The Logistic Map

Introduction

One of the most challenging topics in science is the study of chaos. As an example of chaos, consider fluid flowing round an object. If the velocity of the fluid is not very large the fluid flows in a smooth steady way, called "laminar flow", which can be calculated for simple geometries. However, if the velocity is increased the flow becomes more complicated, and depends on time and space in a non-trivial way which is essentially impossible to calculate. This "turbulent" behavior is called "chaos".

Note that chaos does not mean that the behavior is completely random; one does see recognizable structures, such as little whirlpools, called "vortices", but it is very complicated. The reason that chaotic behavior is essentially impossible to calculate is that the future behavior of the fluid depends in a very sensitive way on what the fluid is doing now. If we consider two configurations of the velocity field, which differ only by a very small amount, then, in the chaotic regime, those differences grow exponentially time, (the coefficient in the exponential is called the "Lyapunov exponent" which we will calculate in this notebook for a particular model) so that eventually the two velocity fields become completely different. This "sensitivity to initial conditions" is generally taken as the definition of chaos. Because the flow of air in the atmosphere is chaotic, it is impossible to predict the weather in detail for more than about a week because we cannot integrate the equations with sufficient accuracy, even assuming that we had sufficiently detailed data on the state of the weather now (which we don't).

While a lot of chaos theory was developed by Poincaré around the start of the 20th century, there were few subsequent developments (because the field is so difficult) until, starting in the 1980s, the mass availability of computers meant that the equations could easily be integrated numerically, and perhaps most importantly, the solutions visualized by computer graphics.

As the velocity of the fluid increases, rather than there being a *discontinuous* change from laminar to turbulent flow at a critical velocity, the motion changes only *gradually* as it passes the instability where it first becomes non-laminar, and it is only at larger values of the velocity that fully developed turbulence appears. An important question, then, is the nature of this **"transition to chaos"**. He we will use the graphical ability of *Mathematica* to consider a very simple looking model which has a transtion to chaos. There does not seem to be a single route to chaos, but the model that we study here, shows one of them, known as **"period doubling"**. It is, in fact, the world's simplest model with a transition to chaos.

The Model

Most realistic systems with chaotic behavior, such as fluid flow around an obstacle, are described by a non-linear partial differential equation, which determines how the velocity changes in space and time. We will see later in the course that chaos also occurs in many ordinary (non-linear) differential equations. Here we will consider something even simpler, an **"iterative map"**. This means that starting from an initial value of a variable, x_0 say, we generate a sequence of values, x_1, x_2, etc. from the map (i.e. function), $x_{n+1} = f(x_n)$ where we here make a simple choice f(x) = 4 λ x (1 - x), i.e.

$$x_{n+1} = 4\lambda\, x_n(1-x_n),$$

where λ is parameter. In other words, $x_1 = 4\lambda\, x_0(1-x_0)$, $x_2 = 4\lambda\, x_1(1-x_1)$, etc. We will be interested in the behavior of successive iterations of this map, as a function of the parameter λ. In particular we will study the behavior of the x_n for large n.

We consider λ in the range from 0 to 1, so, if x_0 is between 0 and 1, it is easy to see that all subsequent values of x also lie in this range. In fact the largest value of x_{n+1}(which occurs for x_n= 1/2) is equal to λ.

This so-called **"logistic map"** has been used as model for population dynamics, but here we just treat it as a toy model which has a transition to chaos.

A first look at the properties of the model

We create the function f in *Mathematica* as follows:

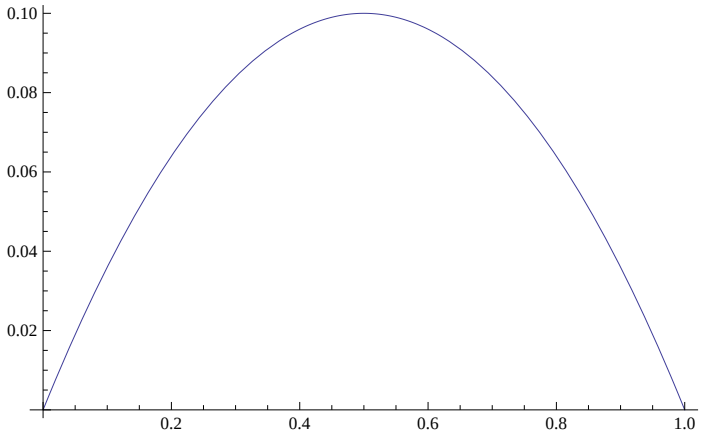
```
Clear["Global`*"]
```

In[3]:=
```
f[x_] := 4 λ x (1 - x);
```

It is a parabola which vanishes at x = 0 and 1. We plot it for λ = 0.4:

In[4]:=
```
Plot[f[x] /. λ → 0.4, {x, 0, 1}]
```

Out[4]=



Let us first see what happens to successive iterations for a few values of λ. If we start with λ < 0.25, then one can easily show analytically that $x_{n+1} < x_n$ and $\lim_{n\to\infty} x(n) = 0$. To verify this in *Mathematica* we put λ = 0.1 and generate a list of successive values with the *Mathematica* function **NestList[func, x0, n]**, in which func(x) is the function to be iterated, x0 is the starting value of x, and n is the number of iterations

In[5]:=
```
λ = 0.1; NestList[f, 0.5, 10]
```

Out[5]=
```
{0.5, 0.1, 0.036, 0.0138816, 0.00547556, 0.00217823,
 0.000869395, 0.000347456, 0.000138934, 0.0000555659, 0.0000222251}
```

Here we have set λ = 0.1, and then iterated f(x) 10 times with the initial value 0.5. The initial value of x is unimportant provided it is not a 0 or 1 in which case all the subsequent values of x would be zero. In this example we see that successive values eventually tend to zero. We say that zero is a **fixed point**, where the initial and final values of x are equal, i.e. X is a fixed point of the function f if

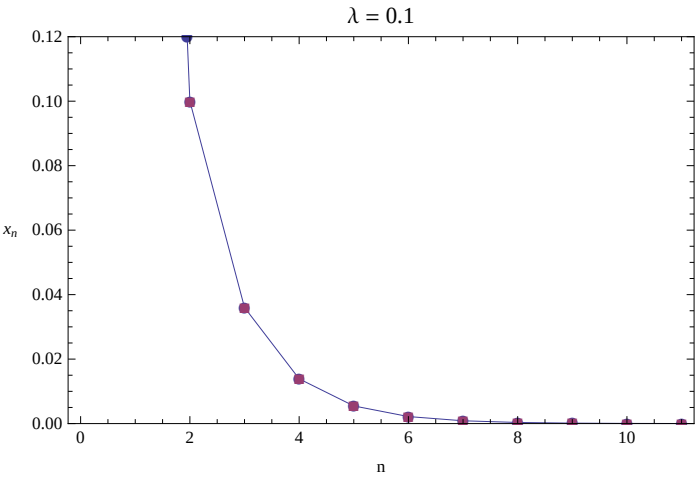
```
f(X) = X
```

It is best to see successive iterations graphically, i.e.

In[6]:=
```
λ = 0.1;
```

In[12]:=
```
pt = ListPlot[{NestList[f, 0.5, 10], NestList[f, 0.5, 10]}, Joined → {True, False},
  Axes -> False, Frame -> True, FrameLabel -> {"n", "xn"}, PlotRange -> {0, 0.12},
  PlotMarkers → Automatic, RotateLabel -> False, PlotLabel -> "λ = 0.1"]
```

Out[12]=



We already discussed fixed point iteration in an earlier part of the class, and there we showed that a fixed point is *stable*, i.e. you converge towards it if you start off not too far away, provided

$|f'(X)| < 1$

Since f'(x) = 4 λ (1 - 2x), the fixed point at x = 0 is stable when 4λ < 1, i.e. λ < 0.25. For λ > 0.25 this fixed point is unstable, and we need to find out what happens instead.

