Pergamon

Comput. & Graphics, Vol. 18, No. 6, pp. 873–884, 1994
Copyright © 1994 Elsevier Science Ltd
Printed in Great Britain
0097-8493/94 $6.00 + .00

0097-8493(94)00107-3

DISCONTINUOUS AND ALTERNATE *Q*-SYSTEM FRACTALS

MICHAEL LEVIN
Department of Biological and Biomedical Sciences, Harvard Medical School, Boston, MA 02115, USA,
e-mail: mlevin@husc8.harvard.edu

Abstract—This paper presents various patterns and computer art from three extensions to common fractal methods: Here, I use discontinuous functions (such as Boolean functions and the conditional function), plots of root-finding methods other than Halley's and Newton's methods (such as Aitken's and Muller's methods), and fractals produced using arithmetic in *q*-systems other than the complex number system. Still fractal images and parametrization "movies" are discussed; these allow several new properties of these algorithms (and their combinations) to be uncovered. Also, the interaction of randomness with fractals is explored.

1. INTRODUCTION

The advent of high-speed computers and graphical display devices has made possible the recent plethora of images produced by chaotic algorithms (see [1–3], and references therein). Two common methods of producing fractal images involve Julia sets[4, 5] and root-finding method plots (*e.g.*, [6]). These fractals are usually produced by iterating a function employing a small set of mathematical operations (arithmetic, and sometimes trigonometric functions) which are defined within the complex number system. In this paper, I demonstrate two extensions of these methods, explore the interaction of Julia sets with randomness (noise), and show fractals associated with several types of root-finding methods.

2. THE BASIC FRACTALS

2.1. *Algorithms*

Julia sets are produced by viewing points in the R^2 plane as complex numbers and plotting the behavior of a function that is recurrently applied to each number in turn. These sets have applications in the study of phase transformations[7, 8], dynamical systems[9, 10], and developmental biology[11]. The basic Julia set algorithm used in this paper will be that defined by [12]. This is an escape-time fractal (where the escape value is given by the constant ε and is typically on the order of 100.0), and the pseudo-code (for a display using 8 colors and a black background) is as follows:

for each point (x, y) in some range minx<x<max*x*, miny<y<maxy, with constant c, and a complex function f( ),
the color of pixel [x, y] is determined as follows:

```
z = x+yi
for g from 0 to 15 in increments of 1:
    z = f(z,c);
    if $|z|>\varepsilon$, break loop.
if $|\mathrm{real}(z)|<\varepsilon$ or $|\mathrm{imaginary}(z)|<\varepsilon$, color = g mod 8,
    otherwise color = black
```

Root-method plots (*e.g.*, [6, 12, 13]) are produced by considering how fast a particular iterative root-finding method (such as Newton's method) converges on a root for a given function, when it is started with an initial guess x + y*i*. The rate of convergence determines the color of each point (x, y) and results in a fractal when a set of points in the R^2 plane are taken, in turn, as the initial guesses. The basic algorithm is as follows:

for each point (x,y) in some range minx<x<maxx, miny<y<maxy, and a complex function f( ),
the color of pixel [x,y] is determined as follows:

```
z = x+yi
for g from 0 to 50 in increments of 1:
    z = Ø(z);
    if $|(\mathrm{real}(z_g)^2+\mathrm{imaginary}(z_g)^2) - (\mathrm{real}(z_{g-1})^2+\mathrm{imaginary}(z_{g-1})^2)| < \varepsilon$ then break;
if $|(\mathrm{real}(z_g)^2+\mathrm{imaginary}(z_g)^2) - (\mathrm{real}(z_{g-1})^2+\mathrm{imaginary}(z_{g-1})^2)| < \varepsilon$ and g is even,
    then color = g mod 8,
    otherwise color = black
```

In this algorithm, convergence is considered to have occurred when the difference between the current "guess" and the previous one is smaller than the constant ε (usually on the order of 10^{-5}). The function being iterated, Ø( ), depends on which root finding method is being used. In the two for which fractal plots have been described in the literature, Newton's method, and Halley's method, the function Ø for an equation F( ) in complex numbers is as follows:

Newton's Method

$$z_{n+1} = z_n - \lambda \cdot \frac{F(z_n)}{F'(z_n)}$$

Halley's Method

$$z_{n+1} = z_n - \lambda \cdot \frac{F(z_n)}{F'(z_n) - \left(\dfrac{F''(z_n)\cdot F(z_n)}{2\cdot F'(z_n)}\right)}$$

Other methods, such as Stephenson's, Aitken's, and Muller's methods, are given in [14] and can be used with the above algorithm.

2.2. *Implementation*

These algorithms are easily coded in C, and run on any workstation with a graphical display. For this project, a library of complex number functions was produced, which defines all common functions (arithmetic, trigonometric, hyperbolic trigonometric, inverse trigonometric, logarithmic, exponential, *etc.*) for the complex data type. Because of the large numbers of pixels involved, and the many floating-point operations that have to be performed for each point (especially for Muller's method), a high-resolution plot is very time-consuming. However, these algorithms are ideally suited for SIMD (single instruction, multiple data) parallel architectures, since each point's color can be computed independently of the others. Thus, on the Connection Machine 2 (a SIMD parallel computer made by Thinking Machines Corp.), a plane of points can be computed in almost the same time it takes to compute a single point on a conventional machine. For the purposes of the studies described in this paper, the algorithms were implemented in C* (a parallel version of C), and run on the CM-2, with all points being computed in parallel. The software also included a dynamical parser package: functions entered by the user were parsed, and a dispatch table constructed. This allowed entry of functions at run-time, without recompiling the code.

The fact that whole planes of points could be computed in seconds of real-time also made it possible to produce animated parametrizations. Most commonly available fractal videos consist of a static fractal shape and a cycling color-map (though there are exceptions, such as those produced by Homer Smith, Robert Devaney, and Heinz-Otto Peitgen). More interesting is the ability to take a function and to produce an animation on videotape where each frame of the movie consists of a fractal of the function where one of the

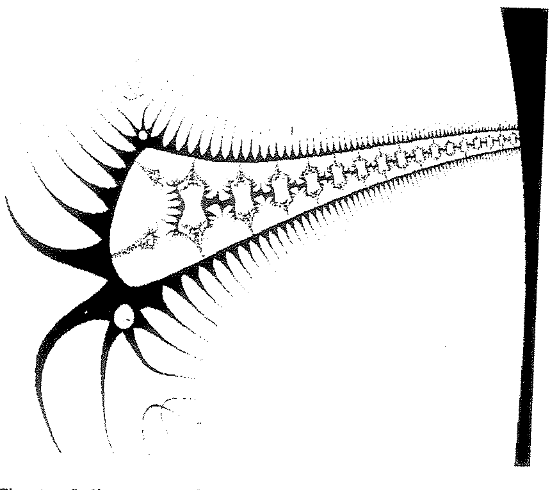

Fig. 1a. Julia set, x: $1.3 \rightarrow 4.7$, y: $1.0 \rightarrow 4.0$, $z = \sin(z) + e^z + 5 - 0.2i$; limit = 100.0.

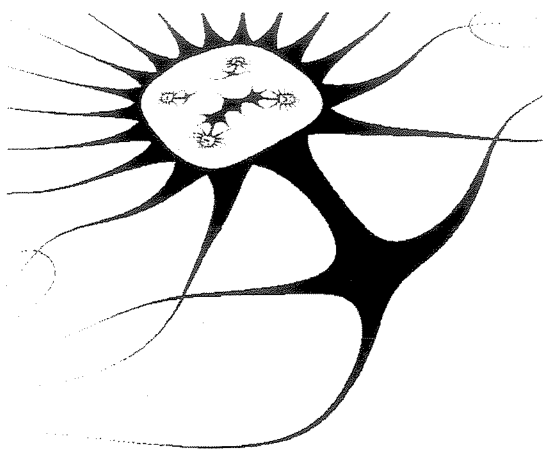

Fig. 1b. Julia set, x: $-1.2 \rightarrow -0.2$, y: $0.1 \rightarrow 1.4$, $z = \text{conj}(z)^{4.3} + |z|^z + v$.

parameters is slowly changed. For example, one can test to see how varying the parameter a from 2.0 to 10.0 alters the Julia set or root-method plot of $f(z) = z^a + \sin(z)$ by producing a movie where each frame uses a slightly different value of a. It is also possible to smoothly convert between two fractals f( ) and g( ) by making each frame the fractal plot of $a \cdot f(z) + (1 - a) \cdot g(z)$ as a goes from 1 to 0. Finally, it is possible to make movies of slowly varying relaxation parameter λ (in root-method plots), and of slowly-changing q-systems (see below). The movies produced from ordinary fractal formulas are quite striking; it is impossible to do them justice on paper, but several interesting results can be observed by such parametrizations.

