9-14-2017

Generalized Fractals for Computer Generated Art: Preliminary Results

Charles F. Babbs
Purdue University, babbs@purdue.edu

GENERALIZED FRACTALS FOR COMPUTER GENERATED ART: PRELIMINARY RESULTS

Charles F. Babbs, MD, PhD*

*Weldon School of Biomedical Engineering, Purdue University, West Lafayette, Indiana, USA

Abstract. This paper explores new types of fractals created by iteration of the functions $x_{n+1} = f_1(x_n, y_n)$ and $y_{n+1} = f_2(x_n, y_n)$ in a general plane, rather than in the complex plane. Iteration of such functions generates orbits with novel fractal patterns. Especially interesting are N-th order polynomials, raised to a positive or negative integer power, p,

$$f_1(x_n, y_n) = \left[\sum_{i=0}^{N}\sum_{j=0}^{N} a_{ij} x_n^j y_n^i\right]^p \text{ and } f_2(x_n, y_n) = \left[\sum_{i=0}^{N}\sum_{j=0}^{N} b_{ij} x_n^j y_n^i\right]^p.$$

Such functions create novel fractal patterns, including budding, spiked, striped, dragon head, and bat-like forms. The present faculty working paper shows how to create a rich variety of complex and fascinating fractals using this generalized approach, which is accessible to students with high school level skills in mathematics and coding.

Keywords. Algorithmic art, Chaos, Complex plane, Dynamics, Escape set, Fractals, Graphics, Iteration, Julia set, Mandelbrot set, Prisoner set.

Background

The fractal patterns in algorithmic or computer generated art are based on the concept of iteration. This concept can be demonstrated easily in just one dimension. Let f(x) be a function of real variable, x, and consider the sequence of computations x_0, $x_1 = f(x_0)$, $x_2 = f(x_1)$, $x_3 = f(x_2)$, and so on, such that the input or argument of f(x) in step n + 1 equals the output or value of f(x) in step n. This process in which $x_{n+1} = f(x_n)$ is called iteration or recursion.

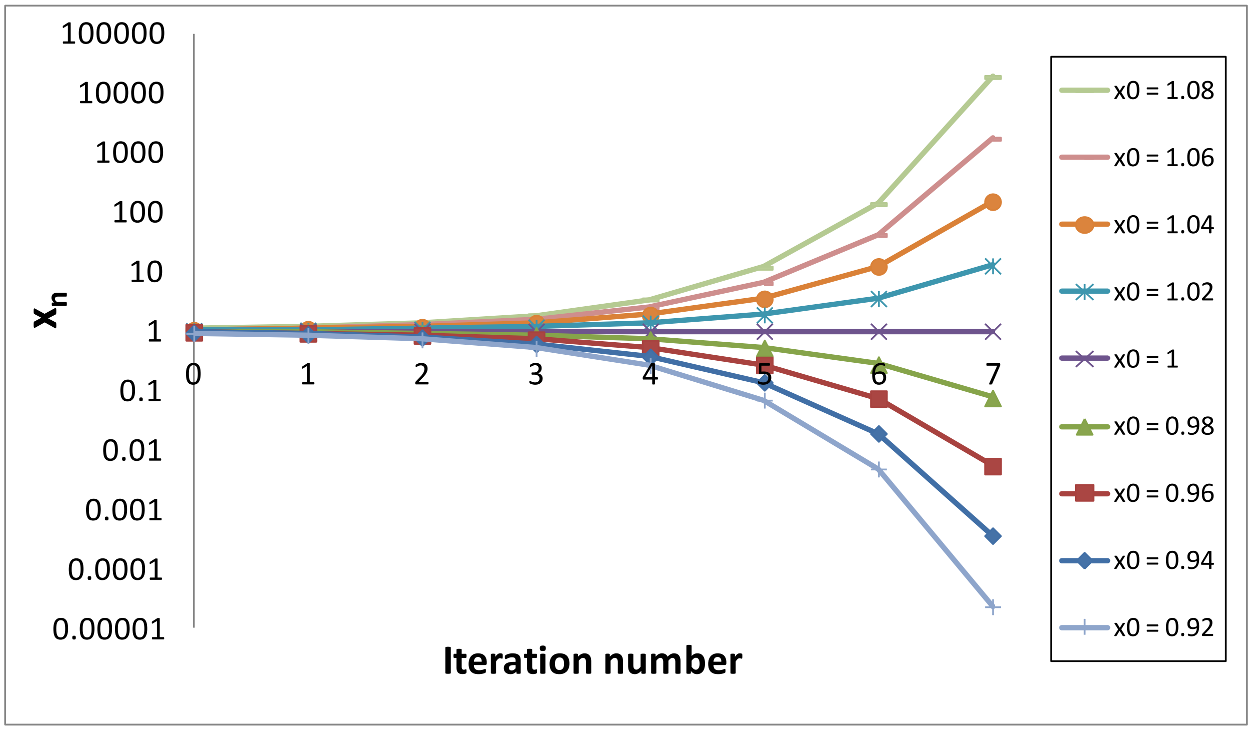


Figure 1. Example of iteration, a prisoner set, an escape set, and a Julia set in one dimension, x, which is represented on the vertical axis. Note log scale. Here $x_{n+1} = x_n^2$. Iterations with seed values $x_0 < 1$ rapidly dive toward zero. These seed values constitute the prisoner set. Iterations with seed values $x_0 > 1$ rapidly climb toward infinity. These seed values constitute the escape set. The boundary at point $x_0 = 1$ is the Julia set.

Consider the following one-dimensional example. Suppose $f(x) = x^2$, and $x_0 = 2$. Then, as $n \to \infty$, $x_n \to \infty$. However, if $x_0 = ½$, then as $n \to \infty$, $x_n \to 0$. The set of all seed values, x_0, for which for which the iterated values $x_n \to \infty$ is called the escape set. The set of all seed values of x_0 for which iterated values, x_n, remain finite is called the prisoner set. As shown in Figure 1 for this one-dimensional example, the escape set includes all $x_0 > 1$, and the prisoner set includes all $x_0 < 1$. The boundary between the escape set and the prisoner set is called the Julia set. Here the Julia set is $\{1\}$. In general, the prisoner set is the set of seed values, x_0, for which the iterated values, x_n, remain bounded as n approaches infinity. The escape set is the set of seed values, x_0, for which the iterated values, x_n, are unbounded as n approaches infinity. The Julia set includes any points at the border between the prisoner set and the escape set.

When this process is extended to two dimensions the Julia sets in some cases become extremely intricate and infinitely long curves in the plane showing increasingly finer, but self-similar detail, no matter how much that a region of the plane is magnified. Such an infinitely long pattern in two dimensions is called a fractal. Many beautiful fractal-like patterns can be found in nature, including trees, mountains, and coastlines.

Classically, fractals are generated mathematically in two dimensions by the iteration of functions of complex numbers, $z = x + iy$, for ordinary, real numbers x and y and $i = \sqrt{-1}$. Complex numbers can be represented in the complex plane with x-values on the horizontal axis and y-values on the vertical axis. In the present paper, instead of working exclusively in the complex plane, which has restrictive and particular rules, we explore new types of fractals generated by iteration of the arbitrary functions, f_1 and f_2 in two dimensions, namely

$$x_{n+1} = f_1(x_n, y_n) \tag{1a}$$

$$y_{n+1} = f_2(x_n, y_n) . \tag{1b}$$

This concept of translating the complex plane into a more general two dimensional discrete map was described previously by Wang, Chang, and Gu (2007) and also by Sprott and Pickover (1995). It provides a general and mathematically accessible framework for creation of artistic patterns for algorithmic art, many of which are fractal in nature. For any given pair of functions, f_1 and f_2, a computer program can test all possible values of seed values, x_0, y_0 , to check for escape behavior. Then, all points x_0, y_0 in a selected region of the plane can be color coded by escape speed, as follows. Define a grid-like two-dimensional array of discrete points, x_0, y_0, in the plane, usually near the origin, with each point representing a picture element (pixel) in the final digital image. For each particular x_0, y_0, begin the iteration process $x_{n+1} = f_1(x_n, y_n)$ and $y_{n+1} = f_2(x_n, y_n)$ for iterations n = 0, 1, 2, 3, ..., n_{max}. Continue iteration until either the distance of point x_n, y_n from the origin exceeds a certain preset limit, R, or until the number of iterations equals the maximum allowable number of iterations, n_{max}. If $x_n^2 + y_n^2 > R^2$ for the "practical escape radius", R, then stop iteration and assign point (x_0, y_0) to the escape set. Otherwise, if n = n_{max} and $x_n^2 + y_n^2 < R^2$, then assign point (x_0, y_0) to the prisoner set. Furthermore, for points x_0, y_0 in the escape set use the last tested value of n (call this value n*) as a measure of the escape speed. Smaller values of n* indicate faster escape.

Using n* in this way, the two dimensional map can be color coded by escape speed for artistic effect. If $n^* = n_{max}$, then the pixel centered at x_0, y_0 is colored to represent the prisoner set. If $n^* < n_{max}$, then the pixel centered at x_0, y_0 is colored differently as a function of n*. In this way the colors of points outside the prisoner set are assigned according to escape speed in a way that highlights the boundary of the prisoner set.