If we start from a value of λ greater than 0.25 but less than 0.75, successive points "flow" to a **fixed point** at a *non-zero* value of x. However, for values of λ a little larger than 0.75 the fixed point "*bifurcates*" to a "**limit cycle**" of period 2. This then bifurcates again (i.e. the **period doubles**) at a larger value of λ to a limit cycle with period 4. As λ increases the period continues to double at successively closer and closer values of λ until, at around λ = 0.89, the **period becomes infinity** and we have **chaotic behavior**. This is illustrated in the following figure with four values of λ (λ = 0.6, fixed point), (λ = 0.8, period 2 limit cycle), (λ = 0.88, period 4 limit cycle), and (λ = 0.91, chaos):

In[14]:=
```
λ = 0.6;
g1 = ListPlot[{NestList[f, 0.5, 10], NestList[f, 0.5, 10]}, Joined → {True, False},
  Axes -> False, Frame -> True, FrameLabel -> {"n", "xn"}, PlotRange -> {0.5, 0.62},
  PlotMarkers → Automatic, RotateLabel -> False, PlotLabel -> "λ = 0.6"];
```

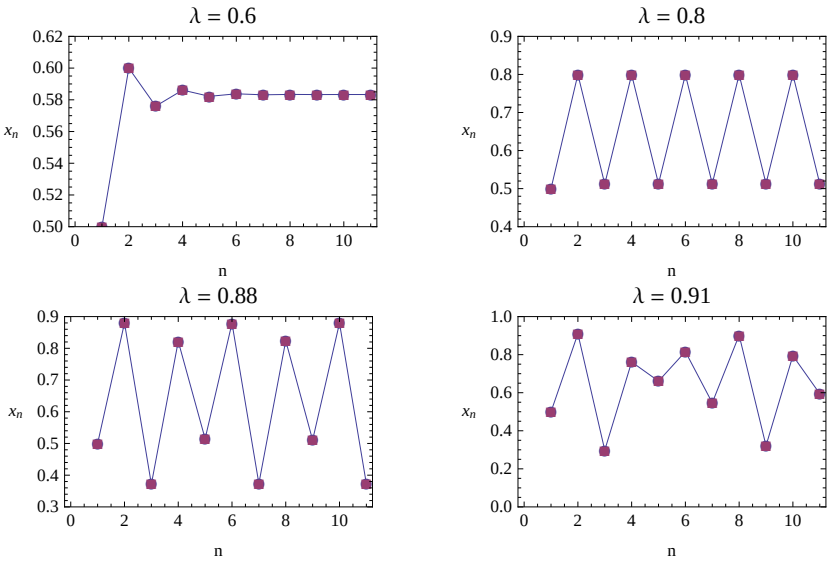
In[15]:=
```
λ = 0.8;
g2 = ListPlot[{NestList[f, 0.5, 10], NestList[f, 0.5, 10]}, Joined → {True, False},
  Axes -> False, Frame -> True, FrameLabel -> {"n", "xn"}, PlotRange -> {0.4, 0.9},
  PlotMarkers → Automatic, RotateLabel -> False, PlotLabel -> "λ = 0.8"];
```

In[16]:=
```
λ = 0.88;
g3 = ListPlot[{NestList[f, 0.5, 10], NestList[f, 0.5, 10]}, Joined → {True, False},
  Axes -> False, Frame -> True, FrameLabel -> {"n", "xn"}, PlotRange -> {0.3, 0.9},
  PlotMarkers → Automatic, RotateLabel -> False, PlotLabel -> "λ = 0.88"];
```

In[17]:=
```
λ = 0.91;
g4 = ListPlot[{NestList[f, 0.5, 10], NestList[f, 0.5, 10]}, Joined → {True, False},
  Axes -> False, Frame -> True, FrameLabel -> {"n", "xn"}, PlotRange -> {0, 1},
  PlotMarkers → Automatic, RotateLabel -> False, PlotLabel -> "λ = 0.91"];
```

In[18]:=
```
Show[GraphicsGrid[{{g1, g2}, {g3, g4}}, Spacings → Scaled[0.05]]]
```

Out[18]=



The sequence towards which the x values converge is called an **attractor**. We have seen that an attractor may be a **fixed point**, a **limit cycle**, or a **chaotic** attractor. We will be interested in the nature of the attractor as a function of λ.

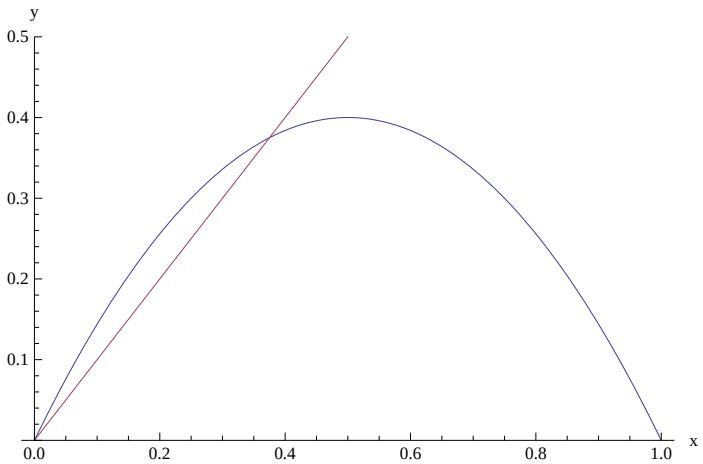
Fixed Points

In previous section, we saw that, for certain values of λ, the sequence of $x_n$converges to a fixed point value X where X = f(X).We illustrate a fixed point graphically for the case of λ = 0.4

In[19]:=
```
λ = 0.4;
```

In[20]:=
```
Plot[{f[x], x}, {x, 0, 1}, AxesLabel → {"x", "y"}, PlotRange → {0, 0.5}]
```

Out[20]=



The fixed point value is the point of intersection of the line y = x with the curve y = f(x).

Mathematica has a convenient function called **FixedPoint[f, start]** for computing fixed points of maps. The first argument is the function and the second is the starting value for x. For example, still considering λ = 0.4, we get

In[21]:=
```
FixedPoint[f, 0.5]
```

Out[21]=
```
0.375
```

FixedPoint gives a result when two successive values are *exactly* the same. If convergence is very slow, or if roundoff errors prevent exact agreement, it is best to put in a less stringent test for determining whether convergence has been reached. This is done with the option **SameTest** to the **FixedPoint** command. The following example shows how it works.

In[22]:=
```
FixedPoint[f, 0.5, SameTest → (Abs[#1 - #2] < 10^-11 &)]
```

Out[22]=
```
0.375
```

The notation is that of a pure function, in which **#1** and **#2** are the arguments and the function is terminated by **&**.

We can find the value of λ where the fixed point at non-zero X becomes unstable quite easily using pencil and paper. We need to simultaneously solve

$x = f(x), \quad |f'(x)| = 1$

The solution is

$\lambda = 3/4, \; x = 2/3$

(There is also a second solution $\lambda = 1/4$, x = 0 which is the other limit of stability of the $x \neq 0$ fixed point; as we have seen, for $\lambda < 0.25$ the stable fixed point is at x = 0.) *Mathematica* will also give us the above result

In[24]:=
```
Clear[x, λ]
```

In[25]:=
```
Solve[{1 == 4 λ (1 - x), 1 == Abs[4 λ (1 - 2 x)]}, {x, λ}]
```

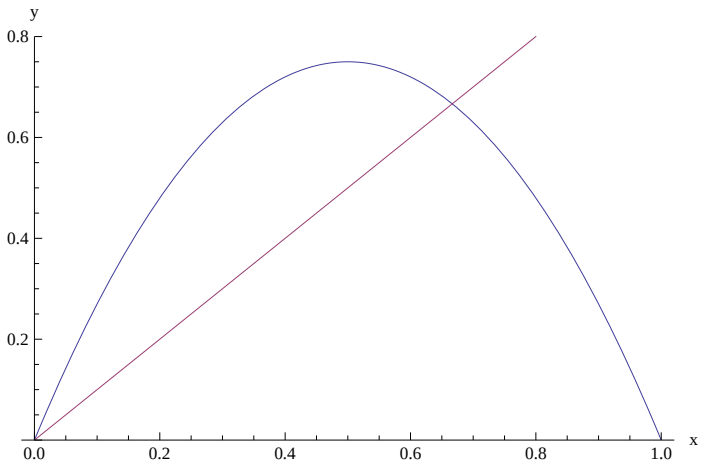
Out[25]=
$$\left\{\left\{x \to 0, \lambda \to \frac{1}{4}\right\}, \left\{x \to \frac{2}{3}, \lambda \to \frac{3}{4}\right\}\right\}$$

Let's plot **f(x)** and **x** at the borderline case $\lambda = 0.75$.