2.3. *Sample images and observations on the Julia sets*

In general, the algorithms discussed in this paper represent mappings between elements in the set of possible functions ("set #1"), and the set of possible morphologies ("set #2"). In order to study the properties of the members of the first set, it is often useful to pick natural and simple formulas (*e.g.*, [5]). In this paper, I am instead concerned with exploring some of the members of the second set, as well as the properties of the algorithms themselves. Thus, no attempt is made to choose functions that are natural, simple, or have an immediate application to any specific scientific problem.

Some sample images using the Julia set algorithm are seen in Figs. 1–8. [12, 15, 11] show a wide variety of morphologies that can be produced by this algorithm. In general, colors do not merge—there are many instances of borders demarcating separate structures. Many objects possess a definite inside and outside. There are almost no straight lines; curves predominate. Repeated magnification reveals ever more intricate detail, but almost every function possesses a top-most view—a scale beyond which nothing happens. There are also recurrent spatial motifs. A large percentage of the formulae result in a circular form, with big spikes at its lowest magnification, with different things being seen in the center as one zooms in. Formulae occur in

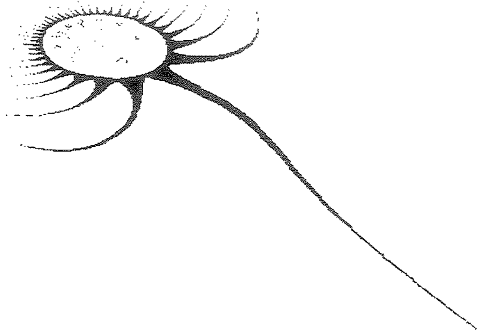
Fig. 2. Julia set, x: $-1.55 \rightarrow 0.55$, y: $-0.77 \rightarrow 1.71$, $z = 0.1 \cdot z^{9.5} + 0.9 \cdot \sin(z) + 2 - 3i$; limit = 60.0.

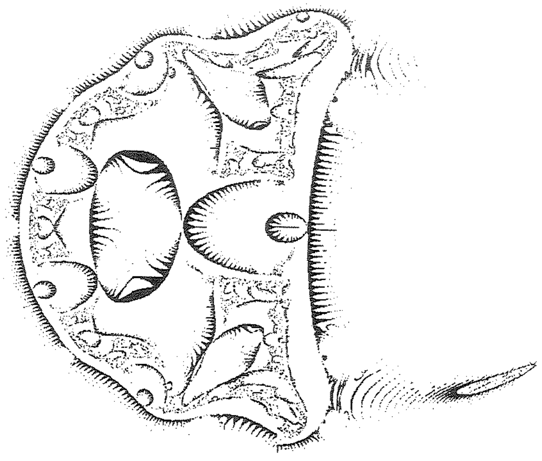
Fig. 5. Julia set, x: $-1.2 \rightarrow 1.0$, y: $-1.2 \rightarrow 1.2$, $z = e^{z/\cos(z)} + \text{reverse}(z)^{9.8}$; limit = 100.0.

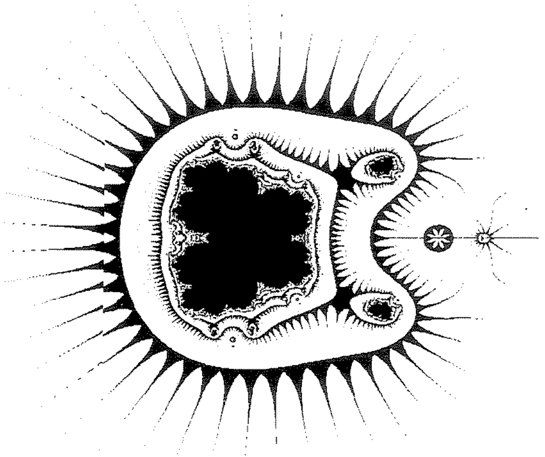
Fig. 3. Julia set, x: $-2.5 \rightarrow 2.5$, y: $-2.6 \rightarrow 2.6$, $z = z^{3.5} + 1.0/(\log(\sin(z))$; limit = 100.0.

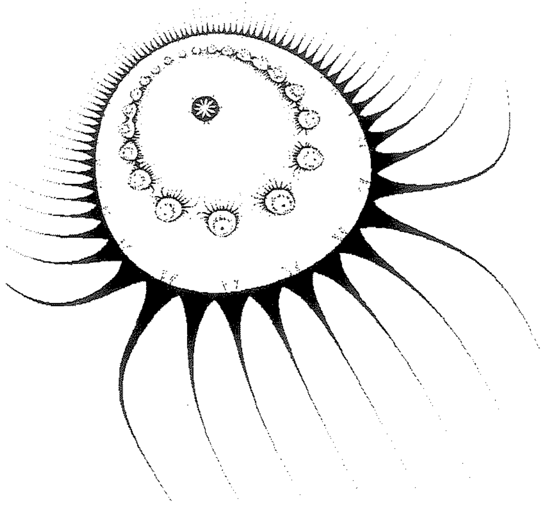
Fig. 6. Julia set, x: $-0.472 \rightarrow -0.205$, y: $0.7128 \rightarrow 0.996$, $z = (z^{22} + \sin(z))/z^{3.5}$; limit = 100.0.

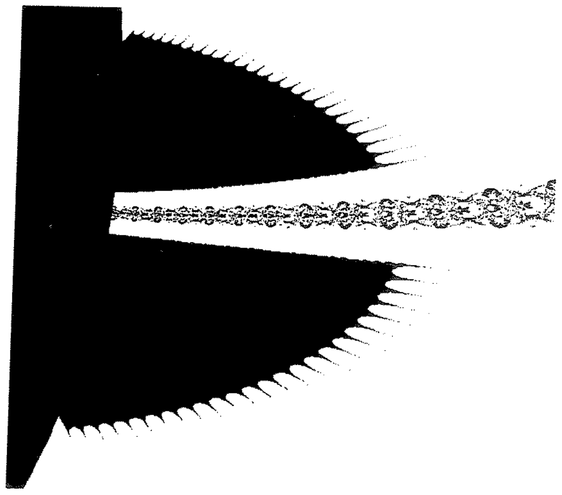
Fig. 4. Julia set, x: $-2.62 \rightarrow -2.55$, y: $3.572 \rightarrow 3.694$, $z = z^{1/\sin(z)} + e^{z}$; limit = 100.0.

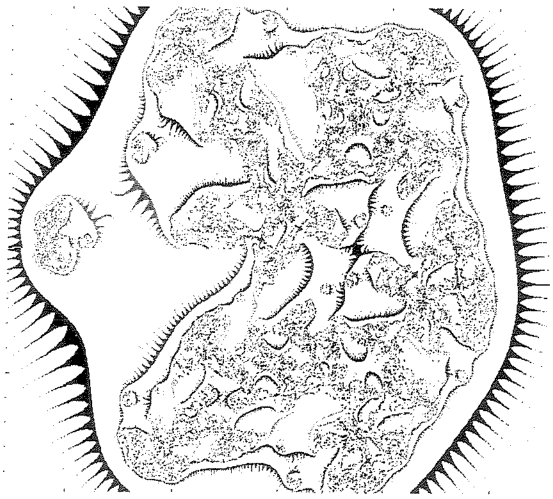
Fig. 7. Julia set, x: $-1.3 \rightarrow 1.3$, y: $-1.0 \rightarrow 1.0$, $z^{z} + z^{6} + -0.7 + 0.4i$, limit = 100.0.

