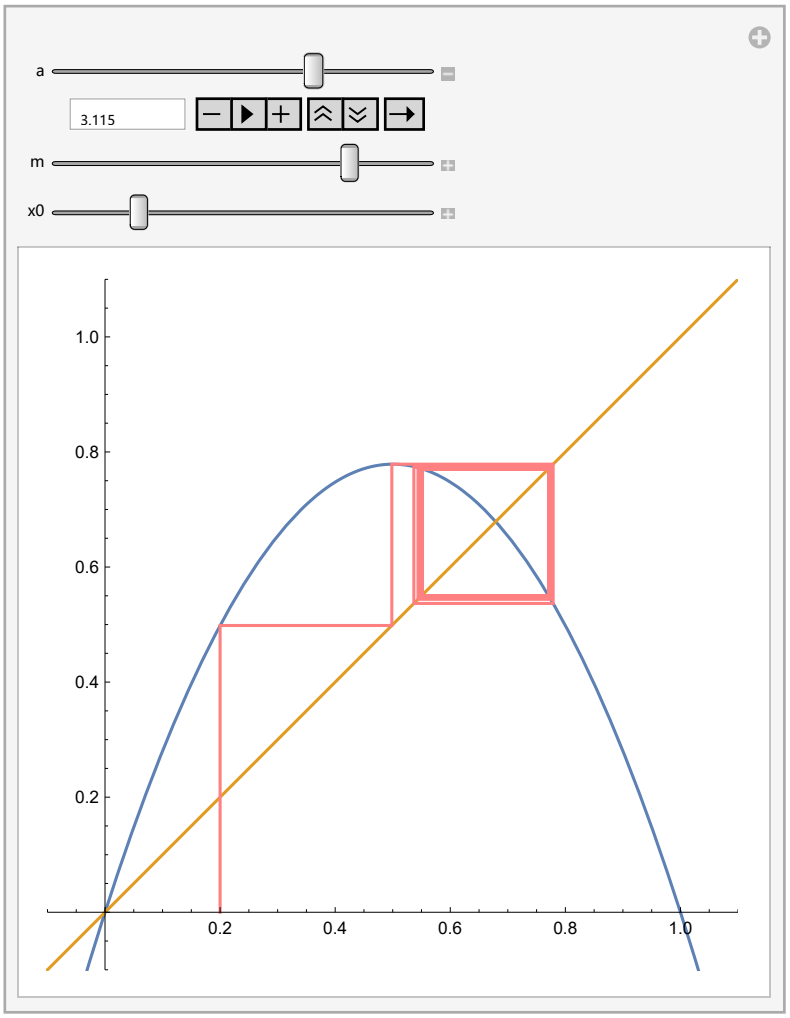
In[11]:=
```
ListLinePlot[Table[lps[iter[a, 10, 0.1]], {a, 1, 4}], PlotLegends → Automatic]
```
绘制点集的线条 表格 绘图的图例 自动

Out[11]=



In[36]:=
```
Manipulate[
交互式操作
 Show[Plot[{f[a][x], x}, {x, -0.1, 1.1},
 显示 绘图
   PlotRange → {{-0.1, 1.1}, {-0.1, 1.1}}, AspectRatio → 1],
    绘制范围                                宽高比
  ListLinePlot[lps@iter[a, m, x0]
   绘制点集的线条
   , PlotStyle → Pink, PlotRange → All]
     绘图样式      粉色   绘制范围      全部
], {{a, 2}, 1, 4}, {m, 10, 100, 1}, {{x0, 0.2}, 0, 1}]
```

Out[36]=

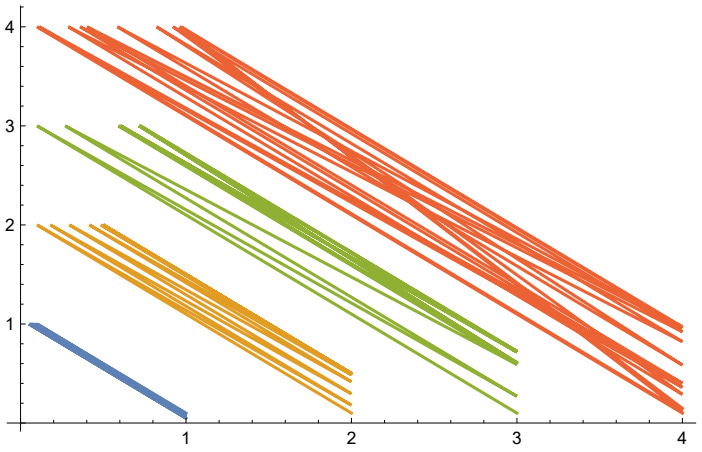


倍增周期分插图

临时效果

In[28]:=
```
ListLinePlot[
绘制点集的线条
 Table[Partition[Riffle[iter[a, 10, 0.1], a, {1, -2, 2}], 2, 1], {a, 1, 4, 1}]]
表格 划分 交互插入
```