In[26]:=
```
λ = 0.75;
```

In[27]:=
```
Plot[{f[x], x}, {x, 0, 1}, AxesLabel → {"x", "y"}, PlotRange → {0, 0.8}]
```

Out[27]=



We see that f '(X) < 0 (X is the intersection point). Since the magnitude of the derivative is 1, we have |f '(X)| = -1.

We have seen that for certain choices of λ bigger that 0.75 successive values converge to a limit cycle of length 2. This is a fixed point of the twice iterated function f(f(x)). Lets call this f2(x). First we **Clear** λ and x (we don't want to **Remove** λ because **Remove** not only removes information about λ but also removes λ from the list of variables that *Mathematica* knows about; this gives problems when defining f2. **Clear** just removes the information about λ):

In[28]:=
```
Clear[λ, x]
```

In[29]:=
```
f2[x_] = f[f[x]]
```

Out[29]=
```
16 (1 - x) x λ^2 (1 - 4 (1 - x) x λ)
```

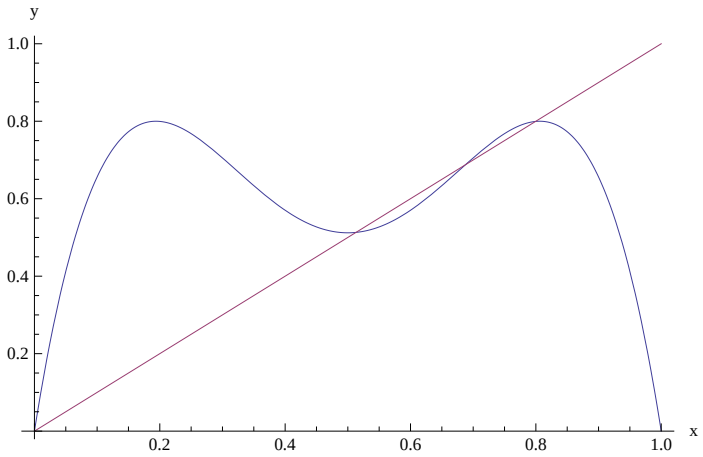
Let's take $\lambda = 0.8$, where we found above that the system goes into a limit cycle of length 2

In[30]:=
```
λ = 0.8;
```

and plot y=f2 (as well as y=x) against x

In[31]:=
```
Plot[{f2[x], x}, {x, 0, 1}, AxesLabel → {"x", "y"}]
```

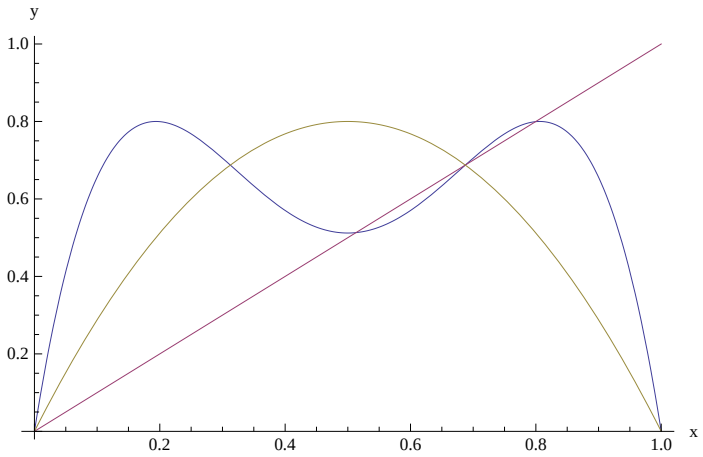
Out[31]=



We see that there are now 3 fixed points. The middle one is also a fixed point of f (note that any fixed point of f is also a fixed point of f2). See the plot below which also includes f (the parabola).

In[32]:=
```
Plot[{f2[x], x, f[x]}, {x, 0, 1}, AxesLabel → {"x", "y"}]
```

Out[32]=



We can locate this fixed point using **FindRoot** (not **FixedPoint** because it is untable)

In[33]:=
```
unstableFP = FindRoot[ f[x] == x, {x, 0.7}]
```

Out[33]=
```
{x → 0.6875}
```

This fixed point is unstable because | f2'| (and also | f ' |) is greater than 1 at this value of x. Indeed, one can see from the figure above that f2[x] has a greater slope at the middle fixed point than the slope of x (i.e. its slope is greater than 1). We can also find the precise numerical values of the derivative of f and f2 at this fixed point,

In[34]:=
```
{f'[x] /. unstableFP, f2'[x] /. unstableFP}
```

Out[34]=
```
{-1.2, 1.44}
```

showing that they are greater in magnitude than 1, confirming that the fixed point at 0.6875 is unstable.

The other two are fixed points of f2 (the two values of x correspond to the values of x in the period-2 cycle of f) and these are stable because |f2'| < 1 at these values of x, as is clear from the above figure and which we shall confirm quantitatively.

We can find the stable fixed points of f2 using the **FixedPoint** command

In[35]:=
```
stableX = FixedPoint[f2, 0.5, SameTest → (Abs[#1 - #2] < 10^-11 &)]
```

Out[35]=
```
0.513045
```

We verify that | f2' < 1 | at this value of X

In[36]:=
```
f2'[stableX]
```

Out[36]=
```
0.16
```

The value 0.513045 is the lower of the two stable fixed points of f2 in the above figure. Depending on the different initial value for $x_0$we would converge either to this one or the other one, which is close to x=0.8:

In[37]:=
```
stableX2 = FixedPoint[f2, 0.8, SameTest → (Abs[#1 - #2] < 10^-11 &)]
```

Out[37]=
```
0.799455
```

In[38]:=
```
f2'[stableX2]
```

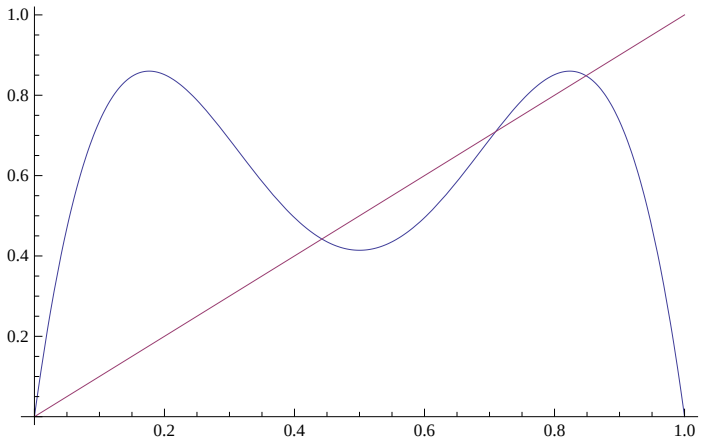
Out[38]=
```
0.16
```

These last results are for λ = 0.8, where the fixed point of f2 (2-cycle of f) is stable. We now ask *Mathematica* to tell us where this attractor becomes unstable. We will help *Mathematica* by specifying that the derivative is -1 (rather than +1) at the instability point (We can see this, for example from the figure below, which is for λ = 0.86, near the limit of stability. The stable fixed point values of f2 are the first and third fixed points (not counting the trivial one at 0), as discussed above, which are near 0.4 and 0.9. The derivative is clearly negative there. The intermediate fixed point, at about 0.7 is unstable since one can see from the figure that its derivative is greater than 1. (Remember that the slope of the straight line is one.)