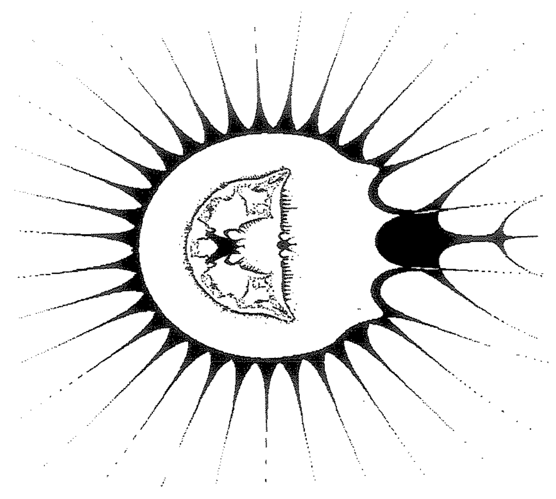

Fig. 8. Julia set, x: $-3.3 \rightarrow 3.6$, y: $-3.66 \rightarrow 3.6$, $z = (e^z)^{3.2} + \text{conj}(z)^{8.8}$, limit = 100.0.

families, with similar overall morphologies. For example, formulae in purely z^n where n is a small ($1 < n < 10$) positive real number result in circular forms with spikes, while formulas in $\sin(z)$ result in horizontal tubular forms, with repeated segments and wavy outer boarders. Note the contrast between the sharp morphologies of Figs. 1–6, and the confused internal arrangements (and failure of layers to join) of Fig. 7. There is also a clear (though difficult to formalize) distinction in the structuredness of various formulas (for example, compare Figs. 1–6 and 8 to Fig. 9).

These fractals possess interesting symmetry characteristics (Lakhtakia[7]). Some images contain repeated segments (Fig. 1a). Many parts of some images are self-similar (Fig. 6). They are not exactly repeated units, nor are they as repetitive as most Mandelbrot set images. There is usually some top-level shape, within which there are self-similar structures. Some images are symmetrical along one axis (as in Figs. 3, 8, and 9). Many are radially symmetrical, or pseudo-symmetrical (Figs. 1a, 1b, 2, and 6). Others, such as Fig. 5, seem to be bilaterally symmetrical along one axis, but possess one or more discrete features which spoil the symmetry. Still others (such as Fig. 7) are not symmetrical at all. It was also seen that adding a constant to formulas of the type $z = z^n$ (so that it becomes $z = z^n + v$) tends to disrupt symmetry.

Parametrization studies show that these algorithms display several interesting properties. Movies that are made by having each frame represent the Julia set (or root-method plot) of a complex function f( ) with a slightly different value for some parameter, show smooth morphological changes as a function of time. Since the parametrizations are linear (*i.e.*, a parameter changes by a constant amount between frames), different rates of changes in morphology observed are a natural property of the parametrization.

It is seen that, in general, most movies consist of periods of slow, smooth, continuous change interspersed with periods of very rapid major changes. Throughout, it is seen that certain structures (usually, the outermost layer) stay relatively constant in shape. Most movies show a shape that is generally static in size, while some have periods of growth. Some movies eventually settle down to a state where they change very little, though even in this state, very minor modifications can be seen from time to time. The movies show a clear avoidance of disjoinment of layers (though they often expand and contract to smoothly join with other layers).

These movies are in general quite striking. For example, one movie (x: -2.0 to 2.0, y: -2.0 to 2.0, limit = 100.0, $z = a\cdot(z^6 + z^z) + (1.0 - a)\cdot z^{8.5} + 0.5$, 400 frames determined by the variable *a* going from 0.0 to 1.0) consists of a number of layer subdivisions and fusions, and rotations of internal components. About 75% of the way through, a small part of the interior (on both sides of the axis of symmetry) acquires its own independent outer layer, moves outward, breaks through the outermost layer of the main figure (which then slowly rejoins to seal perfectly), moves away, and eventually disintegrates. Other movies display deformations of structures through bending, duplications, elongation, *etc.*, alternating periods of motion and stasis, contraction and expansion, *etc.* Even with a constant-size change in the parameter, each movie has its own characteristic rate of change.

The fact that it is possible to make these movies is interesting in itself. The functions themselves are highly nonlinear (*i.e.*, they are chaotic— $|f(x) - f(x + \Delta x)|$ is large even for small Δx), and it was a surprise to the author to discover that a relatively large step size is enough to produce a smooth transition between frames, resulting in a continuous movie. It may have been the case that given the nonlinear nature of these functions, small changes in the function produced images that were very different, thus making it impossible to produce a smoothly changing movie.

In making movies which slowly convert between two different formulas, different step sizes are needed to produce continuous movies between different shapes. That is, there are shapes which easily morph into one another, and there are shapes which do not (for ex-

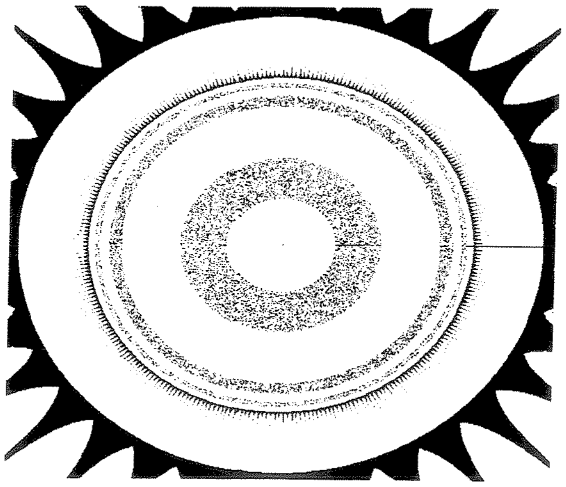

Fig. 9. Julia set, x: $-1.5 \rightarrow 1.5$, y: $-1.5 \rightarrow 1.5$, $z = z^{8.5}$, limit = 20.0.

ample, the shape produced by $z = \sin(z)$ is very difficult to convert into $z^{9.5}$). Thus, these morphologies form "natural kinds," the closeness of which is defined by the grain size needed to smoothly convert from one image to another. This is a higher-order property of the morphology, and cannot be determined directly by inspection from the formula itself (no regularity has been found in a large series of experiments). Also, even though the parameter is being changed in a linear manner (for example, a goes from 0.0 to 1.0 in equal intervals), such movies often display nonlinear transformations between images (*i.e.,* one image may be displayed unchanged for 95% of the movie, and then suddenly morph into the other one at the very end).

2.4. *Randomness and Julia sets*

So far, the algorithm has assumed that z coordinates (positions) are given exactly. Some interesting results can be obtained when this is altered by letting some components of the function have access to exact positions, and others to have access to randomly-altered positions. The algorithm then becomes: {pixel z's color $= \varnothing(z, z + \Delta z)$} where Δz is random. Fig. 10a shows the morphology of $z = z^{3.5}$. A plausible initial hypothesis as to what would happen if one added a small random offset to each point value is that the overall shape would be the same, but would be more fuzzy and diffuse, with less precise detail. Interestingly, this is not what happens. Fig. 10b shows the function $z = (z + \Delta z)^{3.5}$, where each $\Delta z = z \pm$ a random number of range $\log(z)$. Observe the fact that part of the image is completely untouched, while another part is duplicated and spatially shifted. One of the copies is more diffuse than the other.

Fig. 11a shows the morphology produced by $z = z^4 + \sin(z) - 26i$. Note the sharp and well-defined edges, but lack of horizontal or vertical straight lines. Fig. 11b shows the same thing done with each $\Delta z =$ a random number of range $\pm z$, raised to the 0.01th power). Note the fact that the interior detail has been obliterated by a clear rectangle; note also the "picture-frame" rectangle of fuzzy pattern around it, and the perfect (unchanged) detail outside that. It is interesting that adding a random component to these functions can produce a straight edge, where there was none before.

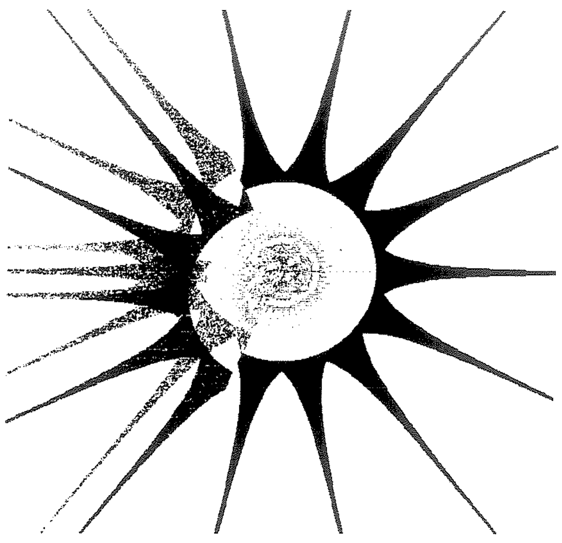
Fig. 10b. Julia set, x: $-5.0 \rightarrow 5.0$, y: $-5.0 \rightarrow 5.0$, $z = (z + \text{rand}(\log(z)))^{3.5}$, limit = 100.0.

