Fun with Newton's Method

Kerry Mitchell[1]

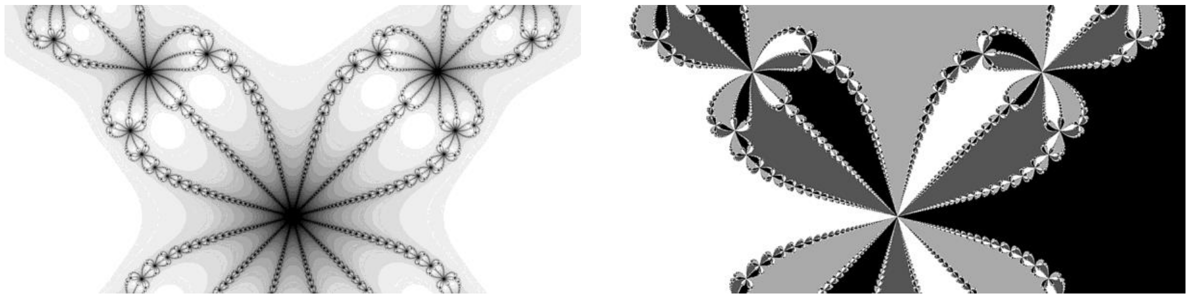
[1]Artist, Peoria, Arizona; lkmitch@gmail.com

Abstract

Newton's method is a well-known method for finding solutions to nonlinear equations. Beyond its root-finding capabilities, the chaotic nature of Newton's method has been widely explored in creating fractal images. This work presents extensions to, and variations from, the standard method, not for advancing the mathematics behind it, but for expanding its use in creating fractal art. Several examples of art using these methods are presented.

Newton's Method

Newton's method [1] is a general-purpose routine for iteratively finding solutions of equations. Because of its simple theoretical foundation, it has been employed from high school homework assignments to NASA fluid mechanics modeling [2]. Assume that the equation is $f(z) = 0$ and it is desired to find values of z such that the equation is true. Newton's method works by assuming that the function $f(z)$ is approximately linear around the solution point. That linear approximation is used to find an estimate to the solution (the value z where the linear function is 0). This new guess is used to create a new linear approximation and a new guess, etc., until either the method converges on a solution (the usual case) or does not (the fun case). Figure 1 shows the result of applying Newton's method in creating fractal images. Figure 1(a) shows the method applied to finding the solutions of $z^4 - 1 = 0$. Each pixel is used as the complex-valued starting value of z and is shaded according to how long the method takes to find a solution. In Figure 1(b), the pixels are colored according to which one of the four solutions is found. The regions of constant grayscale (basins) at the top, left, and right of both images represent cases where a solution is found quickly and directly. Between those regions are fractal structures, in this case, along rays at 45° angles to the axes. In these areas, the method takes longer to converge on a solution and does so while bouncing around. The fractal nature of these basin boundaries shows that a small change in the starting point can lead to a large change in the solution found, an indication of chaos in Newton's method. Many artists have used this as a basis for their work; see for example, Paul Bourke's images [3], Kalantari's work with polynomiography [4], and a previous piece of mine [5]. The motivation of the current work is to exploit the chaotic nature of Newton's method (and variants thereof) for the creation of images. None of the techniques discussed below are assumed to provide any mathematical benefit to the user.



(a) (b)

Figure 1: *Newton's method applied to the equation $z^4 - 1 = 0$: (a) rendered by how many iterations are needed to find a solution,(b) rendered by the angle of the solution found.*

Adding an Adjustment Factor

To provide some flexibility in the method, consider its iterative formula:

$$z_{\text{new}} = z_{\text{old}} - [f(z_{\text{old}}) / f'(z_{\text{old}})],$$

where $f(z) = 0$ is the equation to be solved and $f'(z)$ is the derivative of the function f with respect to z. (This equation derives from approximating $f(z)$ as a linear function around the point where $f(z) = 0$). The bracketed term on the right-hand side of the equation can be considered to be an adjustment to the previous approximation to the solution; if z_{old} is the solution, then $f(z_{\text{old}}) = 0$, $z_{\text{new}} = z_{\text{old}}$, and there is, in effect, no adjustment to z_{old}. By introducing a factor α, (an approach new to me), the relative effect of the adjustment term can be varied. The new iteration formula is simply:

$$z_{\text{new}} = z_{\text{old}} - \alpha\, [f(z_{\text{old}}) / f''\, (z_{\text{old}})],$$

Setting α to 1 returns the standard Newton's method.