In[39]:=
```
Clear[λ]
```

In[40]:=
```
Plot[{f2[x] /.λ → 0.86, x}, {x, 0, 1}]
```

Out[40]=



In[41]:=
```
Clear[λ]
```

In[42]:=
```
Solve[{f2[x] == x, f2'[x] == -1}, {x, λ}]
```

Out[42]=
$$\left\{\left\{x \to 0,\ \lambda \to -\frac{i}{4}\right\},\ \left\{x \to 0,\ \lambda \to \frac{i}{4}\right\},\ \left\{x \to \frac{3}{5} - \frac{i}{5},\ \lambda \to \frac{1}{2} - \frac{i}{4}\right\},\right.$$
$$\left\{x \to \frac{3}{5} + \frac{i}{5},\ \lambda \to \frac{1}{2} + \frac{i}{4}\right\},\ \left\{x \to \frac{1}{10}\left(4 - \sqrt{6} - \sqrt{14 + 4\sqrt{6}}\right),\ \lambda \to \frac{1}{4}\left(1 - \sqrt{6}\right)\right\},$$
$$\left\{x \to \frac{1}{10}\left(4 - \sqrt{6} + \sqrt{14 + 4\sqrt{6}}\right),\ \lambda \to \frac{1}{4}\left(1 - \sqrt{6}\right)\right\},$$
$$\left\{x \to \frac{1}{10}\left(4 + \sqrt{6} - \sqrt{14 - 4\sqrt{6}}\right),\ \lambda \to \frac{1}{4}\left(1 + \sqrt{6}\right)\right\},$$
$$\left.\left\{x \to \frac{1}{10}\left(4 + \sqrt{6} + \sqrt{14 - 4\sqrt{6}}\right),\ \lambda \to \frac{1}{4}\left(1 + \sqrt{6}\right)\right\}\right\}$$

There are several solutions. To see which we want it is useful to compute the numerical values:

In[43]:=
```
λ /. N[%]
```

Out[43]=
```
{0. - 0.25 i, 0. + 0.25 i, 0.5 - 0.25 i, 0.5 + 0.25 i, -0.362372, -0.362372, 0.862372, 0.862372}
```

The only solution with λ real and positive is $\lambda = (1 + \sqrt{6})/4 = 0.86237$. At this value of λ, the 2-cycle becomes unstable and gives way to a 4-cycle.

We can test for 4-cycles by looking at the fourth iterated function f4(x) = f2(f2(x)):

In[44]:=
```
Clear[λ, x]
```

In[45]:=
```
f4[x_] = f2[f2[x]]
```

Out[45]=
$$256\,(1-x)\,x\,\lambda^4\,(1-4\,(1-x)\,x\,\lambda)\,\left(1-16\,(1-x)\,x\,\lambda^2\,(1-4\,(1-x)\,x\,\lambda)\right)$$
$$\left(1-64\,(1-x)\,x\,\lambda^3\,(1-4\,(1-x)\,x\,\lambda)\,\left(1-16\,(1-x)\,x\,\lambda^2\,(1-4\,(1-x)\,x\,\lambda)\right)\right)$$

We take $\lambda = 0.88$ where above we found a length-4 cycle above:

In[46]:=
```
λ = 0.88;
```

This length-4 cycle is confirmed because we find a fixed point of f4:

In[47]:=
```
FixedPoint[f4, 0.5, SameTest → (Abs[#1 - #2] < 10^-11 &)]
```

Out[47]=
```
0.512076
```

This value for x is one of the 4 values on the 4-cycle of f to which the trajectory converges. Starting with different values for x_0,we could converge to the other values of x.

Using *Mathematica*, we can determine where the stable fixed point of f4 (4-cycle of f) becomes unstable. Since this is too complicated to solve analytically, and since **NSolve** gives a large number of roots, I prefer to use **FindRoot**, giving a reasonable initial guess based on the results from the iterations.

In[48]:=
```
λ /. FindRoot[{f4[x] == x, f4'[x] == -1}, {x, 0.88}, {λ, 0.9}]
```

Out[48]=
```
0.886023
```

For $\lambda > 0.886023$ the 4-cycle gives way to an 8-cycle. We can also find where the higher order cycles become unstable, though the calculations become increasingly complicated and require quite accurate initial guesses for the parameters λ and x. For, example, to determine where the 8-cycle becomes unstable I find

In[49]:=
```
Clear[λ]
```

In[50]:=
```
f8[x_] = f4[f4[x]];
```

In[51]:=
```
λ /. FindRoot[{f8[x] == x, f8'[x] == -1}, {x, 0.891}, {λ, 0.891}]
```

Out[51]=
```
0.891102
```

To conclude this section, we have seen that if the results of successive iterations of the map suggest that there is convergence to a fixed point, or limit cycle of a certain length, it is useful to confirm this hypothesis with the **Fixed-Point** command. We can also use the stability criterion |fn'(X)| < 1 (where fn(x) is the n-th iterate of f(x)) to determine where cycles of length n become unstable, using *Mathematica*'s commands, **Solve**, **NSolve** and **FindRoot**.

General Behavior

We now investigate systematically the behavior as a function of λ. It is convenient to define a function which will iterate the map n times. Furthermore, the ultimate behavior, such as fixed point or limit cycle, only appears after an initial number of iterations during which the x values are in the process of "converging" to the limiting behavior. The length of this "initial transient" increases as one approaches a bifurcation point where the period doubles. We therefore want our function to drop the first m, say, iterations and only print a list of the remaining n-m values. The former is accomplished by the *Mathematica* commnd **Drop** so we define

In[52]:=
```
iterate[m_, n_] := Drop[NestList[f, 0.5, n], m]
```

The initial value of x has been set to 0.5. We will use *Mathematica* commands to display the points in the (x, λ) plane. We display a point by the command **Point**, so we define the following function:

In[53]:=
```
drawpt[y_] := Point[{λ, y}]
```

The result of a the command **Point,** and related commands, is a "graphics element". Graphics elements can be combined together with the **Graphics** command to produce a "graphics object" which can then be plotted with the command **Show**.

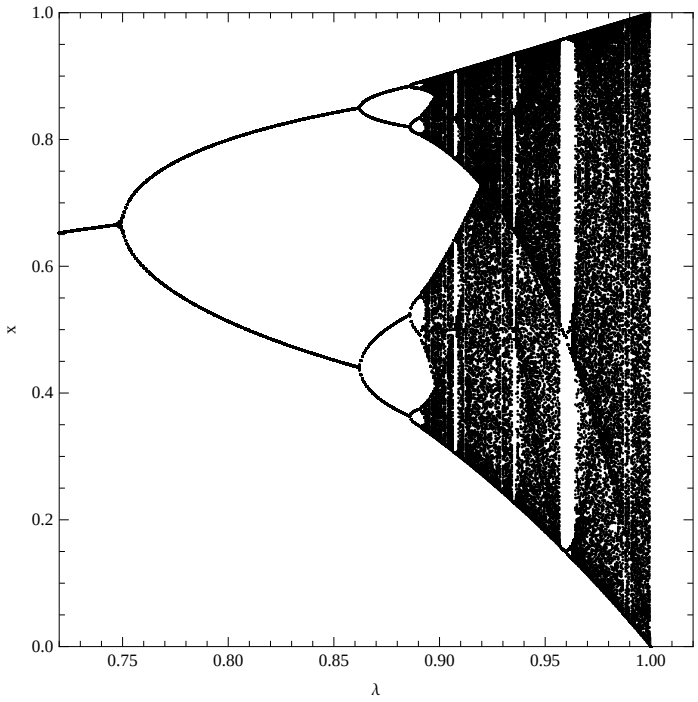
Here, we want to plot a point for each of the values of x generated by our function **iterate**. For this we use the *Mathematica* function **Map[drawpt, list]** which applies (maps) the function **drawpt** to each element of **list**. The following function, **graph**, sets up the lists of points that will be plotted:

In[54]:=
```
graph[λmin_, λmax_, nλ_, mdrop_, n_] := Graphics[{PointSize[0.001],
   Table[Map[drawpt, iterate[mdrop, n]], {λ, λmin, λmax, (λmax - λmin) / nλ}]}]
```

This takes nλ values of λ between λmin and λmax and, for each of these, makes a list of the first n iterates of f, starting with x=0.5, but dropping the first mdrop values. These points can now be plotted using the **Show** command below, in which we have also given some useful plot options.