Thus, errors in reading the positional information field produce much more complicated effects than just fuzziness in the resulting image. They can result in multiple copies of certain parts of the morphology, as well as complicated spatial patterns of regions of imprecise and precise structures. Some regions of the field are very resistant to perturbations; some are not. In all cases, there is a tolerance to such influences, since the results remain the same over a wide range of magnitudes for the influences.

2.5. *Basic root-method plots*

Figs. 12 and 13 show typical Halley's method plots. Fig. 14 shows a typical Newton's method plot. The

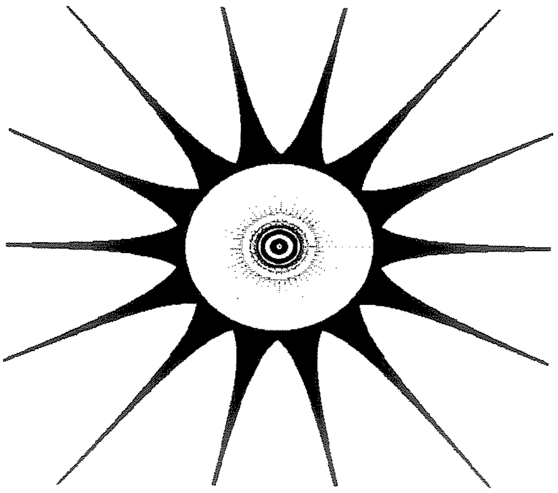
Fig. 10a. Julia set, x: $-5.0 \rightarrow 5.0$, y: $-5.0 \rightarrow 5.0$, $z = z^{3.5}$, limit = 100.0.

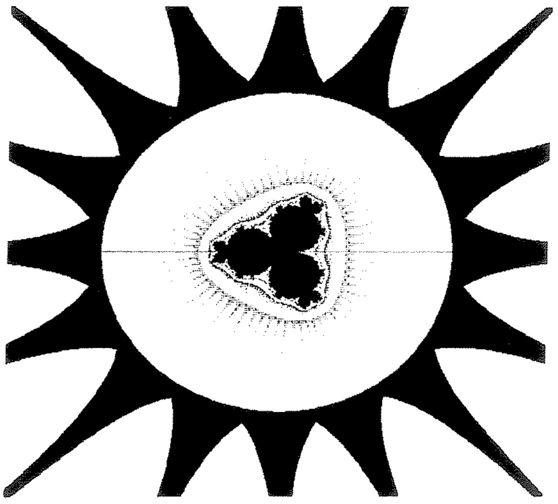
Fig. 11a. Julia set, x: $-2.5 \rightarrow 2.5$, y: $-2.5 \rightarrow 2.5$, $z = z^4 + \sin(z) - 26i$, limit = 100.0.

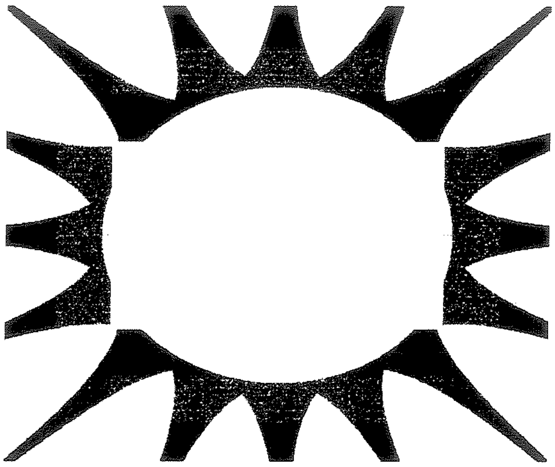
Fig. 11b. Julia set, x: $-2.5 \rightarrow 2.5$, y: $-2.5 \rightarrow 2.5$, $z = z^4 + \sin(z) - 26i$, limit = 100.0.

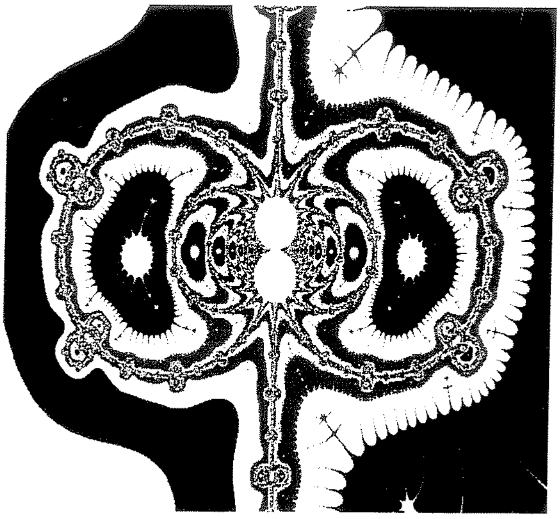
Fig. 13. Halley plot, x: $1.137744 \rightarrow 2.021231$, y: $0.554824 \rightarrow 1.204$, $z = \sin(z)$.

basic characteristics of these have been described[6, 12, 13]. Several interesting things can be seen when movies are made using the same algorithm as above. Firstly, it is seen that changing the lambda relaxation parameter makes movies that look like progressive zoom-ins into the center of the plot. In general, the movies are very striking, with many complex deformations. It is interesting to note that for movies made using Halley plots (but not for any of the other methods described in this paper), the following form has a very special characteristic: when a movie is made using the equation $z = \cos(z^2 + c) - c$, and c is varied from 0.0 to 40.0 in the course of the movie (made over the section of the plane x: $1.13 \rightarrow 2.02$, y: $0.55 \rightarrow 1.204$), the movie contains no deformations. Instead, it looks as if one was moving at varying speeds with a fixed-size viewport over an unchanging (static) image. In contrast to this linear translation effect, all other movies show clear deformations in place (structures expand and contract, there is bending and shearing, *etc.*). By experiment, it is seen that this is a property of the forms $z = \cos(z^n + c) - c$ and $z = \sin(z^n + c) - c$ for small n, but not of $z = \cosh(z^n + c) - c$ or $z = \sinh(z^n + c) - c$, nor of any other formula found so far.

Extension 1: Other root-finding methods

Other root-finding methods can be found in Grove[14]. These can be extended to the complex numbers. Of these, Newton's method is the least computationally demanding; Halley's method is intermediately so. Muller's method and Stephenson's methods are the most demanding. Figs. 15–17 show some other sample root method plots. Each method has certain individual characteristics. For example, Newton and Halley plots are usually quite similar. Stephenson plots contain much empty space and separated islands of color, and occasionally the "bull's-eye" patterns found in Newton and Halley plots. Muller plots contain many small whirl-like regions. Aitken plots often contain very fuzzy areas (these are regions where points of different color intermix, without any large solidly-colored re-

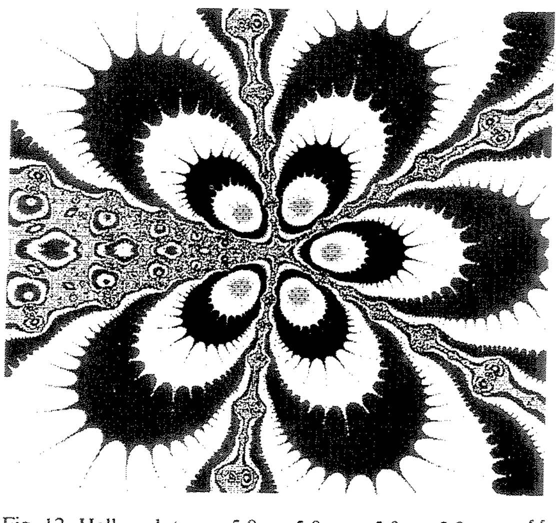
Fig. 12. Halley plot, x: $-5.0 \rightarrow 5.0$, y: $-5.0 \rightarrow 5.0$, $z = z^{5.5} - 1$.