Figure 2 shows some examples for different values of α, all using $f(z) = z^4 - 1$. As in Figure 1(a), the fractals are rendered to show how long it takes the method to converge to a solution, with darker colors indicating that more iterations are needed. As α moves away from 1, the fractal becomes more curved, complicated, and darker, but solutions are still found. In Figure 3, three cases are shown where the adjusted method does not converge on a solution. Here, the colors represent the polar angle of the final value, as in Figure 1(b). In these images, there are still (mainly) four shades of gray, suggesting four solutions, but the presence of more levels indicates that the method has not settled in on a solution.

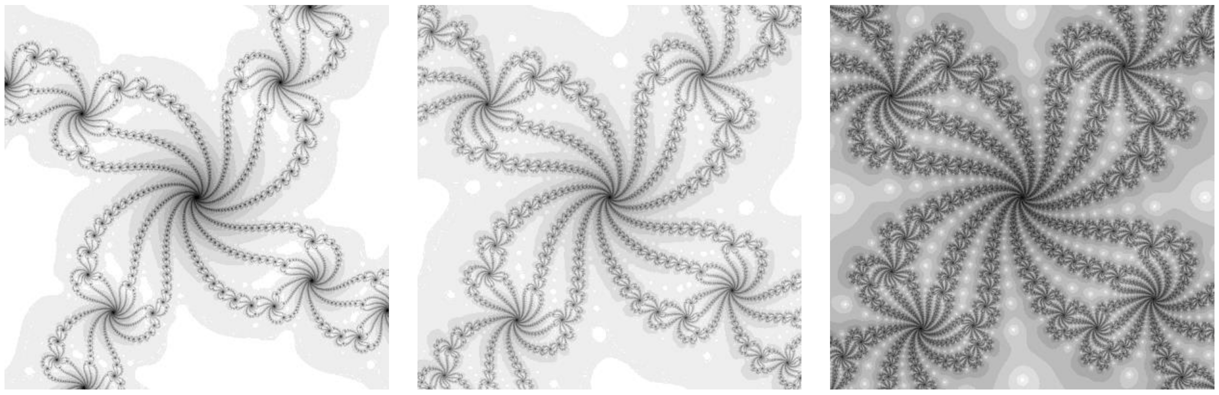
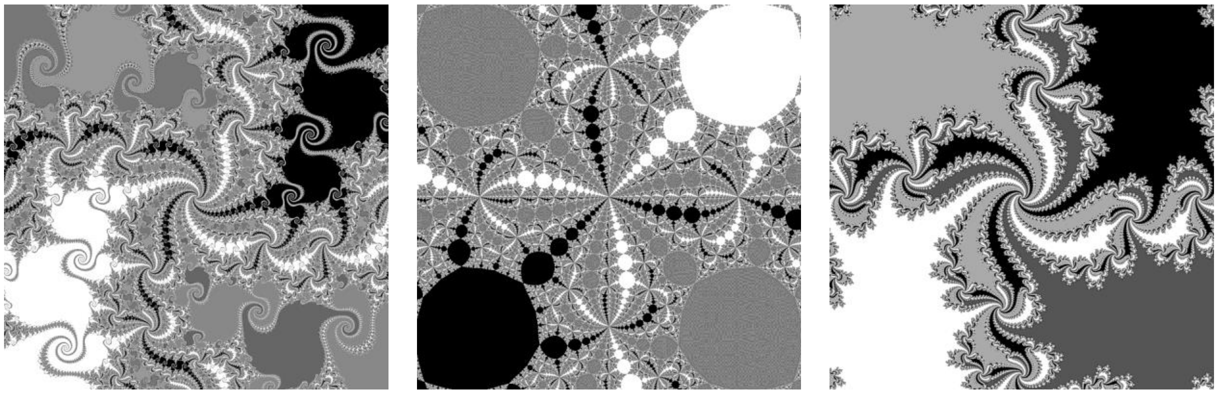