In[55]:=
```
Show[graph[0.72, 1, 400, 300, 700], Axes → False, Frame → True,
 FrameLabel → {"λ", "x"}, PlotRange → {{0.72, 1.02}, {0, 1}}, AspectRatio → 1]
```

Out[55]=

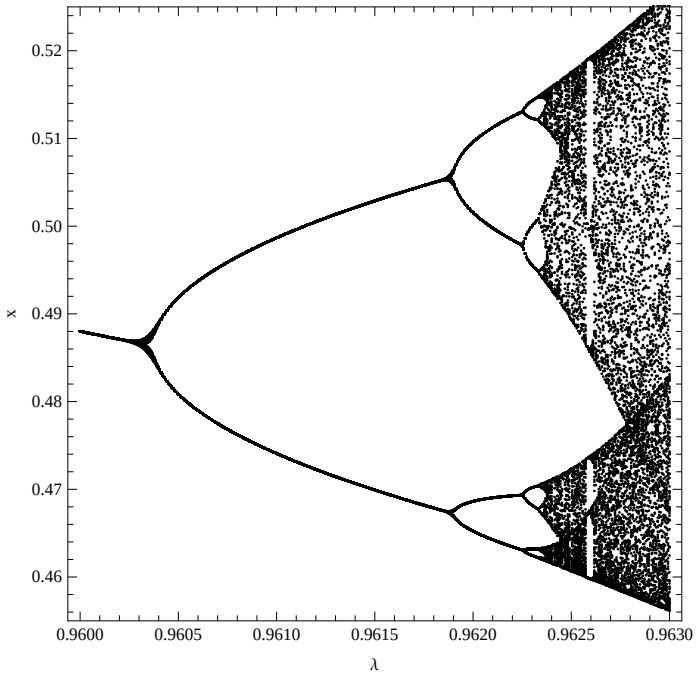


This figure just shows the region for $\lambda > 0.72$, which is the most interesting part. For smaller values of λ one always has a fixed point, which is at 0 for $\lambda < 0.25$ and at a non-zero value for $0.25 < \lambda < 0.75$. This figure just shows the *attactors*, the sets of values of x towards which the iterations *converge*, for different values of λ. There can also be *unstable* fixed points and limit cycles, which we don't see. For example the fixed point which is stable for $\lambda < 0.75$, continues smoothly for $\lambda > 0.75$, but it becomes unstable, so the iterations don't flow towards it and hence it does not appear on the plot for $\lambda > 0.75$. Instead, the *stable attractor*, a length-2 cycle, appears in the figure for λ just greater than 0.75.

Notice how rich the behavior is for $\lambda > 0.75$. We not only see the **period doubling cascade**, discussed above, starting at 0.75 and leading to chaos at about 0.89, but in addition there are "*windows*" of limit cycle behavior even in the region from 0.89 to 1. The largest of these, which starts at $\lambda = (\sqrt{8} + 1)\Big/4 = 0.9571...$ has a **period 3 limit cycle**, which then undergoes **period doubling** with periods 6, 12, 24 etc. until one enters again into a chaotic regime. The plot below, which enlarges this region, shows the period doubling cascade of the middle branch of the period 3 limit cycle,

In[56]:=
```
Show[graph[0.96, 0.963, 400, 300, 700], Axes → False, Frame → True,
 FrameLabel → {"λ", "x"}, PlotRange → {0.455, 0.525}, AspectRatio → 1]
```

Out[56]=



Notice that, even in this greatly enlarged view, one sees the same period doubling structure on smaller and smaller scales. One can continue to expand the scale of the plot to show even smaller scales, limited in practice only by numerical precision.

Note too that there is structure in the density of points even in the chaotic regime.

The separations between successive values of λ where the period doubles get closer and closer together as the period increases. Let us define λ_k to be the value of λ at which the 2^{k-1}period becomes unstable resulting in a 2^k cycle, so, for example, $\lambda_0 = 1/4,\ \lambda_1 = 3/4,\ \lambda_2 = \left(\sqrt{6}+1\right)\Big/4 \simeq 0.8624...$, $\lambda_3 \simeq 0.8860$. Successive differences λ_k - λ_{k-1} are then in a geometric ratio (at least for large k), and the inverse of this, denoted by δ, is called the *Feigenbaum constant*, i.e.

$$\delta = \lim_{k\to\infty} \frac{\lambda_k - \lambda_{k-1}}{\lambda_{k+1} - \lambda_k}$$

Its value is about 4.6692... . This number is important because it is "**universal**", i.e. the same value is obtained for a wide range of functions (actually any function with a quadratic maximum), not just f(x) = 4 λ x (1 - x). Furthermore, for any given function, it is the same for *all* the period doublings which occur at different parameter values (here the parameter varied is λ).

To conclude this section, who would have thought that the simple looking map in the first equation would have the amazingly rich behavior shown in the last two figures. This richness was only fully appreciated once computers and computer graphics had developed to the point that figures like these could readily be produced.

Lyapunov Exponent

A defining feature of a chaotic system is **sensitivity to initial conditions**. If two trajectories which start off close to each other deviate more and more with increasing time, the system is said to be **chaotic**. The rate at which nearby trajectories deviate from each other with time is characterized by a quantity called the *Lyapunov exponent*. Here we

discuss the Lyapunov exponent for the logistic map.

Consider two iterations of the logistic map starting from two values of x which are close together. Let the two starting values be x_0 and $x_0 + \delta x_0$. These map to x_1 and $x_1 + \delta x_1$, ..., $x_n + \delta x_n$. Expanding f(x) about $x_n$we have

$$\delta x_n = f'(x_{n-1})\, \delta x_{n-1}$$

assuming that δx_n is sufficiently small. Hence the separation of two trajectories after n steps, δx_n, is related to their initial separation, δx_0, by

$$\left| \frac{\delta x_n}{\delta x_0} \right| = \prod_{i=0}^{n-1} \left| f'(x_i) \right| .$$

We expect that this will vary exponentially at large n like

$$\left| \frac{\delta x_n}{\delta x_0} \right| = e^{\lambda_L n} \quad (\text{large } n)$$

and so we define the **Lypunov exponent** λ_L by

$$\lambda_L = \lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^{n} \ln \left| f'(x_i) \right| .$$

If $\lambda_L > 0$ neighboring trajectories diverge from each other at large n, which corresponds to chaos. However if the trajectories converge to a fixed point or limit cycle they will get closer together, which corresponds to $\lambda_L < 0$.

Hence we can determine whether or not the system is chaotic by the sign of the Lyapunov exponent.

Below we calculate the Lyapunov exponent for some values of the parameter λ (not to be confused with the Lyapunov exponent λ_L).

The Lyapunov exponent, which we call lya, is obtained by generating a list of x_i using **NestList.** The average of the elements of the a list can be very elegantly calculated in *Mathematica* using **Apply[Plus, list]** and dividing by **Length-[list]**. Hence the average of ln | f'(x)| is given by

In[57]:=
```
lya[l_, xinit_, n_, ndrop_] := (λ = l; xlist = Drop[ NestList[f, xinit, n] , ndrop + 1 ];
  Apply[ Plus, Log[ Abs[ f'[xlist] ] ] ] / Length[xlist])
```

The following determines λ_L for λ=0.91, starting with x_0=0.7, and averaging over 50000 iterations of the map, except for dropping off the first 20 to allow for an initial transient.

In[58]:=
```
lya[0.91, 0.7, 50000, 20]
```

Out[58]=
```
0.232347
```

The positive value indicates that λ=0.91 is in a region of chaos. By contrast if we specify λ = 0.78