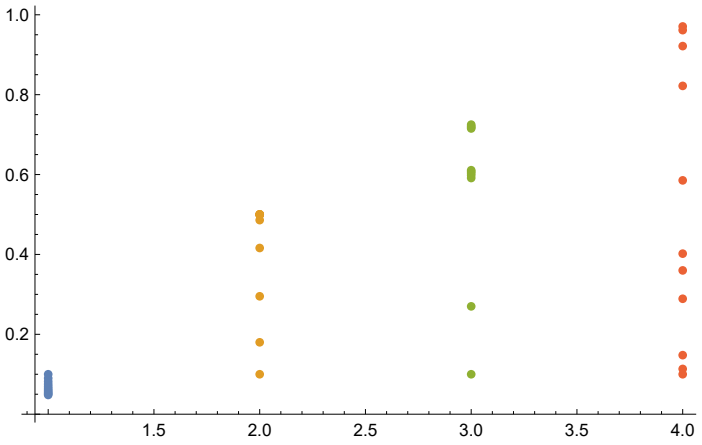
Out[28]=



将上面的划分方式改为不重叠的，两个元素一组

In[29]:=
```
ListPlot[Table[Partition[Riffle[iter[a, 10, 0.1], a, {1, -2, 2}], 2], {a, 1, 4, 1}]]
绘制点集 表格 划分 交互插入
```

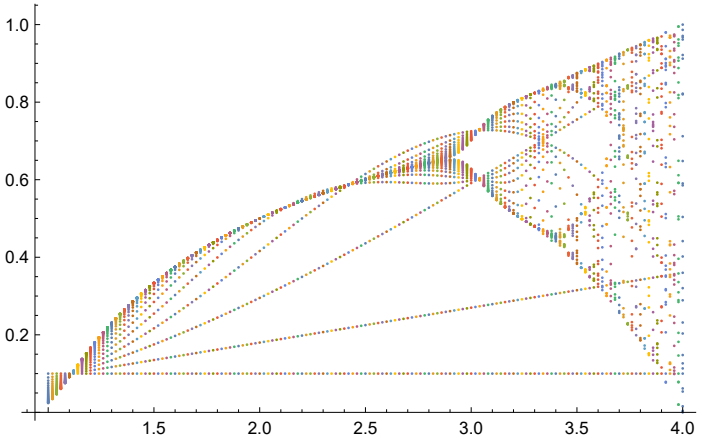
Out[29]=



改变迭代次数，和a的取值间隔

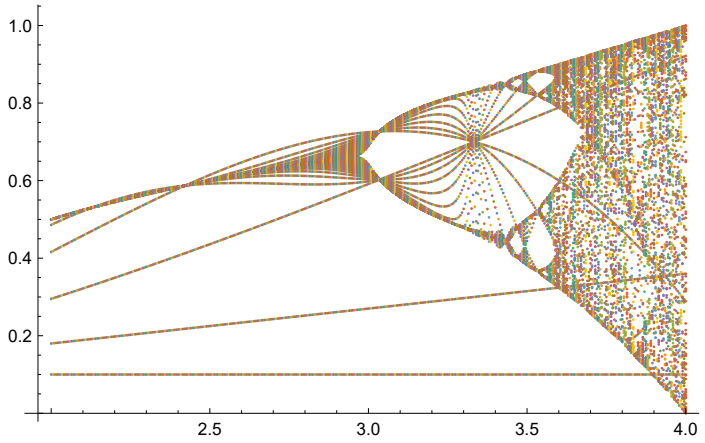
In[30]:=
```
ListPlot[Table[Partition[Riffle[iter[a , 30, 0.1], a, {1, -2, 2}], 2], {a, 1, 4, 0.02}]]
绘制点集 表格 划分 交互插入
```

Out[30]=



In[31]:=
```
ListPlot[Table[Partition[Riffle[iter[a , 80, 0.1], a, {1, -2, 2}], 2], {a, 2, 4, 0.005}]]
```
绘制点集 表格 划分 交互插入

Out[31]=



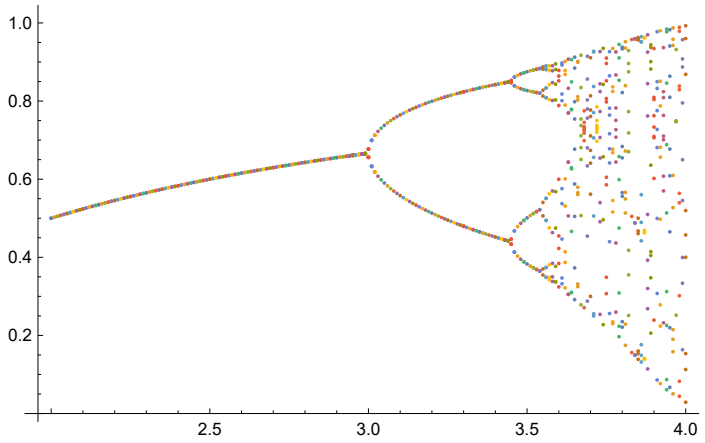
初始值x0只会影响迭代过程，不影响无限次迭代后的（周期性）收敛或发散

计算500次迭代后结果，如果发现周期性，只保留周期点

In[34]:=
```
cal[a_, n_ : 500] := Block[{l, p}, l = iter[a, n];
```
块
```
  p = Position[l, l[[-1]]];
```
位置
```
  p = If[Length[p] > 1, p[[-2, 1]], -10];
```
… 长度
```
  l[[p ;; -2]]]
```

In[35]:=
```
ListPlot[Table[Partition[Riffle[cal[a], a, {1, -2, 2}], 2], {a, 2, 4, 0.01}]]
```
绘制点集 表格 划分 交互插入

Out[35]=



In[39]:=
```
Manipulate[ListPlot[
  Table[Partition[Riffle[NestList[a #(1 - #) &, 0.1, m][[-d ;;]], a, {1, -2, 2}], 2],
    {a, 2, 4, interval}]], {{m, 10, "迭代次数"}, 10, 50, 1},
 {{d, 8, "保留"}, 1, m + 1, 1}, {{interval, 0.5, "a的间隔"}, 0.01, 1, 0.01}]
```
交互式操作 绘制点集 表格 划分 交互插入 嵌套列表

Out[39]=