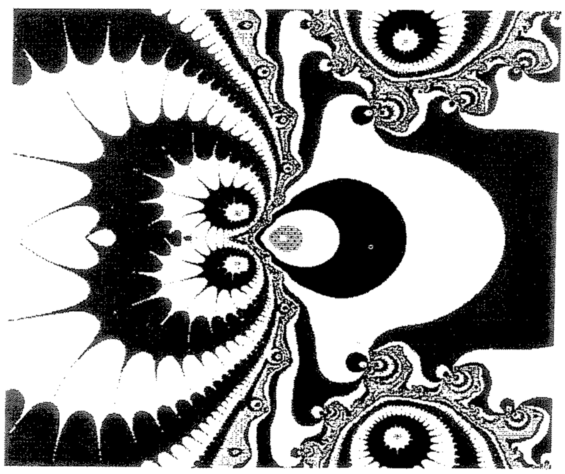
Fig. 14. Newton plot, x: $-5.0 \rightarrow 5.0$, y: $-5.0 \rightarrow 5.0$, $z = z^{2.3} + \sin(z)$.

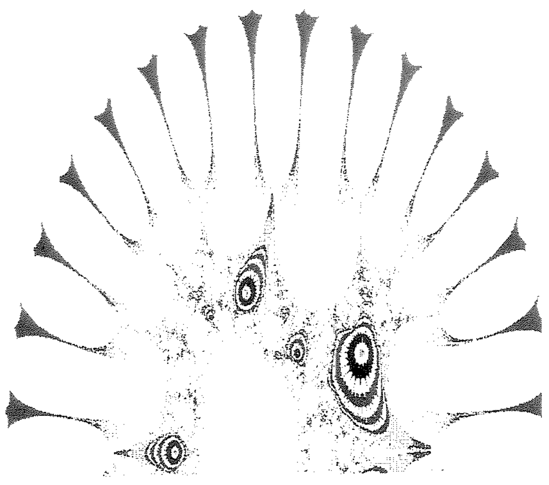
Fig. 15. Stephenson plot, x: $-2.0 \rightarrow 2.0$, y: $0.0 \rightarrow 2.0$, $z = z^5 + 0.5$.

gions). Movies made with these methods are visually very impressive (especially Muller plots).

Extension 2: Discontinuous functions

Most fractals are drawn using the common functions (rows 8–9 of Table 1). However, other functions (such as trigonometric and hyperbolic trigonometric) can be defined over the complex number data type (rows 6–7 of Table 1). In addition, it is possible to define discontinuous functions such as the Boolean (logic) functions, and some others (rows 1–5 of Table 1) over the complex numbers. In these cases, True and False are defined as being precisely equal to 1.0 and 0.0, respectively.

These can then be used to produce fractal images (as Julia sets, or root-method plots) of a much wider range of complex number functions. One last and very powerful addition to this function set involves the "cond" function. This is defined as the standard LISP (cond a b c) function form, which returns "b" or "c" depending on whether the parameter "a" is True or False. This function allows one to section the plane. For example (in LISP notation), "z = (cond (r < z$_0$ 3)(sin z) (pow z 3.0))", applies sin(z) to the x < 3.0

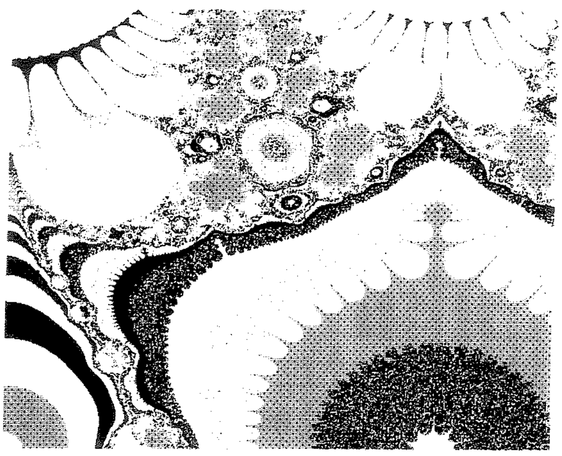
Fig. 16. Aitken plot, x: $0.0 \rightarrow 4.0$, y: $0.0 \rightarrow 4.0$, $z = \sin(z)$.

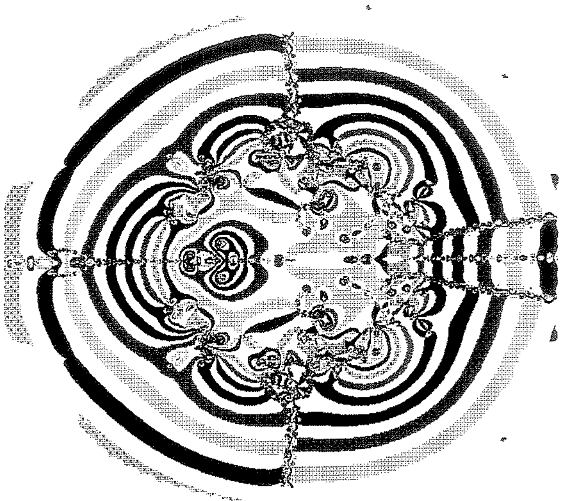
Fig. 17. Muller plot, x: $-2.5 \rightarrow 2.5$, y: $-2.5 \rightarrow 2.5$, $z = z^4 + \sin(z)$.

part of the complex plane, and z^3 to the rest of the plane. Note that the "cond" function violates assumptions of analyticity used in the theory of Julia sets. This makes analysis of functions using it more difficult; however, it allows one to simulate systems which include a choice component at each iteration (*e.g.*, [11]).

When using functions which perform a choice (such as "cond," "min," "max," *etc.*), it is useful to have them keep track of and display the ratio in which the two choices were given, during a particular image. This is because sometimes such a function always returns the same choice, and can thus be simplified, reducing compute time (for example, the function "(min z 0)" will always return "0" if z stays positive). Figs. 18–20 show some examples of Julia sets and root-method plots produced with discontinuous functions.

Extension 3: i-systems other than $i^2 = -1$

The last extension to the common fractal methods involves alternate q-systems[20]. Number systems involving numbers of the form a + bi can be defined by addition and multiplication rules which operate on these numbers and satisfy certain constraints (for example, that addition and multiplication must be commutative and associative, and that multiplication must be distributive over addition). These rules, however, do not constrain the value of i^2 to any particular number. In the ordinary complex number system, i^2 is made to equal -1. However, it is possible to define consistent number systems with i^2 being equal to arbitrary p + qi. This defines the character of the system and determines the properties of operations within it. It turns out that there is not an infinite number of such systems, but that each system can be reduced to one of the following cases:

- numbers a + bi with $i^2 = -1$ (the complex numbers),
- numbers a + bi with $i^2 = 1$ (the "double" numbers),
- numbers a + bi with $i^2 = 0$ (the "dual" numbers).

Nevertheless, it is possible to define all common complex number functions (such as arithmetic, trig-

Table 1. Functions defined over the complex data-type.

1	r >, r <, r =	Decide whether the real part of the first argument is greater/less/equal to that of the second, and return $1 + 0i$ if so, $0 + 0i$ otherwise.
2	i >, i <, i =	Decide whether the imaginary part of the first argument is greater/less/equal to that of the second, and return $1 + 0i$ if so, $0 + 0i$ otherwise.
3	and, or, not, xor	Standard binary functions which return $1 + 0i$ for True, $0 + 0i$ for False
4	reverse	$a + bi$ maps to $b + ai$
5	min, max	Returns the minimum and maximum of 2 complex numbers
6	conj	Complex conjugate
7	sin, cos, tan, sinh, cosh, tanh, $\sin^{-1}$, $\cos^{-1}$, $\tan^{-1}$, $\sinh^{-1}$, $\cosh^{-1}$, $\tanh^{-1}$	Ordinary, hyperbolic, and inverse trigonometric operations
8	log, log10, pow	Logarithms and complex powers
9	$+, -, \div, \cdot$	Standard arithmetic operators
10	cond	If argument #1 is True, return argument #2, else, return argument #3

onometric functions, powers, *etc.*) over any arbitrary q-number system. The definitions are performed as follows:

For a number system with arbitrary A and B such that $q^2 = A + q(2B)$ (the factor 2 is there just to simplify the results), the numbers have two individual terms to which expansion formulas can be applied (for example, $\sin(a + bi)$ can be simplified to functions of a and bi using the regular sine expansion formula). Defining addition and subtraction is easy since one only has to add and subtract like terms, and neither A nor B is involved since q is never squared. Thus,

$$(\mathbf{a} + q\mathbf{b}) + (\mathbf{c} + q\mathbf{d}) = (\mathbf{a} + \mathbf{c}) + q(\mathbf{b} + \mathbf{d}),$$

$$(\mathbf{a} + q\mathbf{b}) - (\mathbf{c} + q\mathbf{d}) = (\mathbf{a} - \mathbf{c}) + q(\mathbf{b} - \mathbf{d}).$$

Multiplication isn't much harder since the distributive law can be used to give:

$$(\mathbf{a} + q\mathbf{b}) \cdot (\mathbf{c} + q\mathbf{d}) = a \cdot c + b \cdot c \cdot q + d \cdot a \cdot q + b \cdot d \cdot q^2,$$

collect like terms:

$$a \cdot c + (b \cdot c + d \cdot a) \cdot q + b \cdot d \cdot q^2,$$

use the definition of q^2:

$$a \cdot c + (b \cdot c + d \cdot a) \cdot q + b \cdot d \cdot (A + 2Bq),$$

collect like terms:

$$(\mathbf{a} \cdot \mathbf{c} + \mathbf{b} \cdot \mathbf{d} \cdot \mathbf{A}) + (\mathbf{b} \cdot \mathbf{c} + \mathbf{d} \cdot \mathbf{a} + \mathbf{b} \cdot \mathbf{d} \cdot \mathbf{2B}) \cdot q.$$

Division becomes:

$$(\mathbf{x} + q\mathbf{y}) \div (\mathbf{u} + q\mathbf{v}) = (\mathbf{ux} + \mathbf{2Bvx} - \mathbf{Avy})/(\mathbf{u}^2 + \mathbf{2Buv} - \mathbf{Av}^2) + (\mathbf{uy} - \mathbf{vx})/(\mathbf{u}^2 + \mathbf{2Buv} - \mathbf{Av}^2)q$$

Sines, cosines, and hyperbolic sines and cosines are slightly more tricky. All q-systems can be classified on the basis of the A and B values used in the definition of q^2:

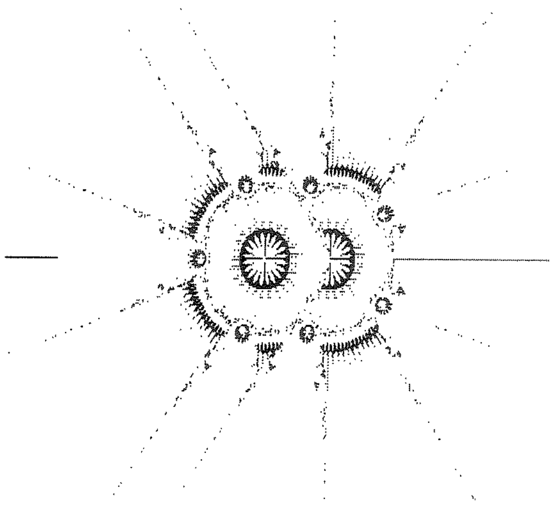
Fig. 18. Julia set, x: $-5.0 \rightarrow 5.0$, y: $-5.0 \rightarrow 5.0$, $z = (z + (r < zz^2))^{-0.5}$.

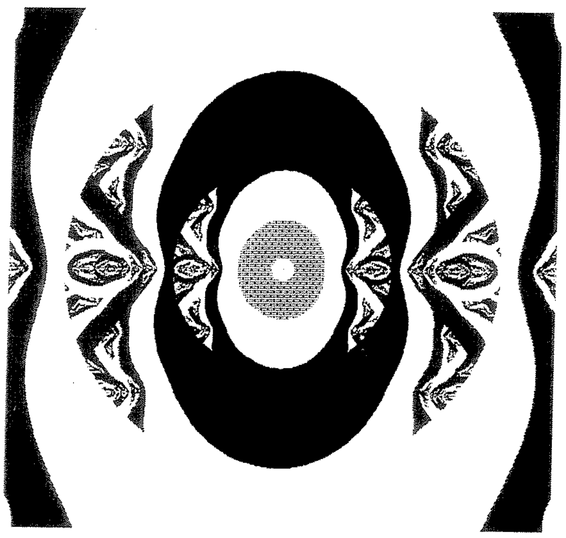
Fig. 19. Newton plot, x: $-5.0 \rightarrow 5.0$, y: $-5.0 \rightarrow 5.0$, $z = (\mathrm{cond}(r < |z|2)\sin(z)z^2)$.

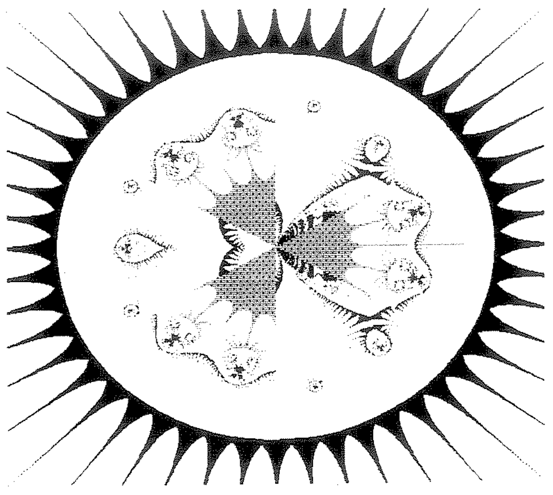
Fig. 20. Julia set, x: $-5.0 \rightarrow 5.0$, y: $-5.0 \rightarrow 5.0$, $z = z^{12} + z^{((r < \sin(z)\, z)/z)}$.

systems with $B^2 + A < 0$

are isomorphic with the i-system where $i^2 = -1$,

systems with $B^2 + A = 0$

are isomorphic with the l-system where $l^2 = 0$,

systems with $B^2 + A > 0$

are isomorphic with the h-system where $h^2 = 1$.

The i, l, and h-systems are the classical systems. To find the expression for f(a + b*i*) in an arbitrary *q*-system, one must convert that problem to a problem involving one of the classical systems. This is because all the common functions have easy definitions in the classical systems. For each *q*, there is a classical q̂ which behaves just like *q*. Thus, one must find an expression for *q* in terms of the classical q̂, with which it is replaced:

$$q = B + \hat{q} \cdot D \quad \text{where} \quad D = \{\mathrm{sqrt}(-(B^2 + a)) \text{ or } 1 \text{ or } \mathrm{sqrt}(B^2 + A)\} \quad \text{for the i, l,}$$

and h systems respectively.

The transformation is later reversed (express the q̂ in terms of q) by:

$$\hat{q} = -B/D + q \cdot 1/D$$

For example, to find sin(x + qy):

a. replace *q* with the classical q̂ getting: $\sin(x + (B + \hat{q} \cdot D) \cdot y)$

b. collect like terms inside the sine function: $\sin(x + B \cdot y + \hat{q} \cdot D \cdot y)$

c. use the sine addition formula, getting:

$$\sin(x + B \cdot y) \cdot \cos(\hat{q} \cdot D \cdot y) + \cos(x + B \cdot y) \cdot \sin(\hat{q} \cdot D \cdot y).$$

Now, using the power series for sines and cosines, we find sin( ) and cos( ) for imaginary values in classical systems:

$$\sin(\hat{q} \cdot D \cdot y) = \hat{q} \cdot \left\{ \begin{matrix} \sinh(D \cdot y) \\ D \cdot y \\ \sin(D \cdot y) \end{matrix} \right\},$$

$$+ \cos(\hat{q} \cdot D \cdot y) = \hat{q} \cdot \left\{ \begin{matrix} \sinh(D \cdot y) \\ D \cdot y \\ \sin(D \cdot y) \end{matrix} \right\}$$

for the i, l, and h systems respectively. Substituting those definitions for the terms involving imaginary sines and cosines in (c) above, we get:

$$\sin(x + qy) = \sin(x + B \cdot y) \cdot \left\{ \begin{matrix} \cosh(D \cdot y) \\ 1 \\ \cos(D \cdot y) \end{matrix} \right\} + \cos(x + B \cdot y) \cdot \hat{q} \cdot \left\{ \begin{matrix} \sinh(D \cdot y) \\ D \cdot y \\ \sin(D \cdot y) \end{matrix} \right\}.$$