In[59]:=
```
lya[0.78, 0.7, 50000, 20]
```

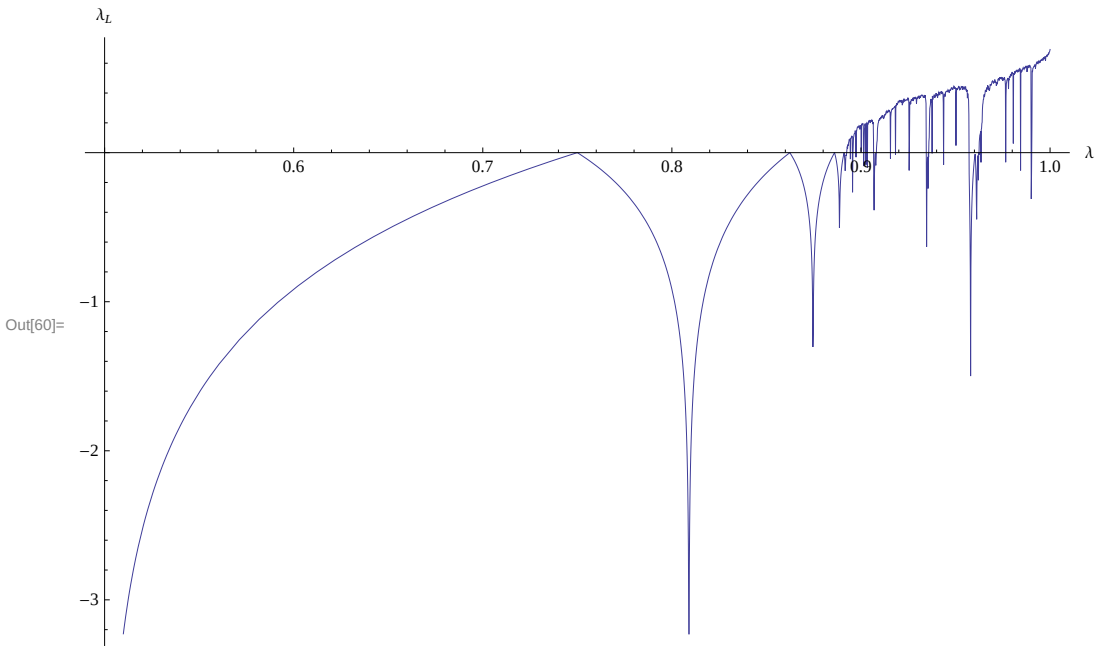
Out[59]=
```
-0.340998
```

we get a negative value, indicating that the trajectory of points x_i, (i = 0, 1, 2, ... ,) converges to an attractor, which, in this case we already know is a length 2 limit cycle.

Next lets plot the Lyapynov exponent for a range of values of λ (this is slow)

In[60]:=
```
Plot[lya[λ, 0.7, 5000, 2000], {λ, 0.5, 1.}, AxesOrigin → {0.5, 0}, AxesLabel → {"λ", "λL"}]
```

Out[60]=



The main significance of this figure is that one can easily distinguish the regions which are chaotic ($\lambda_L > 0$) from the regions which tend to a fixed point or limit cycle ($\lambda_L < 0$). You see several points (the first is at $\lambda = 0.75$) where the Lyapuov exponent hits 0 and then goes negative again. These are the period doubling bifurcations. Precisely at the period doubling point the system is at the limit of chaos, but then becomes non-chaotic when the period doubles. However, at the end of the period doubling regime, at λ about 0.8922, λ_L crosses the axis and the system enters a chaotic regime.

Now let's speed things up with a compiled version in which the function **f[x]** and its derivative are hard wired in. In the **Compile** command, the first argument is a list of variables which are assumed to be numerical, and the second argument is the function. It may be necessary to declare certain variables to be of a particular type such as integer. In this case the variable name is replaced by a list of two elements, the first being the variable and the second element the data type such as **_Real** or **_Integer**, e.g. {n, _Integer}. In the following, it is necessary that n and ndrop are declared to be integer:

In[61]:=
```
lyaC = Compile[{ λ, xinit, {n, _Integer}, {ndrop, _Integer}},
   xlist = Drop[ NestList[4 λ # (1 - #) &, xinit,  n] ,  ndrop + 1 ];
   Apply[ Plus, Log[ Abs[ 4 λ (1 - 2 xlist)] ] ] / Length[xlist] ];
```

Note that the function f(x) = 4 λ x(1 - x) is represented as a pure function (without a name) whose argument is represented by **#** and which is terminated by **&**.

In[62]:=
```
lya[0.9, 0.7, 50 000, 100]  // Timing
```

Out[62]=
```
{0.10782, 0.181632}
```

In[63]:=
```
lyaC[0.9, 0.7, 50 000, 100]  // Timing
```

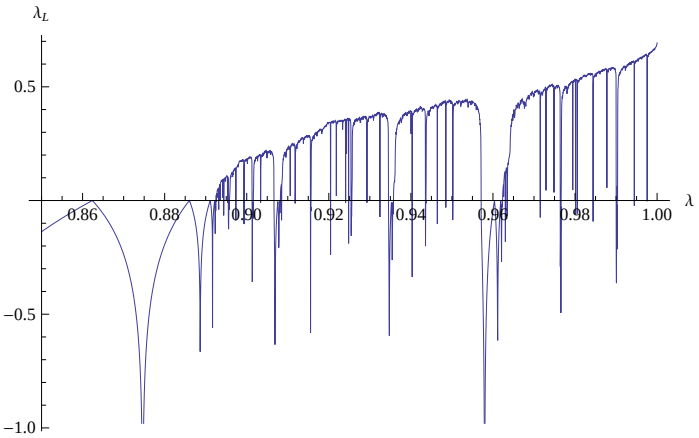
Out[63]=
```
{0.007482, 0.184141}
```

We see that the compiled version is significantly faster (the time is the first element of the list in the output.)

Next we blow up the chaotic region.

In[64]:=
```
Plot[lyaC[λ, 0.7, 30000, 5000], {λ, 0.85, 1.},
 AxesOrigin → {0.85, 0}, AxesLabel → {"λ", "λL"}]
```

Out[64]=

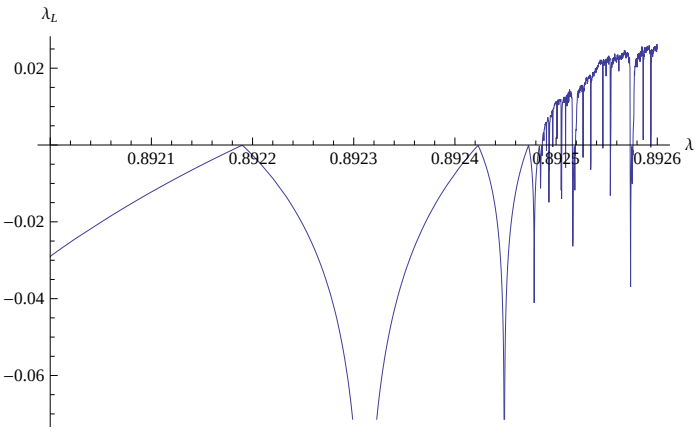


Note that for λ in the range greater than the point where λ_L first goes positive, there are many regions where λ_L is negative, "islands of stability" where the behavior is fixed point or limit cycle.

Now we home in on the principle period doubling transition to chaos by enlarging the scale.

In[67]:=
```
Plot[lyaC[λ, 0.7, 30000, 5000], {λ, 0.892, 0.8926},
 AxesOrigin → {0.892, 0}, AxesLabel → {"λ", "λL"}]
```

Out[67]=

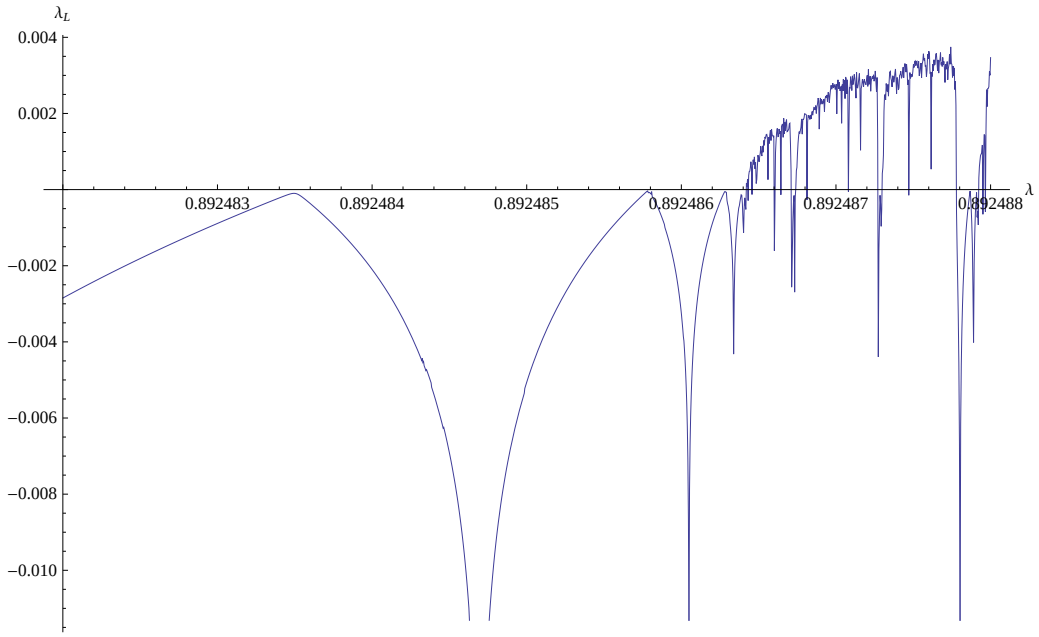


In the above figure, the first value of λ where λ_L hits zero is $\lambda_5 = 0.89219\ldots$. To the left of this zero the stable attractor is a period 16 cycle, and between that zero and the next one at $\lambda_6 = 0.89242\ldots$ we have a period 32 cycle.