(a) (b) (c)

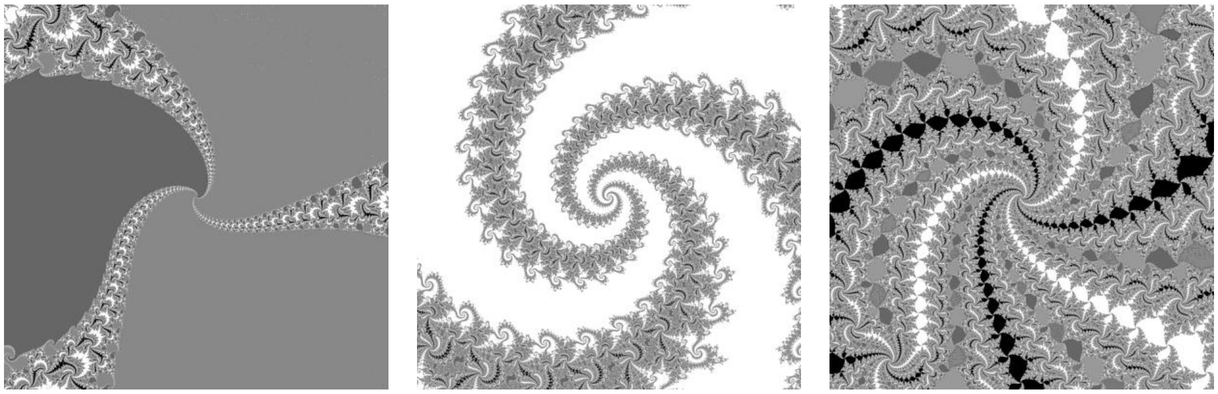
Figure 2: *adjusted Newton's method applied to $f(z) = z^4 - 1$, rendered by number of iterations required for convergence: (a) $\alpha = 1+0.5i$, (b) $\alpha = 1.3+0.5i$, (c) $\alpha = 1.6+0.5i$.*

The fractals shown in Figures 1 and 2 are conceptually similar to Julia set fractals [6]. In the standard quadratic Julia set for $z = z^2 + c$, each pixel represents a different initial value of z, while c is fixed. Each value of c returns a different fractal. Such images demonstrate consistency in the types of structures seen throughout the fractal. Likewise, in Figure 3, each pixel represents a different initial value of z and α is fixed. In contrast, using a Mandelbrot-type of rendering (each pixel represents a different value of α and the initial value of *z* is fixed) displays a variety of structures in the same view. Figure 4 shows three types of spirals from the same α region.



(a) (b) (c)

Figure 3: *adjusted Newton's method applied to $f(z) = z^4 - 1$, rendered by angle of last iteration: (a) $\alpha = 1.470 + 0.899i$, (b) $\alpha = 5.756$, (c) $\alpha = 4.877 + 2.739i$.*



(a) (b) (c)

Figure 4: *types of spirals in adjusted Newton's method fractal for $f(z) = z^4 - 1$ and $\alpha \sim 1.5 + 0.9i$, rendered by angle of last iteration: (a) loose three-armed, (b) tight three-armed, (c) six-armed.*

