Explorations in chaos

An iterated function system is any system where the output of a function is immediately placed back into the input of a function. For example, a simple system might be defined as follows.

$$F_{n+1} = (F_n)^2$$

This means that to find the next value of the sequence, one needs to square the current value. The subsequent value would be found by, again, squaring the result.

When we study iterated function systems, we don't usually care *what* the result of the function is. Instead, we care about *how many times* we have to run some initial value through the system for the result to have some property.

In the example above, if $F_0 = 2$, then $F_1 = 4$, $F_2, = 16$, $F_3 = 256$, etc. We might be interested in, say, how long it takes for F to be greater than 100 (in this case, three iterations), but we don't care what the actual value is at this point.

You might have noticed that this system is rather boring. F_n grows very quickly, as long as $F_0 > 1$. It doesn't matter what your F_0 is: it can be 3, it can be 3.01, or it can be 3.0000001, the end result is the same: the value of the system becomes very large very fast.

However, when one studies complex numbers in an iterated function systems, the game changes. Systems of iterated functions involving complex numbers are significantly more interesting than those with real numbers, and the results are significantly more striking. The next few paragraphs give a primer on complex numbers—skip them if you are already familiar with complex numbers.

For centuries, mathematicians struggled with the idea of the non-real roots of some quadratic equations. For example, what are the roots of $x^2 + 1 = 0$? According to the Dr. Math web site[1], it was the Italian mathematician Girolamo Cardano, who, in the 16th century, first formulated the idea of complex and imaginary numbers as non-real roots to quadratic equations. Carl Gauss is largely credited with the use of *i* the square root of one.

So what is an imaginary number? An imaginary number tackles the problem of taking the square root of a negative number. The square root of -1, for example, is not -1 (because -1 squared is 1), and it's obviously not +1. So, by convention, mathematicians define *i* to be the square root of -1, and in this manner we can express the square root of any negative number:

$$\sqrt{(-27)} = \sqrt{[(27)(-1)]} = \sqrt{(27)}\sqrt{(-1)} = 3i$$

[1] http://mathforum.org/library/drmath/view/52584.html

Complex numbers, then, are a natural extension of imaginary numbers. Complex numbers have a real part (a) and an imaginary part (b) in the form $a + bi$. For example, we might solve the quadratic equation $x^2 - 4x + 5 = 0$ using the quadratic formula:

$$x = (-b \pm \sqrt{(b^2-4ac)})/2a$$
$$x = (4 \pm \sqrt{(16-20)})/2$$
$$x = (4 \pm \sqrt{(-4)})/2$$
$$x = (4 \pm 2i)/2$$
$$x = 2 \pm i$$

Here, there are two complex solutions to the quadratic equation, namely, $2+i$ and $2-i$.

Bearing in mind that i is the square root of -1, we can perform any operation we'd like on complex numbers (though division is a little more complicated). To add or subtract complex numbers, we add and subtract their real and imaginary parts.

$$(3+2i) - (4-3i) = -1+5i$$

To square a complex number $a+bi$ (or to multiply any two complex numbers), we simply multiply as though we were multiplying two binomials.

$$\begin{aligned} (a+bi)(a+bi) &= a^2 + 2abi + b^2i^2 \\ &= a^2 + 2abi - b^2 && \text{(because } i^2 = -1\text{)} \\ &= (a^2 + b^2) + (2ab)i && \text{(collecting like terms)} \end{aligned}$$

This implies a theorem of closure over the complex numbers, which I will state but not prove. *The set of all complex numbers is closed under the operations of addition, subtraction, multiplication, and division*. Simply put: take any two complex numbers, and add, subtract, multiply, or divide them, and your result will be another complex number.

