Unit XII
A Touch of Chaos

Unit XII
A Touch of Chaos

It is one of the hot phrases in science these days. It has even infected popular movies and novels like Jurassic Park. But what is this chaos stuff anyway? And how does it relate to all these pretty pictures known as fractals? In this unit we will learn just a little bit about chaos and those pictures, and a little bit about how some chaotic systems are calculated.

One basic requirement for chaotic behavior is non-linearity of the governing equations. This means that one cannot describe the phenomena by just adding together some set of basic solutions to produce a general solution, as we do with for example the harmonic oscillator, or conventional waves, or the solution to a constant force law. As a result, the solutions can't be easily predicted; they may depend very sensitively on starting conditions or on some of the parameters of the system. It is almost as extreme in some cases as if a tiny change in the value of g completely changed whether things fell up or down!

We'll consider a very simple equation that determines our dynamics, from a class known as logistic difference equations. This particular equation was one of the first that gave evidence of this surprising behavior we now call chaos. It was used in this particular example taken from James Gleick's book *Chaos* as an algorithm that describes population biology. In this example, first studied in detail by population biologist Robert May, population changes in intervals of a breeding cycle, and the population in the next cycle depends positively on the population (more breeders) and yet negatively on the population (limited resources in the environment). The simple form of this is to take the product of a linear increase term (proportional to the population x, with a presumed maximum possible value of 1) and a limited resources term proportional to 1-x. This is all then scaled by some scale constant r that indicates how big these effects are. In other words, we have an iterative calculation:

$$x_{new} = r x_{old}(1 - x_{old}).$$

It would certainly seem at first blush that the behavior of this system would depend very little on r, except perhaps in how quickly equilibrium is reached. However, this looks like an iterative way to calculate the equilibrium population $x_{equilibrium}$, and as we know, iterative calculations may or may not be stable.

Guidebook Entry XII.1: Iterations Again!

First, show that the solution of this equation for the equilibrium population is trivial--find $x_{equilibrium}$ analytically.

However, just because we have the equilibrium value, this does not tell us how quickly we get to that equilibrium, or whether that equilibrium is stable or not. Let's use Excel to investigate this iterative calculation. Make a spreadsheet that contains the parameter "r" in a convenient cell near the top left--I'd suggest A2. Let r start out as 2. What should the equilibrium value for x be?

Put in a starter value for x in B2, let's say 0.5 (half populated). Now, in C2, put the formula for the next iteration:

$$x_{new} = r x_{old}(1 - x_{old}).$$

Make sure to make the reference to r absolute in terms of letter, but NOT in terms of number, that is $A2. Now fill this to the right for 50 or 100 cells. Does it converge to the expected equilibrium value?

Graph the expected equilibrium values of x as a function of r from r just above zero to r = 4. You may use Excel to do this, or Mathematica, or another package, or do it by hand. Sketch the result below.

Now see if your iterated calculation converges to the expected value at some r value between zero and one, say 0.5. What do you find?

Try changing r to exactly 3. Does the iteration converge to the expected value? If not, describe how it does behave.

Check the behavior at r = 3.50. Describe what you find.

Now move up to r = 3.75. How would you characterize the iteration. Does it converge? Does it diverge?

You should have now seen what is known in the chaos business as repeated period doubling, or bifurcation, leading into chaos. But lest you think that you have seen all the surprises, let's check a few other values of r "by hand." Describe what you find at r = 3.82:

Then at r = 3.84:

And finally at 3.86:

All of this fascinating behavior has some constraints, at least. Try r = 4.1, and describe the results.

To appreciate the complexity of this behavior a little better, it is convenient to create a graph that is like the graph you made of $x_{equilibrium}$ as a function of r, except now we plot whatever values of x the interation settles upon rather than the equilibrium value we expect it to find. To do this, you will create a whole set of rows of iterated calculations.

Guidebook Entry XII.2: Creating Robert May's Map

You are going to make a whole series of iterations, each with a different r value. First, choose a spacing between r values, call it Δr, and place that in a convenient cell near the top left, say at A1 (now you see why I suggested you start in the second row!). Make the r value for the second row of iterations (which you probably have in A3) equal to the cell above plus the Δr value. Make sure to have an absolute reference to Δr. Now you can fill down a bunch (not more than 100 or the sheet will get too slow) of r values. Make sure to start the r's somewhere interesting (maybe 2?) and end a bit before 4 by choosing your Δr value carefully. Now you can take the iteration algorithm, and fill it down. Make sure each row goes for about 100 iterations (so 100 cells across) to make sure it has a good chance to converge. Now to create the graph, make an x-y (scatter) plot of the last ten or twenty columns versus the r column. This will then give a plot that follows your $x_{equilibrium}$ graph where the iteration converges, but also give you a visual way of inspecting the period doublings. Attach your graph.

Now you have seen some of the important features of chaotic dynamics: period doubling, order and chaos, sensitive dependence on parameters (and in some cases initial conditions). There is one more feature we can easily examine with this example: self-similarity. In other words, if we take a microscope, and look closely at our graph, we find small replicas of the large graph hidden in the bigger picture, much like parallel mirrors in a fun house or an Escher drawing. To look at this, we will use a rather primitive Pascal program that draws May's map more quickly than Excel.

Guidebook Entry XII.3: Self-similarity

On your hard drive you should find a program called "MayDraw." The program is pretty self-explanatory, now that you know what the map is about. You get to choose starting and ending r values, and number of startup iterations, and then the number of iterations graphed after that. First, graph the whole figure from 2 to 4. Sketch what it looks like below. You may want to play with the number of iterations to make the graph better.

A mouse click causes the program to exit. Start it again. Now concentrate on the region about 3.8. Expand this sufficiently to convince yourself there is a self similar region in the center of this. You may need to try several times, and may want In a homework problem, you will use Excel or Mathematica to look at this more carefully. Sketch what you see here.

Finally, let's use Mathematica to look at this same map. In the process, we will get a taste of programming techniques in Mathematica. To make some sense of the commands, it will help to know that a ";" allows you to string several commands together in the place of a single command, and that "For" and "Do" are commands that allow you to perform repetitive calculations. Feel free to ask questions, or check things out in the Mathematica book.

Guidebook Entry XII.4: Chaos in Mathematica

Enter the following simple Mathematica program. Follow each line with a "return" rather than an "enter" until you reach the end so that execution is delayed until you are finished:

```
mylist = {};
For[r=2,r<4,r+=0.02,
     x=0.5;
     Do[x=r*x*(1-x),{j,50}];
     Do[x=r*x*(1-x);
          AppendTo[mylist,{r,x}],
          {j,30}
     ];
];
ListPlot[mylist,Axes->None]
```

Where is the number of startup iterations specified? Where are the number of graphing iterations specified?

Give an enter, and wait a reasonable time (a couple of minutes). If all went well, it should produce a copy of the May map. Print a copy out and attach it.

What happens if you make the startup iterations very few, like 10? Where do you see the effects most? You may need to magnify a region to see more clearly.

Why would this effect make it hard to find the period doubling points accurately?