A general algorithm for making such escape speed images is listed in Box 1.

Box 1. General computational algorithm for fractal image generation

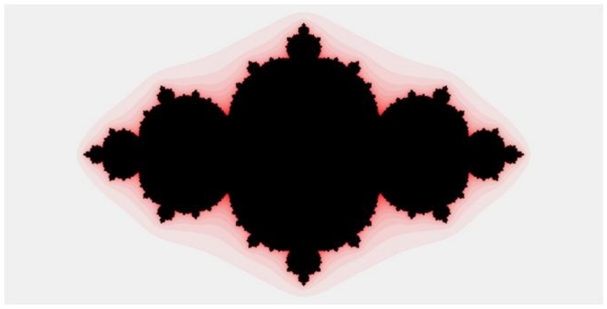
```
set escape radius, R, and maximum number of iterations, $n_{max}$

read image dimensions, image resolution, parameters for functions $f_1$ and $f_2$,
and color parameters

open image file, write image name and format specifications

for each pixel

    determine coordinates $x_0$, $y_0$

    for n = 0 to $n_{max} - 1$

        compute $x_{n+1}$ and $y_{n+1}$ using specified functions $f_1$ and $f_2$

        if $x_{n+1}^2 + y_{n+1}^2 > R^2$, then exit this for loop

    next n

    assign desired red, green, and blue (RGB) color levels (0 to 255)
    to the selected pixel as specified functions, $f_R(n)$, $f_G(n)$, $f_B(n)$

    write RGB coded image data in a convenient format

next pixel

close image file
```

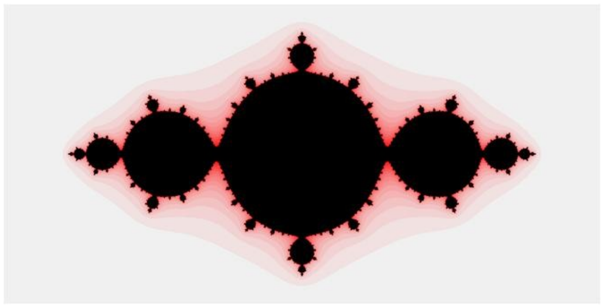
Figure 2 shows a set of escape speed images for iteration of the functions

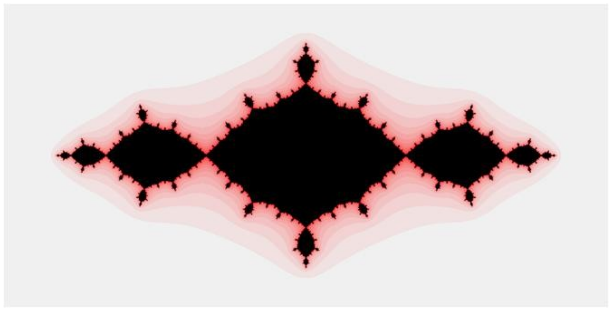
$$x_{n+1} = x_n^2 - y_n^2 + c \tag{2a}$$

$$y_{n+1} = 2x_n y_n \tag{2b}$$

with tunable parameter, c . The images are reminiscent of the famous Mandelbrot set.


(a)


(b)


(c)

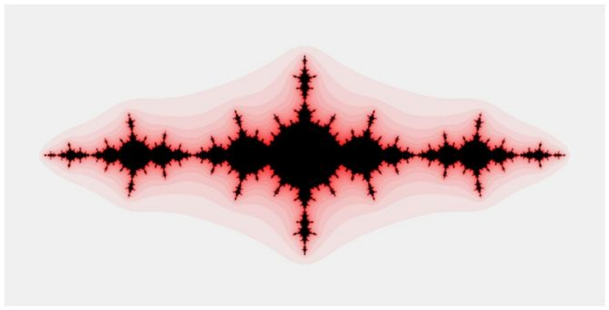

(d)

Figure 2. Escape-time fractal for iteration of Equations 2(a) and 2(b). Practical escape radius, R = 8.4. The number of the last completed iteration, n*, is the iteration number when escape distance from the origin becomes > R, or if there is no escape in n_{max} iterations, then n* = n_{max}. Prisoner set is black (n* = 20); near field escape set is pink $5 \le n^* < 20$; far field escape set is gray (n* < 5). The constant c is a tunable parameter. In (a) c = –0.7. In (b) c = –0.9. In (c) c = –1.1. In (d) c = –1.3. Numerically, the horizontal range of the images extends from x = –2 to x = 2. The vertical range extends from y = –1 to y = 1.

The patterns in Figure 2 are true fractals, as is shown for case c = –0.9 in Figure 2(b) by the expanded scale and higher resolution representation shown in Figure 3.

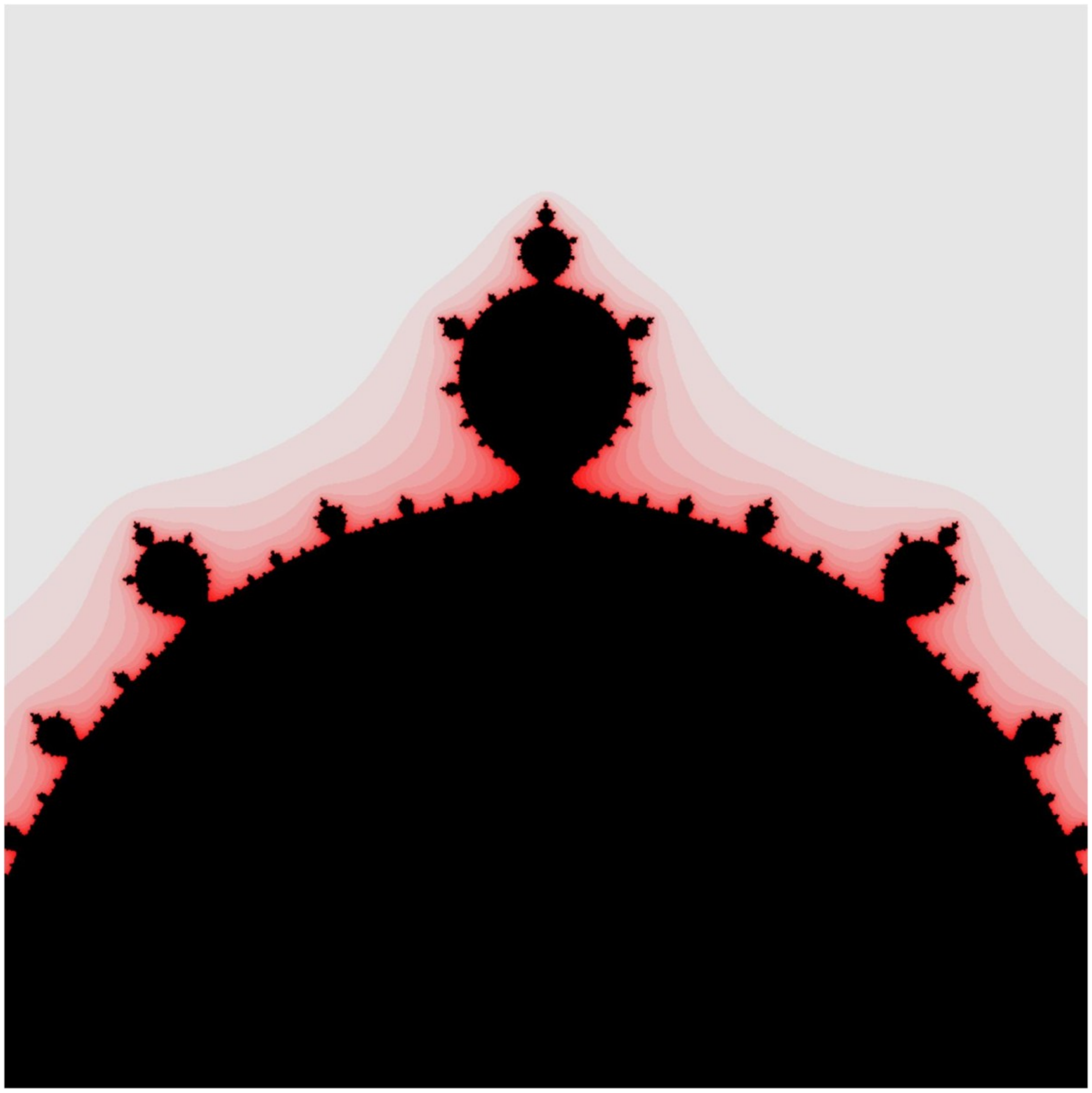


Figure 3. Escape-time fractal for iteration of Equations 2(a) and 2(b) for c = –0.9. Prisoner set is black; near field escape set is pink; far field escape set is gray. Numerically, the horizontal range of the image extends from x = –0.5 to x = 0.5. The vertical range extends from y = 0 to y = 1.