Systems of Real Equations

Newton's method generally works well with complex numbers, because complex-valued equations involving polynomials and transcendental functions usually have solutions. But real-valued equations may not, for example, $x^2 + 1 = 0$ or $\sin(y) - 3 = 0$. In cases like this, the method will bounce around chaotically. One way to exploit this is to use two real equations, solving $f(x) = 0$ for the *x*-component of the pixel, and $g(y) = 0$ for the *y*-component. Let *z* be initialized as the complex pixel coordinate. Then, for each iteration,

- Let *x* be the real part of *z* and *y* be the imaginary part of *z*.
- Use Newton's method (with or without an adjustment factor) on $f(x) = 0$ and $g(y) = 0$, independently.
- Recombine *x* and *y* into *z*; $z = x + iy$.
- Manipulate *z*, for example, multiplying *z* by a complex constant.

I introduced the last step to force interactions between *x* and *y*. Otherwise, the image will be that of the dynamics of two independent variables, which may not be interesting. Figure 5 shows three examples, each solving $f(x) = x^2 + 1 = 0$ and $g(y) = \sin(y) + 3 = 0$. In Figure 5(a), there's no manipulation of the complex variable *z* after the new real *x* and *y* values are combined. In Figure 5(b), *z* is multiplied by $2 - i$ after each iteration, and Figure 5(c) shows the effect of taking the reciprocal of *z* after combining. As before, each image is rendering according to the polar angle of *z* at the last (32nd) iteration, since the routine did not converge to a solution in any of the three cases. No adjustment factor was used.

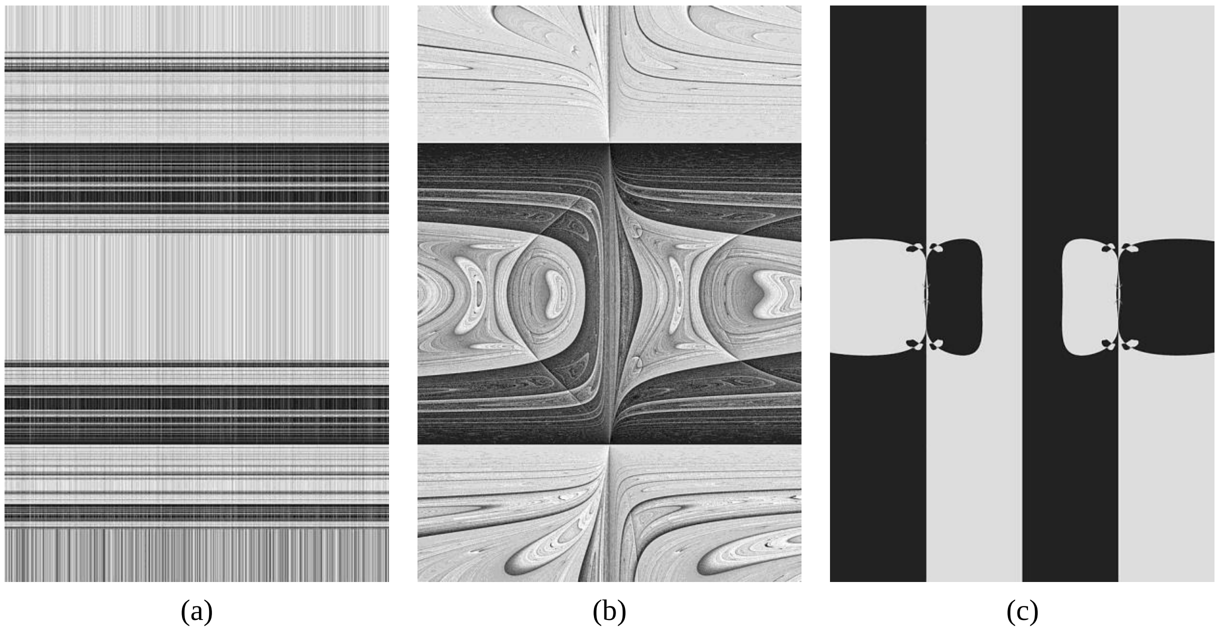


Figure 5: *examples of effects of different ways to manipulate z after independent x- and y-component Newton iterations: (a) none, (b) multiply z by 2 – i, (c) take the reciprocal of z.*