Now, q̂ is replaced by the expression in terms of *q* given above (q̂ = −B/D + *q*·1/D):

$$\sin(x + qy) = \sin(x + B \cdot y) \left\{ \begin{matrix} \cosh(D \cdot y) \\ 1 \\ \cos(D \cdot y) \end{matrix} \right\} - \frac{B}{D} \cdot \cos(x + B \cdot y) \left\{ \begin{matrix} \sinh(D \cdot y) \\ D \cdot y \\ \sin(D \cdot y) \end{matrix} \right\} + q \cdot \frac{1}{D} \cdot \cos(x + B \cdot y) \left\{ \begin{matrix} \sinh(D \cdot y) \\ D \cdot y \\ \sin(D \cdot y) \end{matrix} \right\}.$$

Likewise:

$$\cos(x + qy) = \cos(x + B \cdot y) \left\{ \begin{matrix} \cosh(D \cdot y) \\ 1 \\ \cos(D \cdot y) \end{matrix} \right\} + \frac{B}{D} \cdot \sin(x + B \cdot y) \left\{ \begin{matrix} \sinh(D \cdot y) \\ D \cdot y \\ \sin(D \cdot y) \end{matrix} \right\} - q \cdot \frac{1}{D} \cdot \sin(x + B \cdot y) \left\{ \begin{matrix} \sinh(D \cdot y) \\ D \cdot y \\ \sin(D \cdot y) \end{matrix} \right\},$$

$$\sinh(x + qy) = \sinh(x + B \cdot y) \left\{ \begin{matrix} \cos(D \cdot y) \\ 1 \\ \cosh(D \cdot y) \end{matrix} \right\} - \frac{B}{D} \cdot \cosh(x + B \cdot y) \left\{ \begin{matrix} \sin(D \cdot y) \\ D \cdot y \\ \sinh(D \cdot y) \end{matrix} \right\} + q \cdot \frac{1}{D} \cdot \cosh(x + B \cdot y) \left\{ \begin{matrix} \sin(D \cdot y) \\ D \cdot y \\ \sinh(D \cdot y) \end{matrix} \right\},$$

$$\cosh(x + qy) = \cosh(x + B\cdot y)\begin{Bmatrix}\cos(D\cdot y)\\ 1\\ \cosh(D\cdot y)\end{Bmatrix}$$

$$-\frac{B}{D}\cdot\sinh(x + B\cdot y)\begin{Bmatrix}\sin(D\cdot y)\\ D\cdot y\\ \sinh(D\cdot y)\end{Bmatrix}$$

$$+ q\cdot\frac{1}{D}\cdot\sinh(x + B\cdot y)\begin{Bmatrix}\sin(D\cdot y)\\ D\cdot y\\ \sinh(D\cdot y)\end{Bmatrix}.$$

Thus, the procedure for computing $\sin(x + yq)$ is as follows (given that one knows the A and B which define the q-system):

1. compute $B^2 + A$ and determine which of the i, l, or h systems is being used. Use this to compute D.
2. Now, use the knowns, x, y, B, and D, and the appropriate system (picking out the right one of the vertically written possibilities) to determine the value of the function.

To compute exponentials (base e), the following results are used:

$$e^{\alpha i} = \cos(\alpha) + i\cdot\sin(\alpha),$$
$$e^{\alpha l} = 1 + l\cdot\alpha,$$
$$e^{\alpha h} = \cosh(\alpha) + h\cdot\sinh(\alpha),$$

substituting $\hat{q}$ for q: $e^{x+qy} = e^{x+By+\hat{q}\cdot D\cdot y}$,
using the law of exponents: $e^{x+By}\cdot e^{\hat{q}\cdot D\cdot y}$,
substituting $e^{\hat{q}\cdot D\cdot y}$ for the identities above:

$$e^{x+B\cdot y}\cdot\begin{Bmatrix}\cos(D\cdot y)\\ 1\\ \cosh(D\cdot y)\end{Bmatrix} + e^{x+B\cdot y}\cdot\hat{q}\cdot\begin{Bmatrix}\sin(D\cdot y)\\ D\cdot y\\ \sinh(D\cdot y)\end{Bmatrix},$$

substituting back for $\hat{q}$ in terms of q:

$$e^{x+qy}$$

$$= e^{x+B\cdot y}\cdot\left(\begin{Bmatrix}\cos(D\cdot y)\\ 1\\ \cosh(D\cdot y)\end{Bmatrix} - \frac{B}{D}\cdot\begin{Bmatrix}\sin(D\cdot y)\\ D\cdot y\\ \sinh(D\cdot y)\end{Bmatrix}\right)$$

$$+ q\cdot\frac{1}{D}\cdot e^{x+B\cdot y}\cdot\begin{Bmatrix}\sin(D\cdot y)\\ D\cdot y\\ \sinh(D\cdot y)\end{Bmatrix}.$$

Complex powers can likewise be extended. To compute $(x + yi)^{(v+wi)}$:

$$r = \mathrm{sqrt}(|x^2 + 2\cdot B\cdot x\cdot y - A\cdot y^2|)$$

note that:

for $B^2 + A = 0$, θ is defined only when

$$x + B\cdot y > 0$$

for $B^2 + A > 0$, θ is defined only when

$$x + B\cdot y > \mathrm{sqrt}(B^2 + A)\cdot|y|$$

for $B^2 + A < 0$, θ is *undefined* when

$$x + B\cdot y = 0 \quad\text{and}\quad y = 0.$$

for these three cases, the definition of θ is as follows:

$$\theta = \frac{1}{D}\cdot\begin{Bmatrix}\tan^{-1}\left(\dfrac{D\cdot y}{(x + B\cdot y)}\right)\\ \\ \dfrac{D\cdot y}{(x + B\cdot y)}\\ \\ \tanh^{-1}\left(\dfrac{D\cdot y}{(x + B\cdot y)}\right)\end{Bmatrix}$$

because of the peculiarities of $\tan^{-1}(\ )$ in picking which angle to return, several adjustments are necessary for the first case ($B^2 + A < 0$):

if $x + B\cdot y > 0$,

then θ is exactly as given above (1st tier of column),

if $x + B\cdot y < 0$ and $y \neq 0$,

θ is as given above, $+(y/|y|)\cdot\pi/D$,

if $x + B\cdot y < 0$ and $y = 0$, $\theta = \pi/D$,

if $x + B\cdot y = 0$ and $y \neq 0$,

$\theta = (y/|y|)\cdot\pi/(2\cdot D)$,

if $x + B\cdot y = 0$ and $y = 0$,

θ is undefined (as already mentioned above).

then,

$$(x + yi)^{(u+vi)} = e^{u(\ln(r)-B\cdot\Theta)+A\cdot v\cdot\Theta+q(v(\ln(r)+B\cdot\Theta)+u\cdot\theta)}$$

or, one can get r and θ, define the logarithm as below, and then use the fact that $z^w = e^{w\cdot\ln(z)}$.

Using the r and θ as defined in Part 6 above, logarithms are computed as:

$$\ln(x + qy) = \ln(r) - B\cdot\text{theta} + q\cdot\text{theta}$$

The complex number system is by far the most often used. However, the others also have applications in physics [16–19]. To my knowledge, all widely available fractal images to date are produced using the rules of the familiar $i^2 = -1$ complex number system. Having generalized the library of common and discontinuous functions to all q-systems, it is possible to produce fractals in other systems. These systems involve operations whose properties differ from those defined under $i^2 = -1$, and the fractals produced often look very different from their $i^2 = -1$ counterparts.

To study the properties of various fractals in other number systems, the complex number library was extended to allow operations under any system of the form $i^2 = p + qi$. Figs. 21–24 show several Julia set and root-method plots produced in various q-systems.