Figures 4 and 5 show even higher resolution and more expanded views of the fractal pattern in Figure 3. Figure 4 represents 20-fold magnification of Figure 2(b). Figure 5 represents 200-fold magnification. The fractal pattern is preserved across magnification by more than two orders of magnitude.

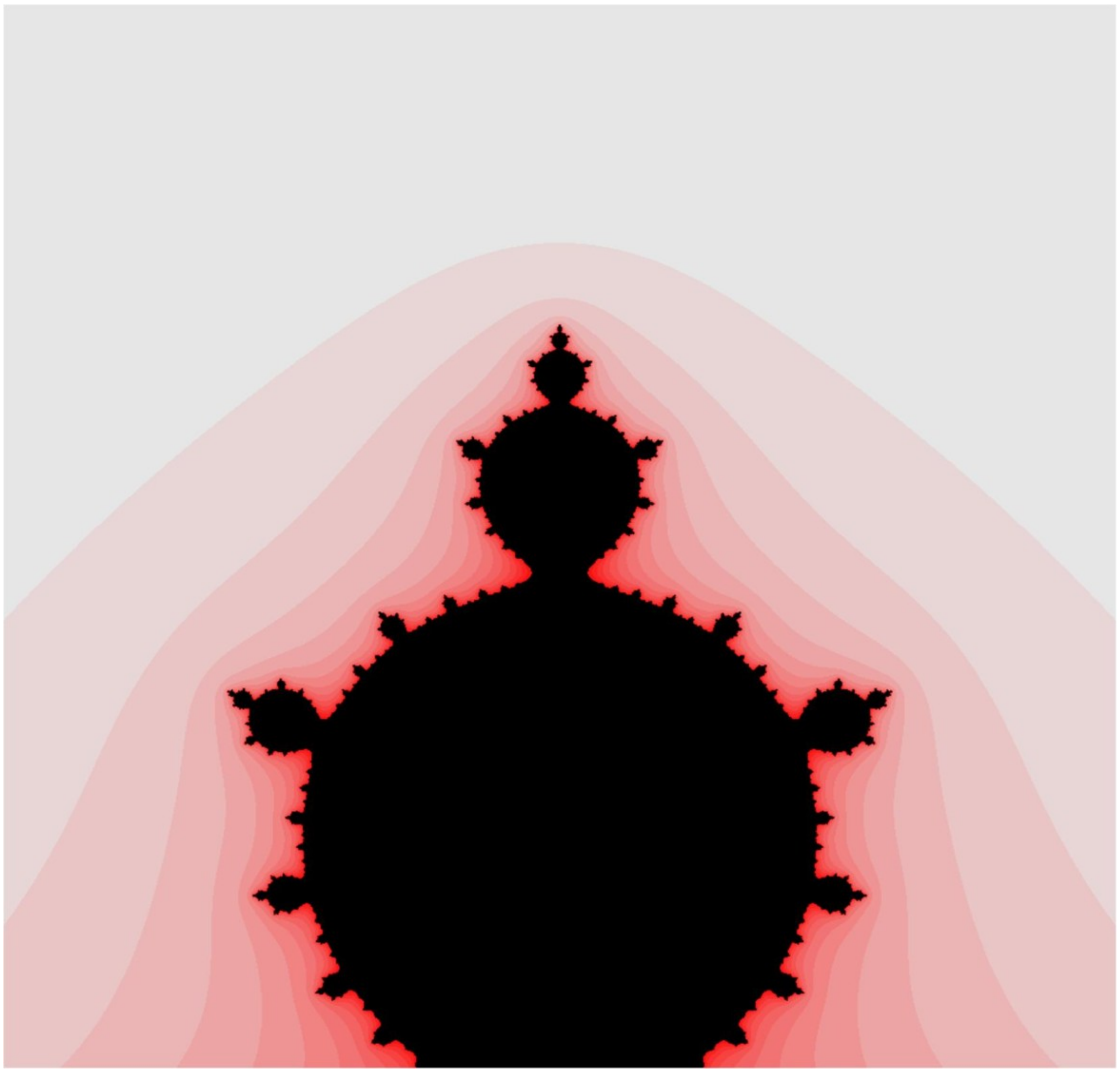


Figure 4. Escape-time fractal for iteration of Equations 2(a) and 2(b) for c = –0.9. Prisoner set is black; near field escape set is pink; far field escape set is gray. Numerically, the horizontal range of the image extends from x = –0.05 to x = 0.05. The vertical range extends from y = 0.75 to y = 0.85.

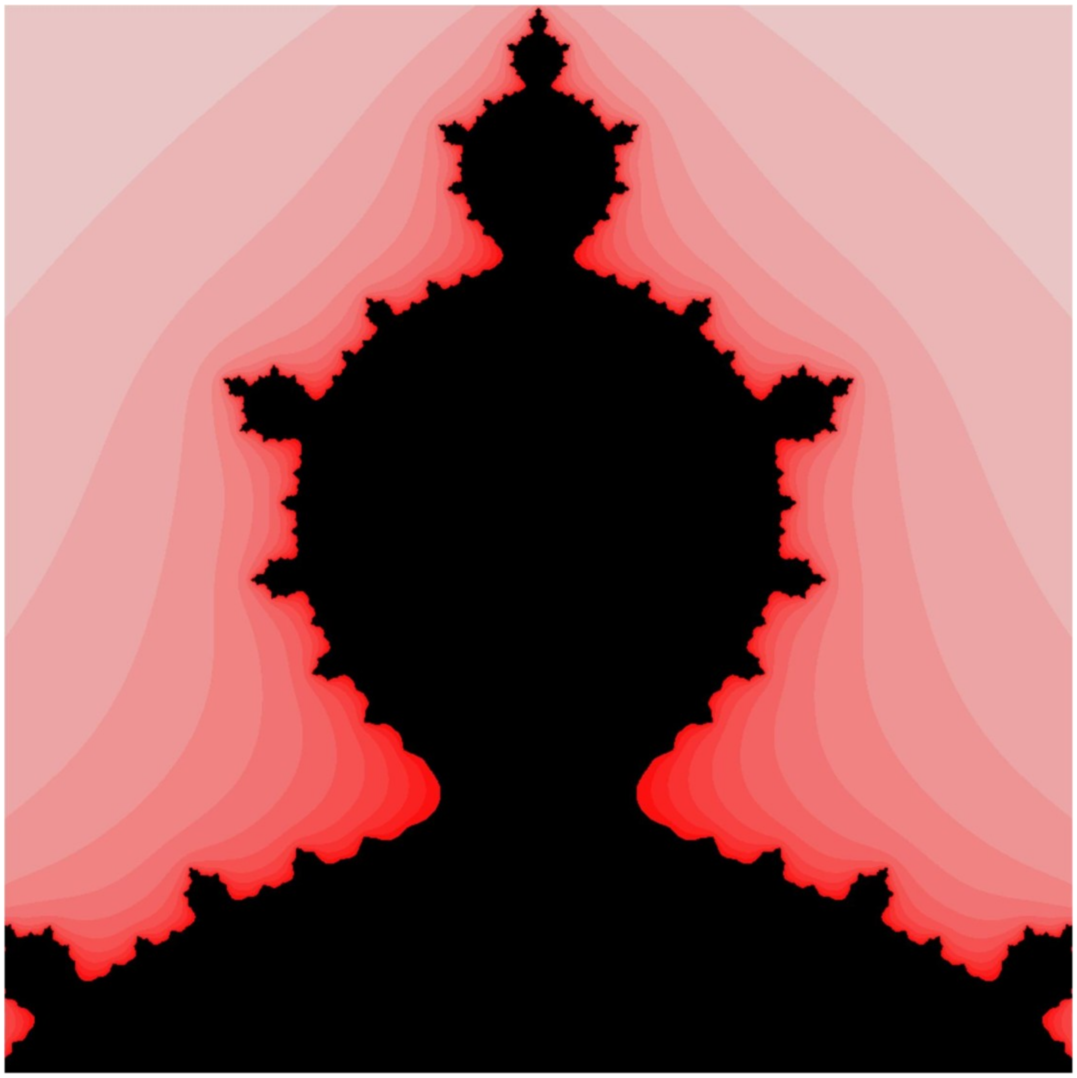


Figure 5. Escape-time fractal for iteration of Equations 2(a) and 2(b) for c = –0.9. Prisoner set is black; near field escape set is pink; far field escape set is gray. Numerically, the horizontal range of the image extends from x = –0.005 to x = 0.005. The vertical range extends from y = 0.81 to y = 0.82.

The present faculty working paper shows how to create a rich variety of complex and fascinating fractals using the general approach of Equations (1) and techniques that are accessible to students with high school level skills in mathematics and coding.