Another way to use multiple real equations is to have the algorithm attempt to solve them simultaneously. That is, solve two different real-valued equations $f(x,y) = 0$ and $g(x,y) = 0$. This requires modifying the iterative equation to a matrix form [7]:

$$\begin{bmatrix} x_{new} \\ y_{new} \end{bmatrix} = \begin{bmatrix} x_{old} \\ y_{old} \end{bmatrix} - \begin{bmatrix} a_{fx} & a_{gx} \\ a_{fy} & a_{gy} \end{bmatrix} \begin{bmatrix} \dfrac{\partial f}{\partial x} & \dfrac{\partial f}{\partial y} \\ \dfrac{\partial g}{\partial x} & \dfrac{\partial g}{\partial y} \end{bmatrix}^{-1} \begin{bmatrix} f(x_{old}, y_{old}) \\ g(x_{old}, y_{old}) \end{bmatrix}.$$

Here, a_{fx}, a_{gx}, a_{fy}, and a_{gy} are the four adjustment factors, affecting: *x* in the *f* equation, *x* in the *g* equation, *y* in the *f* equation and *y* in the *g* equation, respectively. This matrix is the analog of the adjustment factor α. In the standard case, it's simply the 2×2 identity matrix. Clearly, this is a more involved approach, but the extra parameters (four adjustment factors) add more avenues for exploration. Figure 6 shows two examples. My image, "Sand Storm," is in Figure 6(a), showing the adjusted method working on the system: $f(x,y) = x^2 + y^2 - c = 0$ and $g(x,y) = x^2 - y^2 + d = 0$, where *c* and *d* are the horizontal and vertical pixel coordinates, respectively. In this view, no solution is found. Figure 6(b) shows the non-adjusted ($a_{fx} = 1$, $a_{fy} = 0$, $a_{gx} = 0$, and $a_{gy} = 1$) method working on two non-intersecting hyperbolas, one horizontal and one vertical. Both images are rendered according to the angle of $x + iy$.

This method is not limited to two equations in two variables and can be readily extended to systems of three equations in three unknowns, or to higher-order systems. With three variables plus formula parameters and the nine elements of the 3 x 3 adjustment matrix, the possibilities for creating images are greatly expanded. Two examples are presented in Figure 7. Figure 7(a) considers the simultaneous solutions of the system: $x + y^2 + z^3 = 32$, $x^3 + y + z^2 = 12$, and $x^2 + y^3 + z = 12$, which has a solution of $x = 1$, $y = 2$, and $z = 3$. This image is rendered by the number of iterations required for the unadjusted method to find a solution, for x and y given by the pixel coordinates around (1, 2) and $z = 3$. In Figure 7(b), the effects of the adjustment matrix elements are illustrated, with the pixel coordinates determining two of them, while the others are fixed. The three equations represent the surfaces of three non-intersecting hyperboloids and the pixel shades relate to the final angle of $x + iy$.