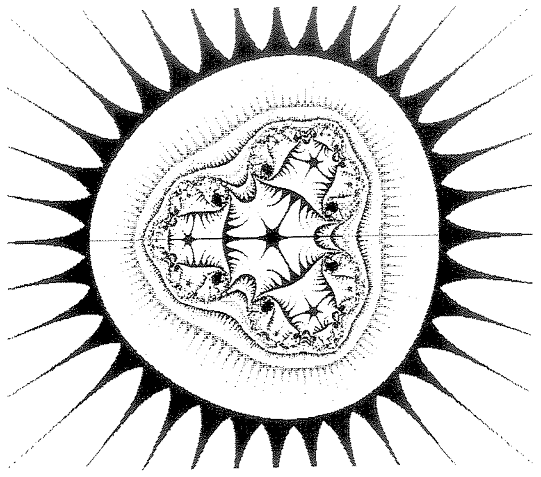
Fig. 21. Julia set, x: $-2.5 \rightarrow 2.5$, y: $-2.12 \rightarrow 2.12$, $z = z^3 + 0.5$, $i^2 = -1.0$.

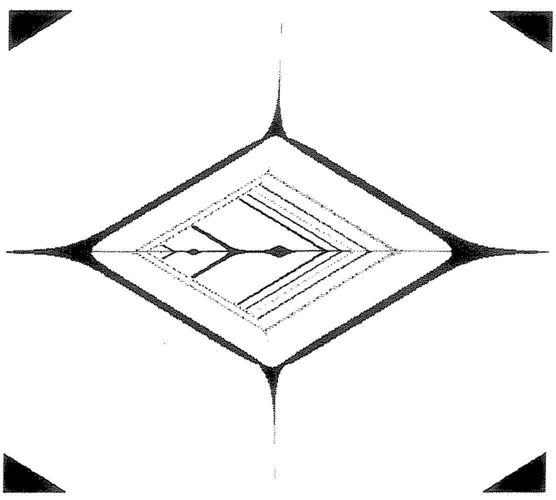
Fig. 22. Julia set, x: $-2.5 \rightarrow 2.5$, y: $-2.12 \rightarrow 2.12$, $z = z^3 + 0.5$, $i^2 = 3.0$.

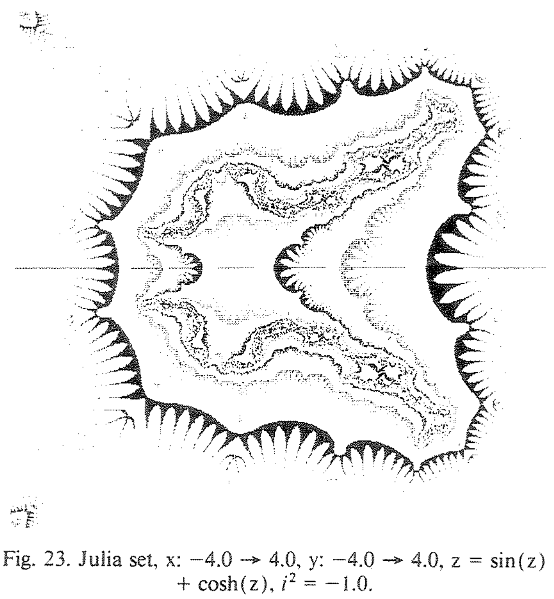
Fig. 23. Julia set, x: $-4.0 \rightarrow 4.0$, y: $-4.0 \rightarrow 4.0$, $z = \sin(z) + \cosh(z)$, $i^2 = -1.0$.

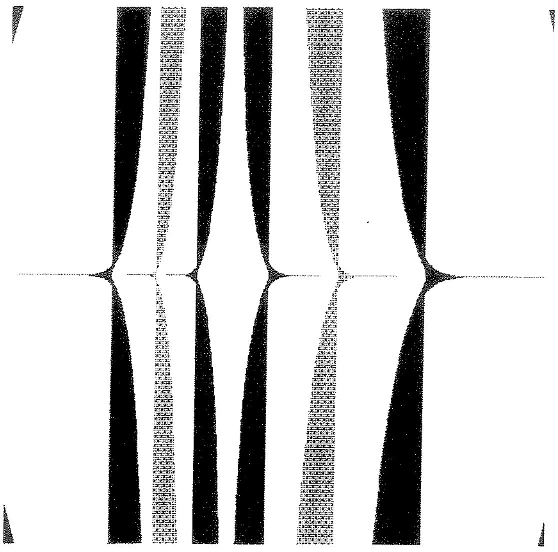
Fig. 24. Julia set, x: $-4.0 \rightarrow 4.0$, y: $-4.0 \rightarrow 4.0$, $z = \sin(z) + \cosh(z)$, $i^2 = 0.0$.

For example, Figs. 21 and 22 represent the same Julia set, drawn in the common $i^2 = -1$ system and in the $i^2 = 3$ system. Notice that the complex round shapes have become converted to simpler angular ones, and that the symmetry of the internal shape has changed from 3-fold to bilateral around the x-axis. In general, it is noticed that fractals under the $i^2 = 0$ system are the least visually complex, followed by the $i^2 > 0$ system. The ones produced by the common $i^2 = -1$ system seem to be the most morphologically interesting.

For any given fractal, changing the q-system from -1 results in changes of scale, shearing, and folding. It is also possible to study the properties of the systems by observing the changes in a single function when its Julia set is drawn in slowly-changing q-systems (*i.e.*, a movie is produced of the Julia set of the static function f( ), where each frame of the movie represents a slightly different i^2 value). These movies clearly demonstrate the shearing and stretching transformations of a fractal image as it traverses the various number systems.

Acknowledgements—I would like to thank Dr. Paul Fjelstad for his very generous help with developing the q-system expressions, and Dmitry Grenader for helpful discussions. I would also like to acknowledge the grant of CM-2 time by the Pittsburgh Supercomputer Center.

REFERENCES

1. B. Mandelbrot, *The Fractal Geometry of Nature*, Media Magic, New York (1982).
2. H.-O. Peitgen and D. Saupe (Eds.), *The Science of Fractal Images*, Springer-Verlag, New York (1988).
3. H.-O. Peitgen and P. H. Richter, *The Beauty of Fractals*, Springer-Verlag, Berlin (1986).
4. G. Julia, Memoire sur l'iteration des fonctions rationelles. *J. Math. Pure Appl.* **4**, 47–245 (1918).
5. U. G. Gujar and V. C. Bhavsar, Fractals from $z \leftarrow z^\alpha + c$ in the complex c-plane. *Comput. & Graphics* **15**, 441–449 (1991).
6. C. Pickover, Halley maps for a trigonometric and rational

function, *Comp. and Math with Appl.* **17**, 125–132 (1989).

7. A. Lakhtakia, V. V. Varadan, R. Messier, and V. K. Varadan, On the symmetries of the Julia sets for the process $z = > z^p + c$. *J. Phys. A*, **20**, 3533–3535 (1987).
8. K. G. Wilson, Renormalization group and critical phenomena. *Phys. Rev. B* **4**, 3174–3205 (1981).
9. A. Lakhtakia, Julia sets of switched processes. *Comput. and Graphics* **15**, 597–599 (1991).
10. M. F. Barnsley and S. G. Demko (Eds.), *Chaotic Dynamics and Fractals*, Academic Press, New York (1986).
11. M. Levin, A computer model of field-directed morphogenesis: Julia sets. *Computer Applications in the Biosciences*, **10**(2), 85–103 (1994).
12. C. Pickover, *Computers, Pattern, Chaos, and Beauty*, St. Martin's Press, New York (1990).
13. C. Pickover, A note on chaos and Halley's method. *Commun. Assoc. Comp. Mach.* **31**, 1326–1329 (1988b).
14. W. E. Grove, *Brief Numerical Methods*, Prentice-Hall, Englewood Cliffs, NJ (1966).
15. C. Pickover, Computer graphics generated from the iteration of algebraic transformations in the complex plane. *Comput. & Graphics* **9**, 147–154 (1985).
16. P. Fjelstad, *Inventing Different Kinds of Physics*. Unpublished manuscript (1989).
17. P. Fjelstad, Extending relativity via the perplex numbers. *Am. J. Phys.* **54**, 416–422 (1986).
18. W. Band, Comments on 'extending relativity via the perplex numbers. *American Journal of Physics*, **56**, 469 (1988).
19. V. Majernik, The perplex numbers are in fact the binary numbers. *Am. J. Phys.* **56**, 763 (1988).
20. I. L. Kantor and A. S. Solodnikov, *Hypercomplex Numbers*, Springer-Verlag, New York (1989).