Fractals created by iteration of general polynomial functions, raised to an integer power

The forgoing framework provides an easy way to specify iteration formulas for fractal patterns. For example, consider N-th order polynomial functions, raised to an integer power, p,

$$f_1(x,y)=\begin{bmatrix} a_{00} & +a_{01}x & +a_{02}x^2 & +a_{03}x^3 & +\cdots+a_{0N}x^N \\ +a_{10}y & +a_{11}xy & +a_{12}x^2y & +a_{13}x^3y & +\cdots+a_{1N}x^Ny \\ +a_{20}y^2 & +a_{21}xy^2 & +a_{22}x^2y^2 & +a_{23}x^3y^2 & +\cdots+a_{2N}x^Ny^2 \\ +\cdots & & & & \\ +a_{N0}y^N & +a_{N1}xy^N & +a_{N2}x^2y^N & +a_{N3}x^3y^N & +\cdots+a_{NN}x^Ny^N \end{bmatrix}^p \tag{3a}$$

$$f_2(x,y)=\begin{bmatrix} b_{00} & +b_{01}x & +b_{02}x^2 & +b_{03}x^3 & +\cdots+b_{0N}x^N \\ +b_{10}y & +b_{11}xy & +b_{12}x^2y & +b_{13}x^3y & +\cdots+b_{1N}x^Ny \\ +b_{20}y^2 & +b_{21}xy^2 & +b_{22}x^2y^2 & +b_{23}x^3y^2 & +\cdots+b_{2N}x^Ny^2 \\ +\cdots & & & & \\ +b_{N0}y^N & +b_{N1}xy^N & +b_{N2}x^2y^N & +b_{N3}x^3y^N & +\cdots+b_{NN}x^Ny^N \end{bmatrix}^p, \tag{3b}$$

or in more compact notation,

$$x_{n+1}=f_1(x_n,y_n)=\left[\sum_{i=0}^{N}\sum_{j=0}^{N}a_{ij}x_n^j y_n^i\right]^p \tag{3c}$$

$$y_{n+1}=f_2(x_n,y_n)=\left[\sum_{i=0}^{N}\sum_{j=0}^{N}b_{ij}x_n^j y_n^i\right]^p, \tag{3d}$$

where p may be either a positive or negative integer. (If p = 0 we have the degenerate prisoner set of {1, 1}.) The form of the fractal is determined by the coefficient matrices

$$\mathbf{A}=\begin{bmatrix} a_{11} & a_{12} & a_{13} & \ldots \\ a_{21} & a_{22} & a_{23} & \ldots \\ a_{31} & a_{32} & a_{33} & \ldots \\ \ldots & \ldots & \ldots & \ldots \end{bmatrix} \text{ and } \mathbf{B}=\begin{bmatrix} b_{11} & b_{12} & b_{13} & \ldots \\ b_{21} & b_{22} & b_{23} & \ldots \\ b_{31} & b_{32} & b_{33} & \ldots \\ \ldots & \ldots & \ldots & \ldots \end{bmatrix}. \tag{4}$$

Here we examine several examples of the iteration of such polynomial functions.

Consider iteration of the second order polynomial function shown in Figure 6, for which

$$x_{n+1} = x_n^2 - y_n^2 - 1 \tag{5a}$$

$$y_{n+1} = x_n + y_n + 2x_n y_n, \tag{5b}$$

$$\mathbf{A} = \begin{matrix} -1 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{matrix}, \quad \mathbf{B} = \begin{matrix} 0 & 1 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 0 \end{matrix}, \text{ and power, } p = 1.$$

Addition of the x and y terms to Equation (2b) transforms a lumpy fractal pattern into a spikey fractal pattern that is reminiscent of stylized sailing ships.

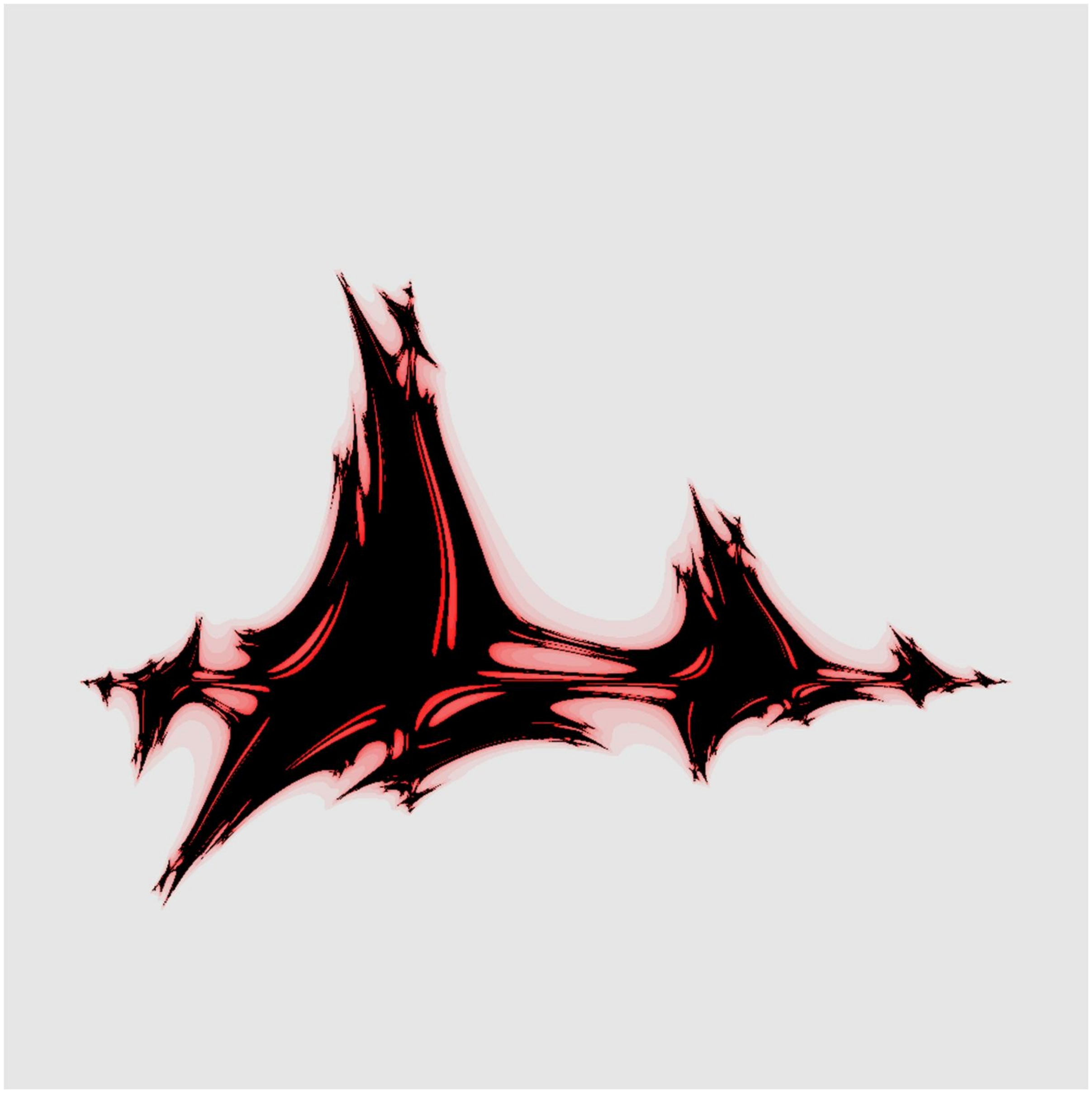


Figure 6. Escape-time fractal in the plane for iteration of Equations (5). Practical escape radius, R = 8.4. Iteration number when escape distance from origin becomes > R = n*. Prisoner set is black (n* ≥ 20); near field escape set is pink 5 ≤ n* < 20; far field escape set is gray (n* < 5). Numerically, the horizontal range of the image extends from x = –2 to x = 2. The vertical range extends from y = –2 to y = 2.

The fractal nature of the pattern in Figure 6 can be revealed by magnification of the image, as shown in Figures 7 and 8, which correspond to 20X and 40X magnification, respectively.