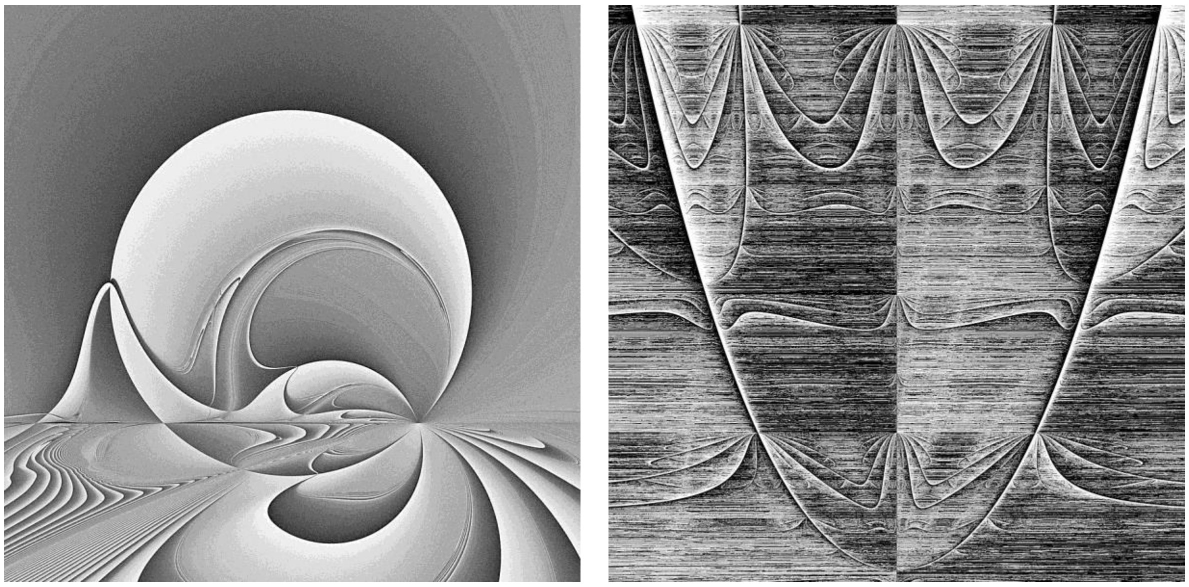
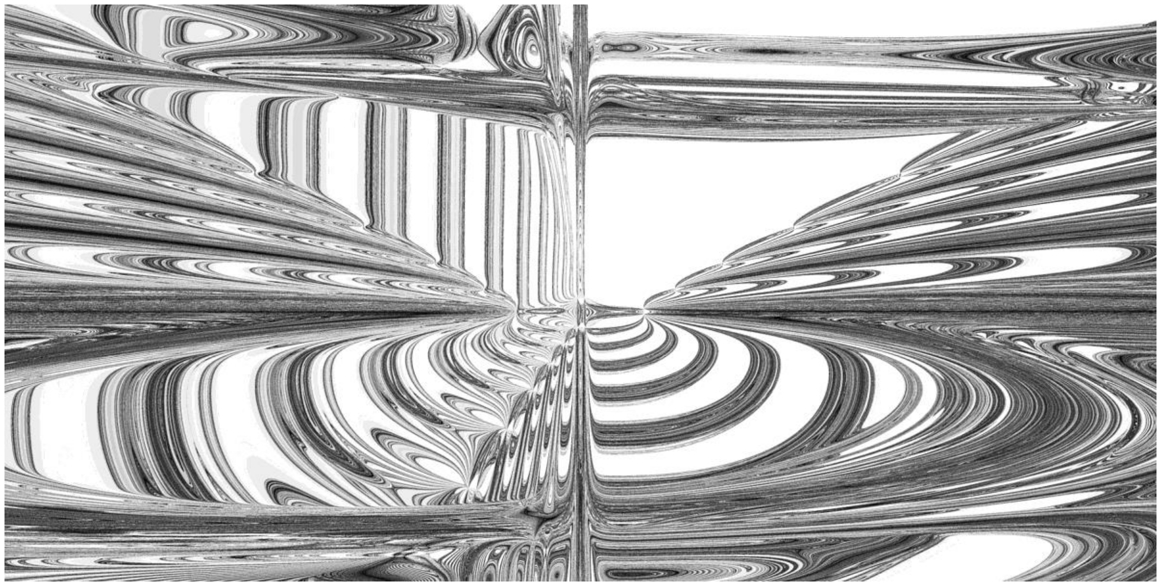
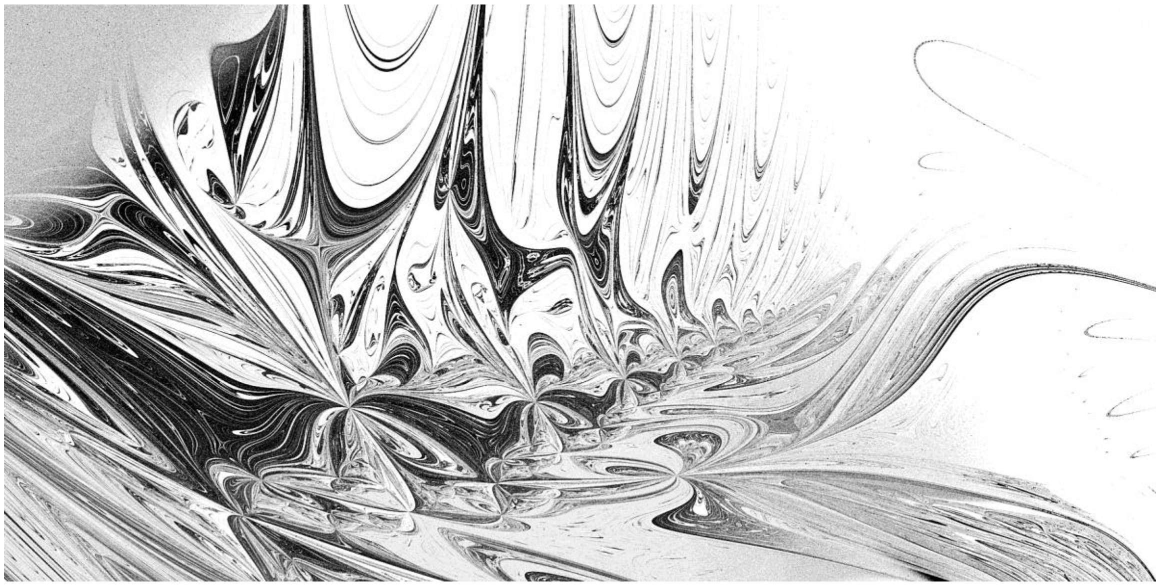