We now enlarge the scale again, (this figure may take some time).

In[68]:=
```
Plot[lyaC[λ, 0.7, 50 000, 15 000],
 {λ, 0.892482, 0.892488}, AxesLabel → {"λ", "λL"}, PlotPoints → 200]
```

Out[68]=



Notice that there is fine detail no matter how much one expands the scale. We see that chaos emerges (i.e. $\lambda_L > 0$) for λ between 0.892486 and 0.892487.

Numerical Precision in the Chaotic Regime

Consider the logistic map in the chaotic regime. We know that two trajectories which start off close together get further and further apart on iterating the map. The rate of separation of the trajectories is determined by the Lyapunov exponent λ_L, i.e.

$$|\delta x_n| = e^{\lambda_L n} |\delta x_0|$$

Hence any roundoff error in the calculation will also grow like $e^{\lambda_L n}$. If ϵ_m is machine precision, then, when $\epsilon_m e^{\lambda_L n}$ is about 1, all precision will have been lost and the results will be just noise. **Hence it is not possible to predict the value of x_n for very large n in the chaotic regime.** This is analogous to being unable to predict the weather arbitrarily far into the future.

To illustrate the loss or precision, we will take $\lambda = 1$, for which the Lyapunov exponent is known to be

$\lambda_L = \log(2)$,

so

$$|\delta x_n| = 2^n |\delta x_0| \quad (\text{large } n).$$

Let's verify this numerically:

In[69]:=
```
{lyaC[1, 0.7, 200 000, 15 000], N[Log[2]]}
```

Out[69]=
```
{0.693149, 0.693147}
```

The numerical value of the Lyapunov exponent is very close to log 2, as claimed.

We define the map and set $\lambda = 1$:

In[70]:=
```
f[x_] := 4 λ x (1 - x)
```

In[71]:=
```
λ = 1;
```

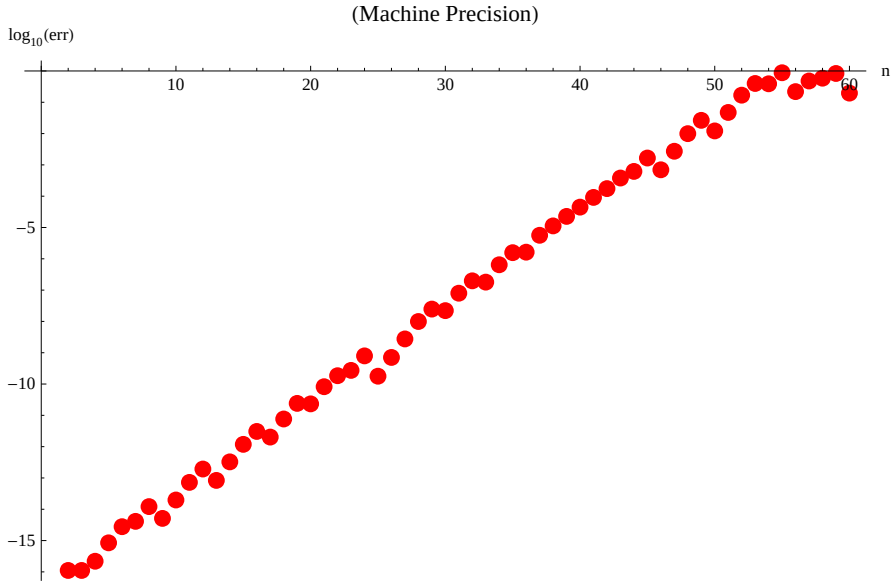
We calculate the error in the result, using machine precision, after n iterations as a function of n.

In[72]:=
```
diff[n_] := N[ Nest[f, 0.3, n] - Nest[f, N[3 / 10, 70], n] ]
```

In the line above, the first term on the right hand side is the result using machine precision, and the second is the result using 70 digits of precision (which is essentially exact by comparison). The starting value is x = 3/10. We now plot how the error increases with n.

In[73]:=
```
err = ListPlot[Table[{n, Log[10, Abs[diff[n]]]}, {n, 2, 60}], AxesLabel → {"n", "log10(err)"},
  PlotStyle → {PointSize[0.02], Hue[0]}, PlotLabel → "(Machine Precision)"]
```

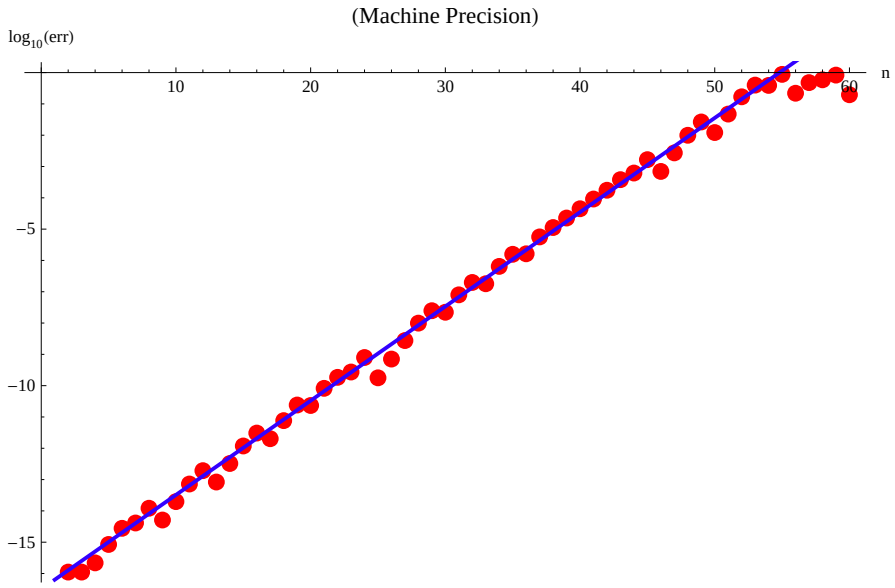
Out[73]=



The vertical axis is $\log_{10}$(err) which is the negative of the number of digits of accuracy (we will call the number of digits of accuracy the *precision*.) After about 55 iterations all precision has been lost. The lost of accuracy is just what one expects with an initial precision of a little better than 16 decimal places and an error growing with the Lyapunov exponent log(2). The following is a plot of y = -16.5 + $\log_{10}$(2) x (note that $\log_{10}$(2) = 0.30103):

In[74]:=
```
lin =
  Plot[-16.5 + Log[10, 2] x, {x, 1, 60}, PlotStyle → {AbsoluteThickness[2], Hue[0.7]}];
```

```
Show[err, lin]
```



It fits the data well:

Alternatively, we can use *Mathematica*'s arbitrary precision numbers. In this case *Mathematica* only keeps the digits that it is *sure* are correct. We can start with much higher precision, essentially whatever we want, but the precision that *Mathematica* claims in subsequent results goes down faster with increasing n than what we found above using machine precision. The difference occurs because, with arbitrary precision numbers, if *Mathematica* is not *sure* of the correctness of higher order digits it does not give them.