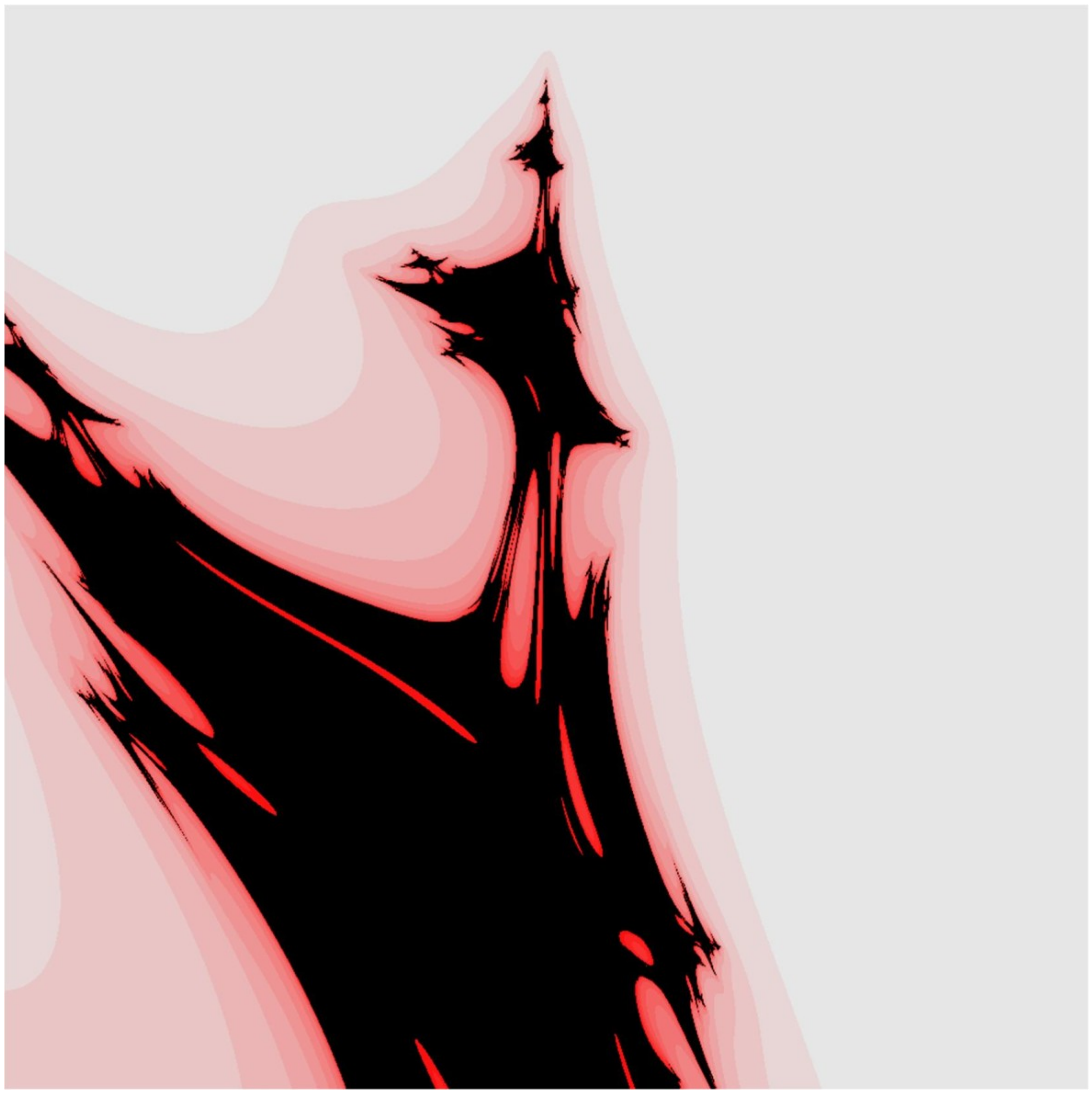


Figure 7. Escape-time fractal in the plane for iteration of Equations (5), corresponding to the top portion of Figure 6. Practical escape radius, R = 8.4. Iteration number when escape distance from origin becomes > R = n*. Prisoner set is black (n* ≥ 20); near field escape set is pink 5 ≤ n* < 20; far field escape set is gray (n* < 5). Numerically, the horizontal range of the image extends from x = –0.6 to x = –0.4. The vertical range extends from y = 0.8 to y = 1.0.

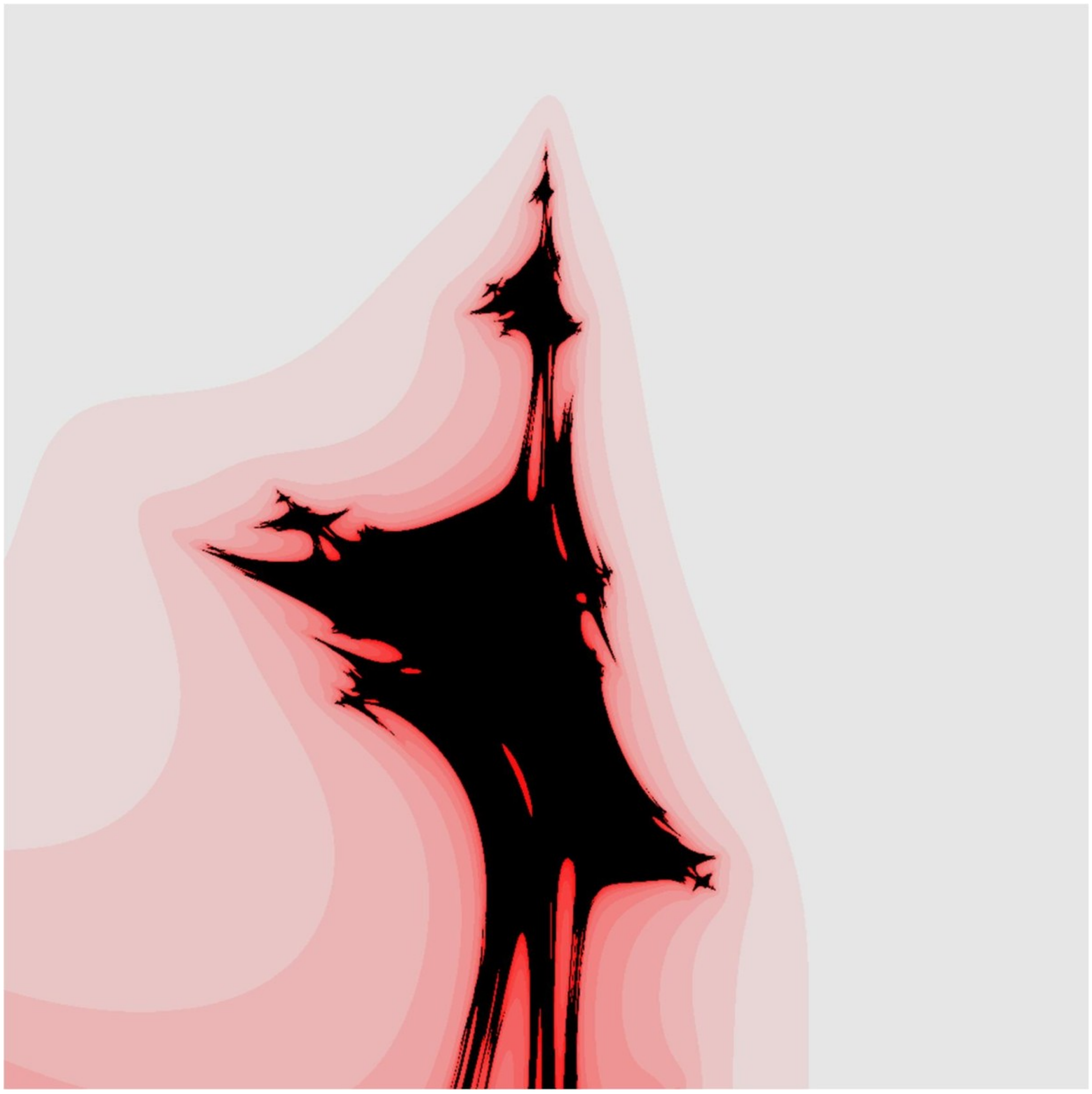


Figure 8. Escape-time fractal in the plane for iteration of Equations (5), corresponding to the top part of Figure 7. Practical escape radius, R = 8.4. Iteration number when escape distance from origin becomes > R = n*. Prisoner set is black ($n^* \geq 20$); near field escape set is pink $5 \leq n^* < 20$; far field escape set is gray ($n^* < 5$). Numerically, the horizontal range of the image extends from x = –0.55 to x = –0.45. The vertical range extends from y = 0.9 to y = 1.0.

Inverse power transformation can lead to dramatically different results. Consider Equations (6):

$$x_{n+1} = x_n^2 - y_n^2 - 1 \tag{6a}$$

$$y_{n+1} = 2 + 2x_n y_n + 2x_n^2 y_n^2 \ , \tag{6b}$$

for which

$$\mathbf{A} = \begin{matrix} -1 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{matrix} \ , \ \mathbf{B} = \begin{matrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{matrix} \ , \text{ and power, } p = -1.$$

For exponent $p = -1$, an interesting pattern appears, as shown in Figures 9(a) through 9(d). The dotted box in Figure 9(a) indicates approximately the region expanded to create Figure 9(b), etc. The structures are reminiscent of the rings of the planet Saturn in that the greater the magnification, the more detail is seen. The self-similar pattern of rings is revealed with successively greater and greater magnification. This kind of self-similarity at different scales is a characteristic of fractals. The magnification in Figure 9(d) is over 100X that in Figure 9(a). Thus Equations (6) appear to produce a new type of striped fractal.

