Complex Positive Axes Systems

Robert Duncan

Positive axes coordinate systems were described in the paper ‘Positive Axes Systems’, also presented on this web site. Here we examine in some detail assigning complex algebras to the positive axes. First, a bit of nomenclature: conventionally a complex number has a ‘real’ part added to an ‘imaginary’ part, with the imaginary number being a real number times ‘the square root of negative one’. Quaternion mathematics makes use of three different types of ‘imaginary numbers’. We take a rather different approach: all numbers are positive real numbers but the algebra used is designed to treat different axes’ values differently. So we talk of ‘Red 3’ meaning 3 units on the Red axis. There are no square roots of negative numbers - rather, the algebraic manipulation of address coordinates depends upon the ‘color’ of axes involved. For example, the coordinate 3 G (green) times the coordinate 2 B (blue) yields 6 Y (yellow). This is similar to how complex numbers are used, but conceptually different. Also similar to how the complex numbers are used, the coordinates that make up an address can be viewed as either 4 distinct numbers - the distances measured along the axes - or as a single number - the sum of the coordinates. The addition of different coordinates is not defined, so, as with complex numbers, the sum can be indicated but not completed. Hence, ‘4 + 3i’ in the complex algebra, and ‘3R + 2Y + 1B’ in the positive axes system, are both single numbers. However, ‘(4, 3i)’ and ‘(3R, 1Y, 0G, 2B)’ are both addresses, not numbers. Positive axes systems can be real number axes only, or have the algebra vary according to which axes are involved in the computation. For simplicity’s sake we will refer to the later as ‘complex positive axes systems’.

In orthogonal systems such as the Cartesian system, a complex coordinate system can be formed only for 2, 4, and 8 spacial dimensions – not for 3-space. So a functional complex system in 3-space, like the one presented here, is interesting in its own right. We will use both squaring and cubing iterative functions, similar to the Mandelbrot function, modified to handle 4 coordinate addresses, with an algebra that simulates the transformations of multiplication by complex numbers. There are many such algebras we could use, so far I’ve investigated only a few, as described in this paper.

We can assign an algebra to the 3-axes 2-space plane (R, G, and B axes) that mimics the complex plane. One such algebra for multiplication transforms the result of a product like this: R times any ‘color’ axis does not alter the result and the product stays on the multiplicand (R times G is G). A G multiplier transforms any color multiplicand counter-clockwise 120 degrees (G times B is R). And a B multiplier transforms any color multiplicand clockwise 120 degrees (B times G is R). Multiplication in this system is commutative. Somewhat surprisingly the Mandelbrot set has exactly the same image

in this system as in the normal complex plane system, although the image is scaled differently. Of course, some operations are different. For example, Gaussian primes in the two systems are quite different.

The standard Mandelbrot iterative function is expressed as:

$$z_{(n+1)} = z_n^2 + c\,, \quad z = a + bi$$

The letter *i* is, of course, the square root of negative one, an imaginary number. Instead of representing a number, *i* is better viewed as a label of a number indicating a different algebra must be applied in its interactions with itself and other numbers. This different algebra is the familiar one: a real times an imaginary is imaginary, an imaginary times an imaginary is a negative real, etc. Using the symbol √(-1) is just an aid to remember the algebra needed to manipulate the real number that is labeled with *i*. Thus, $(3i)^2$ is translated into $3\sqrt{(-1)} * 3\sqrt{(-1)} = 9\sqrt{(-1)}\sqrt{(-1)} = 9(\sqrt{(-1)})^2 = 9 * -1 = -9$.

For the positive 4-axes system we could label each coordinate with a symbol indicating the need for a different algebra. For example, (3 R, 2 Y, 0 G, 4 bluB. Or, we could just always express the addresses as an ordered set of four numbers (3, 2, 0, 4), and know that a particular algebra applies to each of the coordinates. Here are the rules of for one algebra for complex computations in four positive axes 3-space:

Given two addresses (a, b, c, d) and (w, x, y, z), the relevant operations are defined as:

Addition: (a, b, c, d) + (w, x, y, z) = (a+w, b+x, c+y, d+z)

Scaler multiplication: k * (a, b, c, d) = (ka, kb, kc, kd)

Multiplication: (a, b, c, d) * (w, x, y, z) =
(aw + by + cz + dx,
az + bx + cw + dy,
ax + bz + cy + dw,
ay + bw + cx + dz)

Multiplication of addresses in this system is not commutative by this arbitrary rule, and this algebra causes symmetric transformations, often labeled (incorrectly) as ‘rotations’.

Of course, the iteration formula calls for squaring so the squaring formula is this:

Multiplication: (a, b, c, d) * (a, b, c, d) =
(aa + bc + cd + db,
ad + bb + cw + dc,

ab + bd + cc + da,
ac + ba + cb + dd)

and other modifications to the iteration formula call for both subtraction and cubing:

Subtraction: (a, b, c, d) - (w, x, y, z) = (a-w, b-x, c-y, d-z)

Cubing: product of the square and the original address. However, since multiplication is not commutative reversing the order of the factors may give a different result.

After any arithmetic operation the result must be normalized. This algebra I've called algebra #1.

A second algebra (algebra #2) is commutative and differs from Algebra 1 in its multiplication rule (which also impacts both squaring and cubing). Its form is:

Multiplication 2: (a, b, c, d) * (w, x, y, z) =
(aw + bz + cy + dx,
ax + bw + cz + dy,
ay + bx + cw + dz,
az + by + cx + dw)

This form mimics the standard complex plane 'rotations' and yields Mandelbrot like structures. Most of the work presented here uses this multiplication (algebra 2).

In order to ease comprehension of the images, and to simplify programming, data points are selected from a Cartesian 3-space, and their addresses are converted to the positive complex 4-axes system for whatever orientation of the two systems is selected. This four coordinate address can then be iterated with a similar formula as for the complex plane:

$$z_{(n+1)} = z_n^2 - c, \quad z = (a, b, c, d) \ .$$

Or with variations of:

$$z_{(n+1)} = z_n^3 + z_n^2 - z_{(n)}, \quad z = (a, b, c, d)$$

Generally only a few iterations are enough to determine if the point is *not* part of a Duncan set (as my friend and colleague insists on calling these sets). If it is part of the set the original point in Cartesian 3-space is colored black, if not part of the set, the point

can be colored depending on how many iterations it took to exceed the limits of the program or some other arbitrary limiting number.

Fair Warning: Most statements of fact in these papers should be taken as hypotheses based on empirical data. While the computer programs have been meticulously tested for flaws, and conclusions have been reached after many runs of the major programs, no proofs are being offered. The information in this paper is given in the hope that others will continue to explore and discover (and perhaps prove) both the mathematics and the applications that this current effort only hints at.

Forewarned, we can examine in some detail my favorite structure, the iterative cubic, 3-space, complex positive 4-axes, self similar fractal, nicknamed for brevity ‘the flagship’. But first a description of the computer images must be given and understood, as presented in the paper, ‘Computer Generated Images’, found on this same web site.