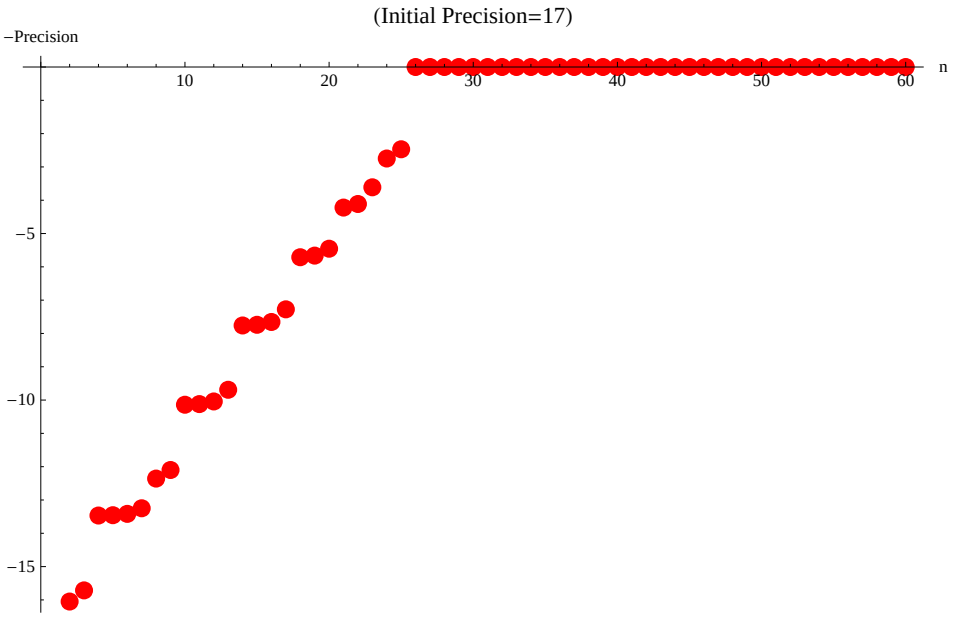
Below we take the initial precision to be 17, hardly more than the default precision of 16, but still (since > 16) an arbitrary precision number.

In[75]:=
```
prec = ListPlot[Table[{n, -Precision[Nest[f, N[3/10, 17], n]]}, {n, 2, 60}],
  AxesLabel → {"n", "-Precision"}, PlotStyle → {PointSize[0.02], Hue[0]},
  PlotLabel → "(Initial Precision=17)"]
```

General::ovfl : Overflow occurred in computation. ≫

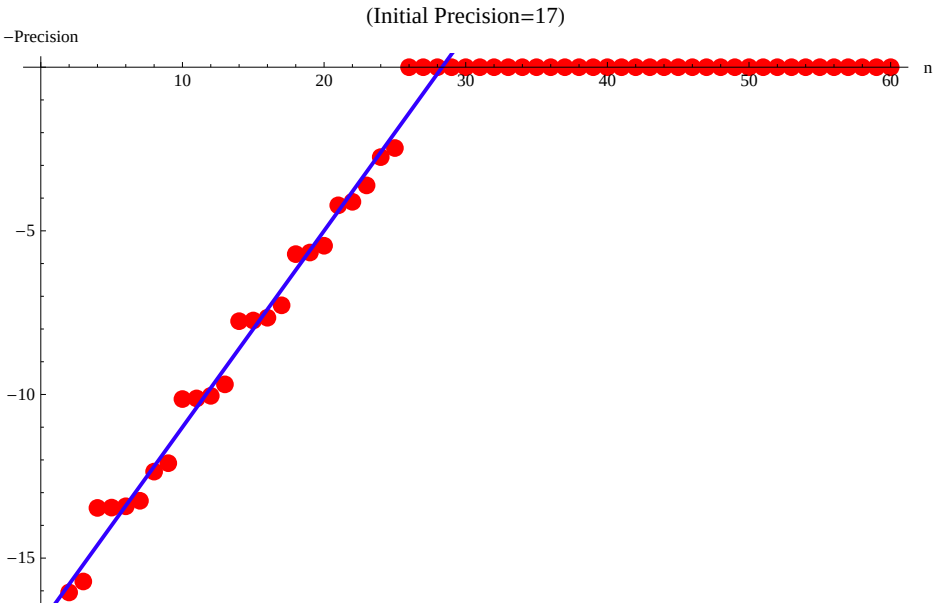
General::ovfl : Overflow occurred in computation. ≫

Out[75]=



We see that the (*guaranteed*) precision is lost about twice as fast as the (*not-guaranteed*) precision found above with machine precision numbers. Here is a fit, with slope 0.6:

In[76]:=
```
lin2 = Plot[-17 + 0.6 x, {x, 1, 32}, PlotStyle → {AbsoluteThickness[2], Hue[0.7]}];
Show[prec, lin2]
```



Finally we show the precision claimed by *Mathematica* after 200 iterations, as a function of the *initial* precision.

In[77]:=
```
ListPlot[Table[Precision[Nest[f, N[3/10, n], 200]], {n, 5, 150}],
 PlotStyle → {PointSize[0.02], Hue[0.7]},
 AxesLabel → {"precision", "final precision"}, PlotLabel → "(200 iterations)"]
```

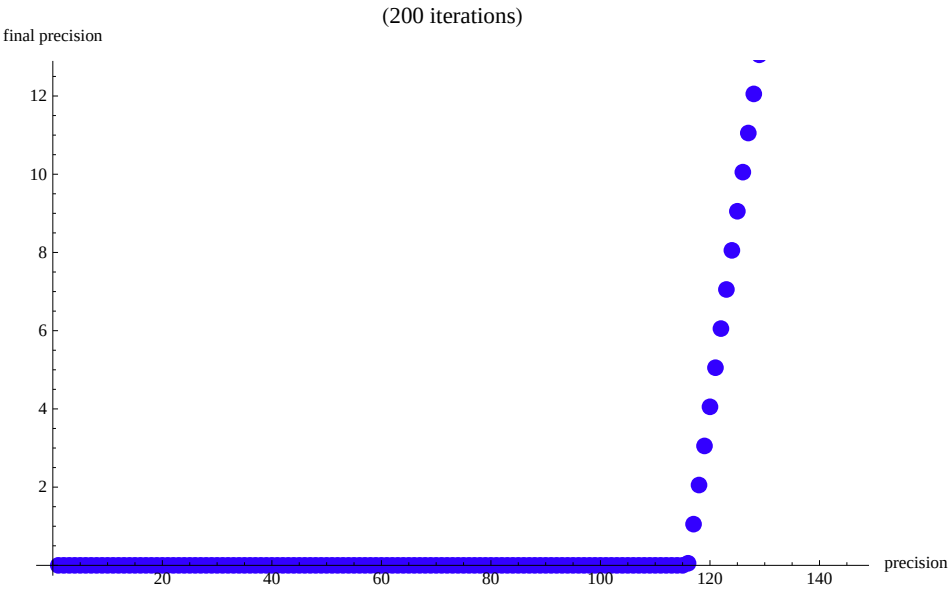
General::ovfl : Overflow occurred in computation. >>

General::ovfl : Overflow occurred in computation. >>

General::ovfl : Overflow occurred in computation. >>

General::stop : Further output of General::ovfl will be suppressed during this calculation. >>

Out[77]=



We see that at least 120 digits of initial precision are necessary to determine $x_{200}$from x_{n+1}= 4 x_n (1 - x_n) (with x_1= 3/10).

Overall, we have found empirically that the number of digits of precision which must be specified is about 0.6 of the number of iterations. Hence, to determine is $x_{10\,000}$, we need about 6000 digits. The answer, according to my computer, is 0.104373:

In[78]:=
```
N[Nest[f, N[3/10, 6100], 10000]] // Timing
```

Out[78]=
```
{0.167607, 0.104373}
```

What about $x_{100\,000}$? I found 0.625851 (it takes quite a long time) :

In[79]:=
```
N[Nest[f, N[3/10, 61000], 100000]] // Timing
```

Out[79]=
```
{35.7088, 0.625851}
```

What is $x_{1\,000\,000}$? The calculation is extremely slow. A student in a previous year's class, Dustin Gilbert, ran his computer for a long time to get the answer, and here it is, 0.751139. (Because it takes so long to execute, this cell is currently set to not execute when the notebook is evaluated. This option can be changed by selecting the cell by clicking on its vertical bar on the right, going to the Cell menu, and selecting Cell Properties.)

```
N[Nest[f, N[3/10, 610000], 1000000]] // Timing
```

```
{47924.1, 0.751139}
```

The calculation took 47,924 secs, a bit over 13 hours!