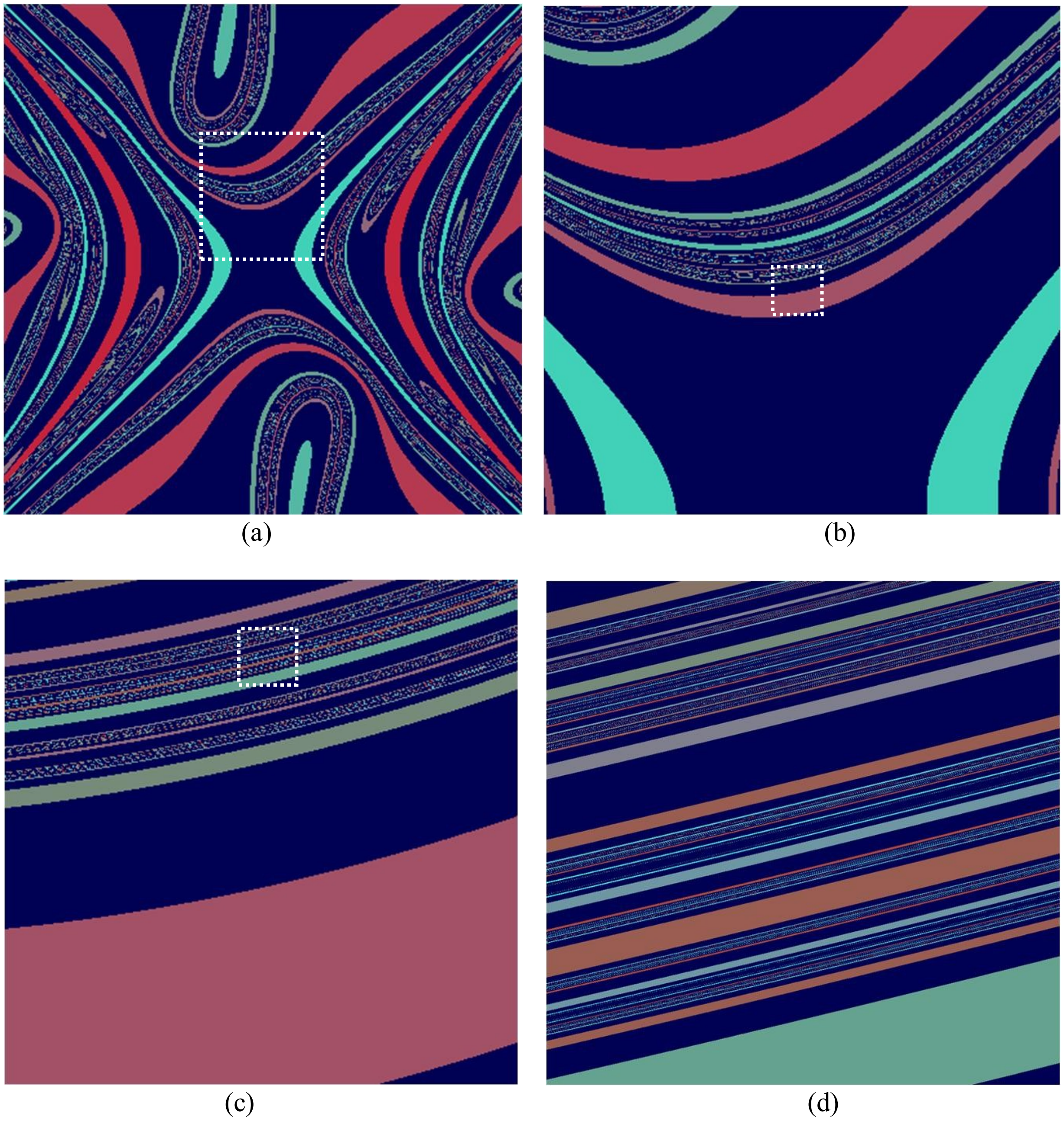


Figure 9. Escape-time fractal in the plane for iteration of Equations (6). Practical escape radius, R = 8.4. Iteration number when escape distance from origin becomes > R = n*. Prisoner set is red ($n^* \geq 20$); near field escape set is pink to green $5 \leq n^* < 20$; far field escape set is blue ($n^* < 5$). Numerically, the ranges of the images extend from (a) $x = -2$ to $x = 2$ and $y = -2$ to $y = 2$; (b) $x = -0.5$ to $x = 0.5$ and $y = 0$ to $y = 1$; (c) $x = -0.05$ to $x = 0.05$ and $y = 0.4$ to $y = 0.5$; and (d) $x = -0.005$ to $x = 0.005$ and $y = 0.48$ to $y = 0.49$. Dotted boxes indicate expanded regions.

Figure 10 depicts iteration of the functions (7a) and (7b),

$$x_{n+1} = y_n + x_n^2 y_n + x_n y_n^2 - y_n^3 \tag{7a}$$

$$y_{n+1} = x_n + x_n^3 + 3x_n^2 + 3y_n^2, \tag{7b}$$

for which

$$\mathbf{A} = \begin{matrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{matrix}, \quad \mathbf{B} = \begin{matrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 3 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{matrix}, \text{ and power, } p = 1.$$

The resulting dragon head structures are new and unusual. In this example the prisoner set and the Julia set look connected.

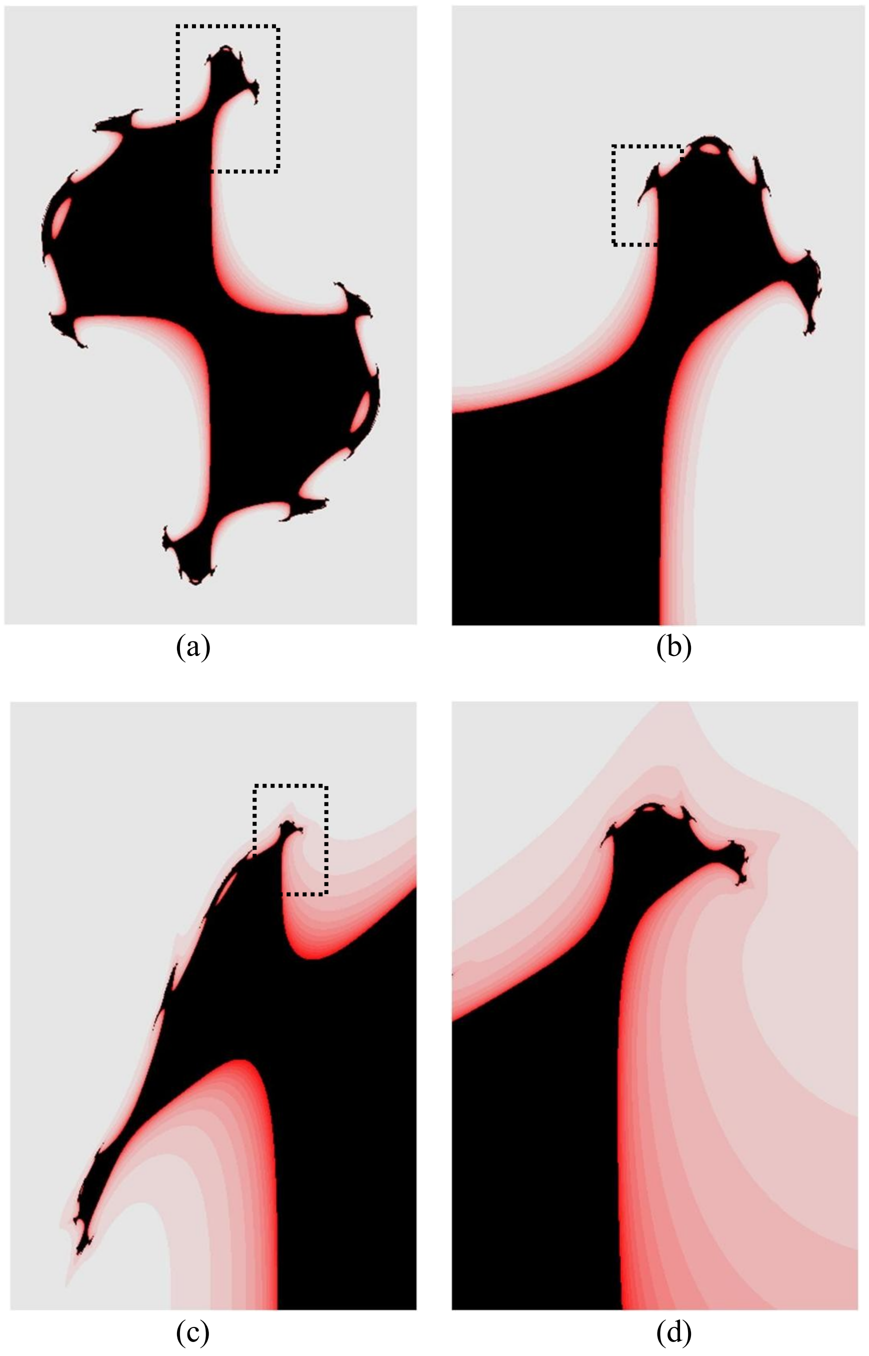


Figure 10. Escape-time fractal in the plane for iteration of Equations (7). Practical escape radius, R = 8.4. Iteration number when escape distance from origin becomes > R = n*. Prisoner set is black ($n^* \geq 20$); near field escape set is pink $5 \leq n^* < 20$; far field escape set is gray ($n^* < 5$). Numerically, the ranges of the images extend from (a) x = –1 to x = 1 and y = –1.5 to y = 1.5; (b) x = –0.3 to x = 0.3 and y = 0.6 to y = 1.5; (c) x = –0.04 to x = 0.02 and y = 1.20 to y = 1.29; and (d) x = –0.004 to x = 0.006 and y = 1.260 to y = 1.275. Dotted boxes indicate expanded regions.