Finally, we need to define the *norm* of a complex number. The norm is analogous to the absolute value of a real number, and its formula looks very similar to the distance formula:
$\text{norm}(a+bi) = ||a+bi|| = \sqrt{(a^2+b^2)}$

Now let's examine an iterated function system of complex numbers.

$$C_{n+1} = (C_n)^2$$

With complex numbers, this system is much more interesting! If we let $C_0 = 1-2i$, then:

$C_1 = 1-2i$
$C_2 = -3-4i$
$C_3 = -7+24i$
$C_4 = -527-336i$
$C_5 = 164833+354144i$

The numbers behave a little more erratically! Both the real and imaginary parts have switched their signs at least twice, without any seeming pattern.

Consider a similar iterated function system of complex numbers.

$C_{n+1} = (C_n)^2 + C_k$ where C_k is a complex constant

By choosing different values for C_k, our system will have different properties. Let's try $1+i$ as C_0 and $-1-i$ as C_k:

$C_1 = -1+i$
$C_2 = -1-3i$
$C_3 = -9+5i$
$C_4 = 55-91i$
$C_5 = -5257-10011i$

For a given iterated function system, we can arbitrarily select a number (by convention, we use 2) and see how long it takes for the norm of C_n to exceed 2. In the example above, $||C_1|| = 1.4$, and $||C_2|| = 3.2$. Therefore, the value of the system exceeds 2 after two iterations.

For smaller numbers, it can take much longer. Let $C_0 = .25-.3i$ and let $C_k = -.25-.25i$.

$C_1 = -.28-.40i$
$C_2 = -.33-.03i$
$C_3 = -.14-.23i$
$C_4 = -.28-.19i$
$C_5 = -.20-.14i$
$C_6 = -.23-.19i$
$C_7 = -.23-.16i$
$C_8 = -.22-.17i$
$C_9 = -.23-.17i$
$C_{10} = -.23-.17i$

Clearly, even $||C_{10}||$ does not even exceed 2. Will the norm of the result ever exceed 2 for these values? We cannot say without further calculations.

Now, consider the complex coordinate plane, or, more precisely, a small section of the coordinate plane. To be precise, consider the section from $a = -2$ to $a = 2$, and from $b = -2$

to $b = 2$. Make a grid of points in the plane, each of them a tiny—but discrete—amount apart, say, at regular intervals of .01 units. The grid will have 201^2=40401 points.

For every point in the grid (does that give you any ideas how you might go about this calculation?), let that value be C_0, and run C_0 through the iterated function system above with the above value of C_k. Count the number of iterations each point requires for the norm of the system to exceed 2. Some points will very quickly exceed this value, others will take a long time—in fact, some points will never reach 2 (like $0+0i$). If we go through, say, 64 iterations and we haven't reached 2, we'll quit, and say it'll never reach 2 (which may or may not be true, but for our purposes, 64 is plenty).

Now we have a single value that corresponds to each point on the plane. We make a chart of values and colors (for example, 1=red, 2=red-orange, 3=orange, 4=orange-yellow, etc., and 64=black). Then, we take a chunk of the computer screen 201 by 201 pixels large, and put *that* point of color at the pixel corresponding to the given point. Voila! What an interesting image!

Right?

Well, yes. If you choose the right value for C_k, you will get some incredibly interesting images that are *self-similar*, meaning that if you zoom in far enough at the right spot, you'll see the original image! These are called *fractal images*, and they employ principles of *chaos theory*—more on that at some time later. For now, the main consequence of chaos theory is this: for many values of C_0, the results will be *significantly* different if C_0 is off by even the slightest bit.

Now go write a program to do all this. Today.

Just kidding.

But I encourage you, if this sounds interesting, to pursue fractal imagery or chaotic systems as a project for Leap.

In the mean time…write a program that does portions of the above algorithm.

At the very least, do this: write a program that takes two numbers, a, and b, and a complex constant $p+qi$, and determines how many iterations are required for the iterated function system $C_{k+1} = (C_k)^2 + (p+qi)$ to exceed a norm of 2.

If you're adventurous, do this for all those 40401 points and store the results in a two-dimensional integer array.

Don't worry about the graphics part of the algorithm. We'll discuss Java graphics on Monday.