Figures 11 and 12 illustrate two more varied examples, in which Equations (8) and (9) are used to generate different fractal patterns. These patterns do not have reflection symmetries. However, the self-similar repeating patterns are clearly present, highly angular, and less frequent than in the classical fractal patterns, such as those of the Mandelbrot set. To generate Figure 11 we have

$$x_{n+1} = y_n + x_n y_n + x_n^2 y_n - y_n^3 \tag{8a}$$

$$y_{n+1} = x_n + 3x_n^2 y_n + 3x_n y_n^2 + 3x_n^2 y_n^2 \ , \tag{8b}$$

for which

$$\mathbf{A} = \begin{matrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{matrix} \ , \ \mathbf{B} = \begin{matrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 3 & 3 & 0 \\ 0 & 0 & 0 & 0 \end{matrix} \ , \text{ and power, } p = 1.$$

Equations (9) may be used to generate an even more abstract form of fractal:

$$x_{n+1} = -1 + x_n^2 y_n + x_n y_n^2 \tag{9a}$$

$$y_{n+1} = x_n y_n + x_n^2 y_n^2 - y_n^2 \tag{9b}$$

for which

$$\mathbf{A} = \begin{matrix} -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{matrix} \ , \ \mathbf{B} = \begin{matrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{matrix} \ , \text{ and power, } p = 1.$$

The resulting pattern is shown in Figure 12.

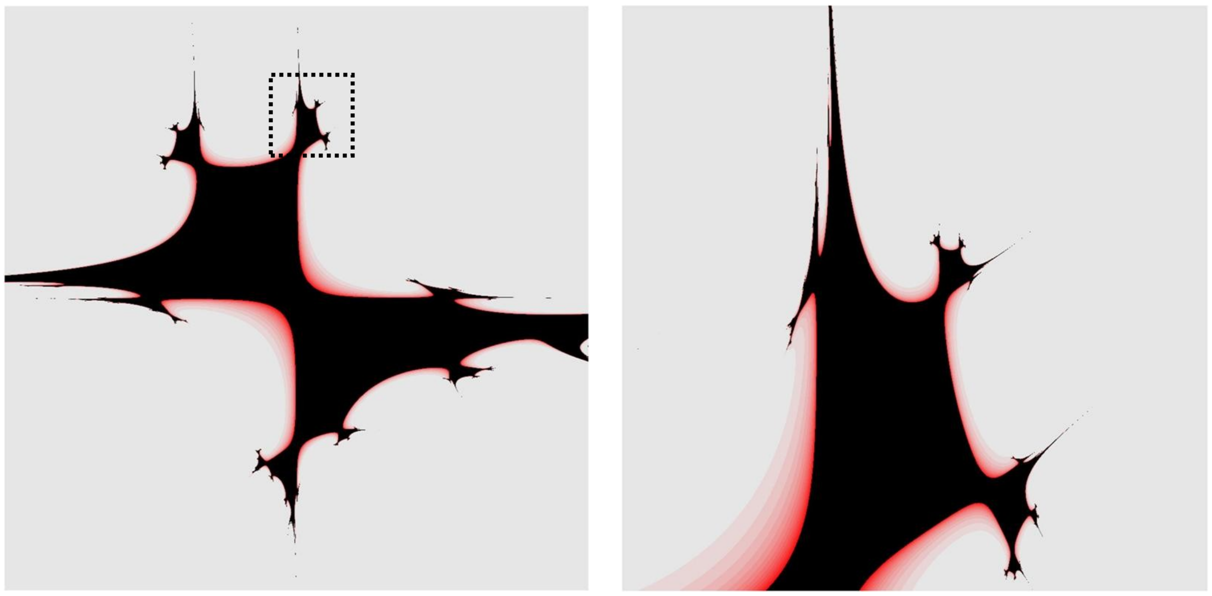


Figure 11. Escape-time coded fractal in the plane for iteration of Equations (8). Practical escape radius, R = 8.4. Iteration number when escape distance from origin becomes > R = n*. Prisoner set is black (n* ≥ 20); near field escape set is pink 5 ≤ n* < 20; far field escape set is gray (n* < 5). Numerically, the ranges of the images extend from (a) x = –2 to x = 2 and y = –2 to y = 2; (b) x = –0.2 to x = 0.4 and y = 1 to y = 1.6. Dotted box indicates expanded region.

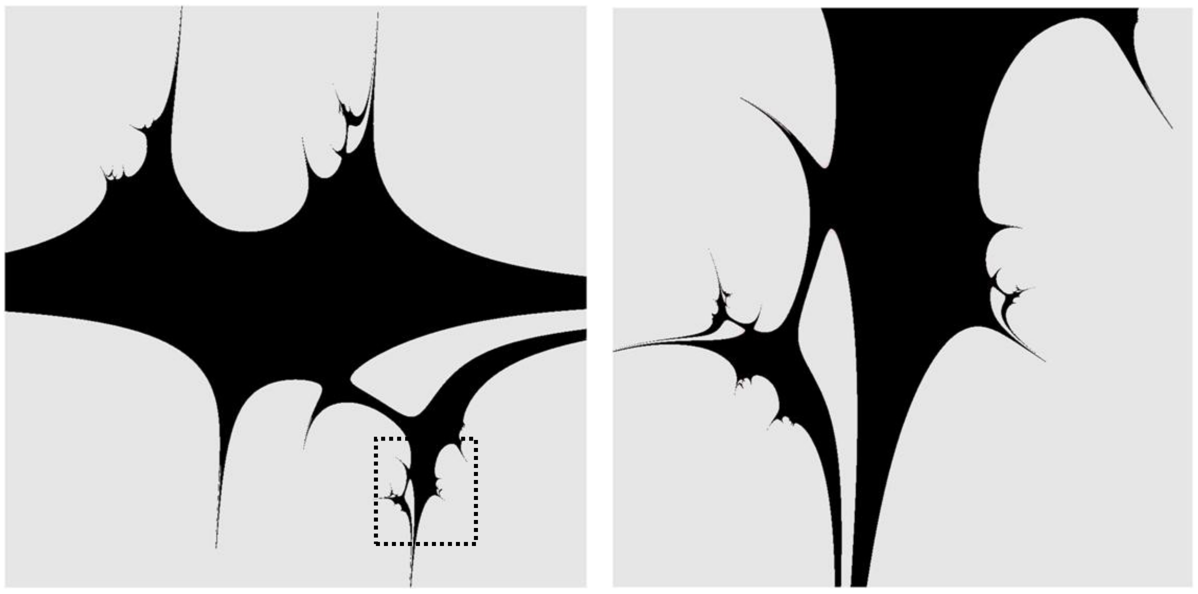


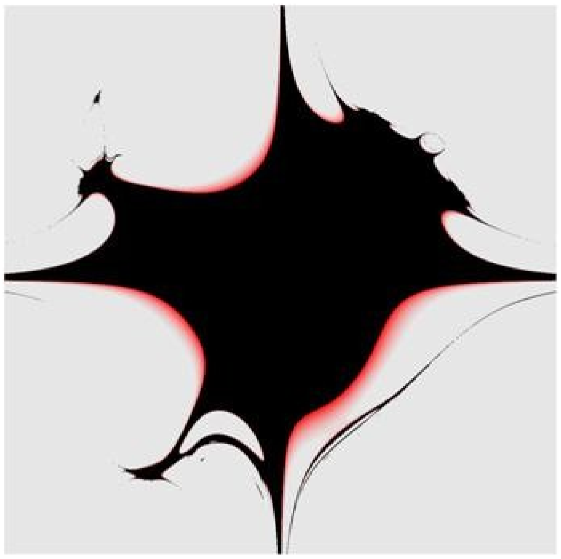
Figure 12. Escape-time coded fractal in the plane for iteration of Equations (9). Practical escape radius, R = 8.4. Iteration number when escape distance from origin becomes > R = n*. Prisoner set is black (n* ≥ 20); near field escape set is pink 5 ≤ n* < 20; far field escape set is gray (n* < 5). Numerically, the ranges of the images extend from (a) x = –3 to x = 3 and y = –2 to y = 2; (b) x = 0.8 to x = 1.8 and y = –2.5 to y = –1.5. Dotted box indicates expanded region.

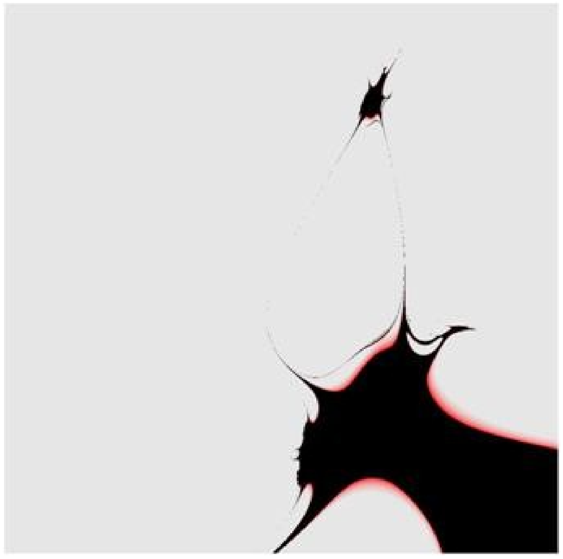
Figure 13 shows the results of just one foray into the realm of randomly chosen, higher order polynomials. The functions in Equations (10) were selected at random, followed by some fine tuning of one coefficient (–3) to enhance visual interest and complexity.

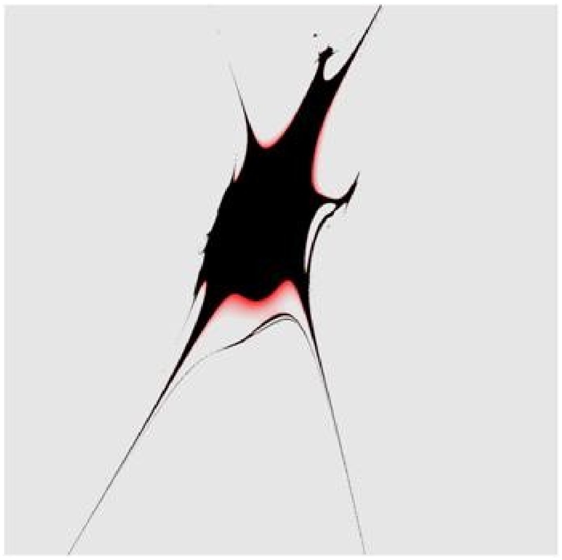
$$f_1 = -1 + xy^2 + x^3y^5 + x^6y^3 \qquad (10a)$$

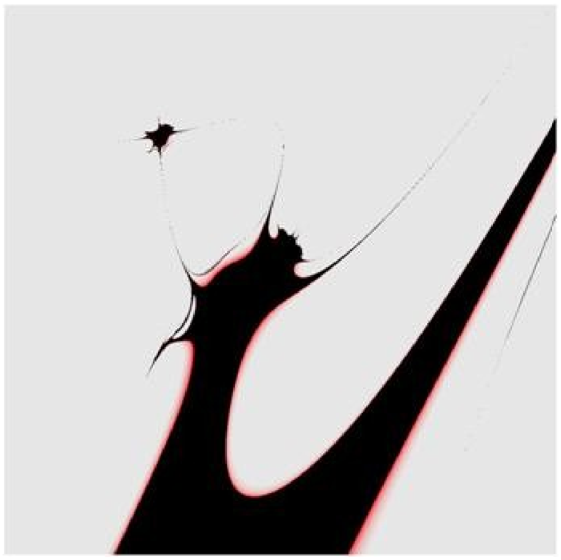
$$f_2 = xy^5 + x^4y - 3x^4y^4 + x^6y^6 \qquad (10b)$$

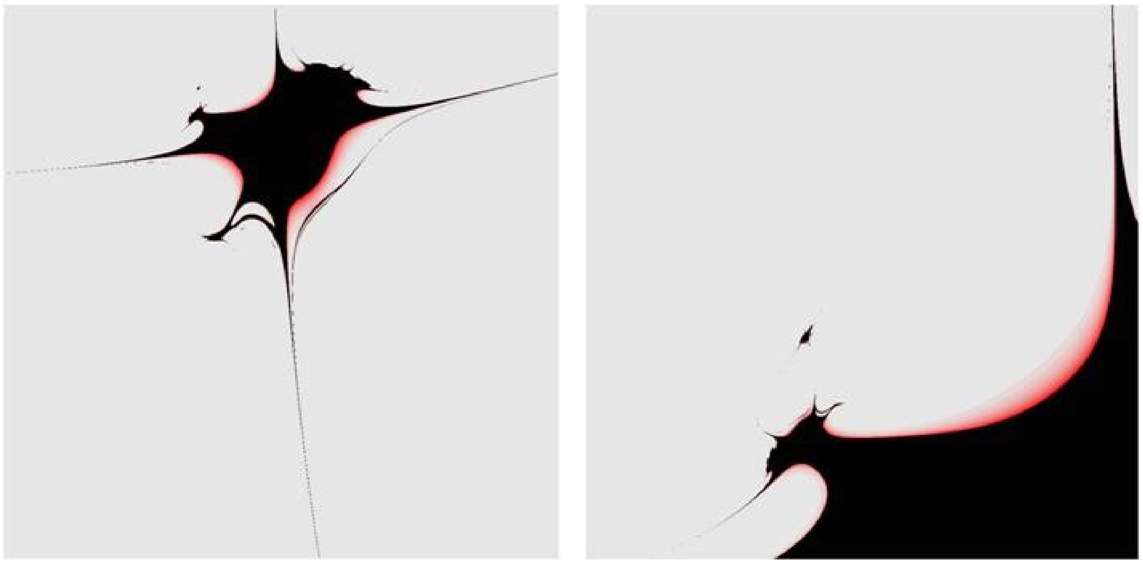
In frames (b) through (f) the region including the small island at the top left of Figure 13(a) is successively magnified. The small island, itself has a small island, which in turn has a small island that is clearly visible at 4000X magnification. This third generation island comprised less than one quarter pixel in the original image of Figure 13(a). An invisible speck in the original 2 unit by 2 unit square is itself a fractal pattern that can be magnified indefinitely.


(a) 1X


(b) 4X


(c) 40X


(d) 160X



(e) 1000X (f) 4000X

Figure 13. Fractals generated by iteration of Equations (10). The horizontal and vertical ranges in dimensionless units are –2 to 2 and –2 to 2, respectively in (a); –2 to –1 and 0.5 to 1.5 in (b); –1.43 to –1.23 and 1.2 to 1.4 in (c); –1.325 to –1.300 and 1.370 to 1.395 in (d); –1.320 to –1.316 and 1.386 to 1.390 in (e); and –1.319 to –1.318 and 1.389 to 1.390 in (f).

Discussion

Here we explore the use of general polynomial functions, followed by optional power function transformation, as fractal generators. The use of complex arithmetic, based on $\sqrt{-1}$, is not required to create fractal patterns. Paired polynomial functions of ordinary real numbers, subjected to iteration in the plane, can be used to create artistic two-dimensional maps of prisoner, escape, and Julia sets with genuine fractal properties. A large contingent of potential fractal patterns awaits exploration, including many undiscovered forms, both organic, and otherworldly. Only pre-college math and programming skills are required. Using this approach, students of many levels of mathematical ability can search for novel types of fractals in a general way, realizing that not all combinations of functions, f_1 and f_2, will produce genuine fractals, but that other combinations of functions may lead to new and undiscovered forms.

Even further generalization of these simple concepts is straightforward. The trick of raising the functions f_1 and f_2 to an arbitrary positive or negative power, p, namely $f_1(x_n, y_n) = \left[\sum_{i=0}^{N}\sum_{j=0}^{N} a_{ij} x_n^j y_n^i\right]^p$ and $f_2(x_n, y_n) = \left[\sum_{i=0}^{N}\sum_{j=0}^{N} b_{ij} x_n^j y_n^i\right]^p$, is itself a specific example of a broader concept for adding additional variety using a secondary function g(f) to obtain $f_1'(x_n, y_n) = g(f_1(x_n, y_n))$, and $f_2'(x_n, y_n) = g(f_2(x_n, y_n))$. In the case of raising to a power, g(u) = u^p. Moreover, if desired, this process could be continued to obtain arbitrarily complicated forms

$f''_1(x_n, y_n) = h(g(f_1(x_n, y_n)))$, $f''_2(x_n, y_n) = h(g(f_2(x_n, y_n)))$, etc., as long as each generation of chained functions maps to the original plane.

Another domain of generalization involves three dimensional or higher dimensional fractals. Clearly, the forgoing process can be expanded to three or more dimensions:

$$\begin{aligned} x_{n+1} &= f_1(x_n, y_n, z_n) \\ y_{n+1} &= f_2(x_n, y_n, z_n) \\ z_{n+1} &= f_3(x_n, y_n, z_n) \end{aligned} \tag{11}$$

or

$$\begin{aligned} x_{n+1} &= f_1(x_n, y_n, z_n, w_n) \\ y_{n+1} &= f_2(x_n, y_n, z_n, w_n) \\ z_{n+1} &= f_3(x_n, y_n, z_n, w_n) \\ w_{n+1} &= f_4(x_n, y_n, z_n, w_n), \text{ etc.,} \end{aligned} \tag{12}$$

for which prisoner sets, escape sets, and Julia sets can be defined analogously to those in two dimensions. In three dimensions prisoner sets are volumes and Julia sets are surfaces. In four dimensions prisoner sets are hyper-volumes, and Julia sets are volumes, and so on. As the number of dimensions increases, the required computation time increases greatly and the challenges of graphical rendering of higher dimensional fractals become more formidable. Nevertheless, the opportunities for discovery are unlimited.

References

JC Sprott and CA Pickover, Automatic generation of general quadratic map basins, Computers & Graphics 19(2), 309-313, 1995

Wang Xing-yuan, Chang Pei-jun, and Gu Ni-ni, Additive perturbed generalized Mandelbrot–Julia sets, Applied Mathematics and Computation 189, 754–765. 2007