(a) (b)

Figure 6: *examples of Newton's method with a 2 × 2 system of equations to find intersections of two curves: (a) an ellipse and a hyperbola, (b) two hyperbolas.*



(a)



(b)

Figure 7: *examples of Newton's method with a 3 × 3 system to find intersections of three surfaces: (a) $x + y^2 + z^3 = 32$, $x^3 + y + z^2 = 12$, and $x^2 + y^3 + z = 12$; (b) three non-intersecting hyperboloids*.

Even without changing Newton's method, things can be done using it to create interesting images. Here, we look at two more techniques, involving changing the equations to be solved during the process of iterating. Figure 8 shows two cases of solving $z^3 – c = 0$. The standard Newton fractal, with $c = 1$, resembles Figure 1(a), with three rays, instead of four. For Figure 8(a), c began at 1 ($1 + 0i$), and was then rotated 85 degrees counterclockwise every iteration. Changing c every iteration in effect requires the method to find a cube root of a different number every time, so there's no single solution on which to converge. However, some three-fold structure is visible in the polar angle of z. In Figure 8(b), the rotation has been increased to 87 degrees. The overall structure is still visible, but now has been augmented by very fine fractal details suggesting an oscillation with a period of 31 iterations. When the rotation is increased to 89 degrees (not shown), all structure is lost in a sea of visual noise.

More extreme than altering a single parameter every iteration is to change the entire equation every iteration. For example, consider alternating between solving $f(z) = z^3 – 1$ and $f(z) = z^9 – 1$. All three solutions of the former equation are also solutions to the latter, so the standard Newton's method can be expected to find a solution. Figure 9(a) shows this to be the case, in that the image is rendered according to how many iterations were required for convergence. However, the influence of the second equation shows up in additional structure, which hints at the ninth-order nature. In Figure 9(b), the method alternates between three equations, $z^2 – 1 = 0$, $z^3 – 1 = 0$, and $z^5 – 1 = 0$. All three have the solution $z = 1$, but the path getting to that solution is somewhat torturous, as suggested by the fractal structure.

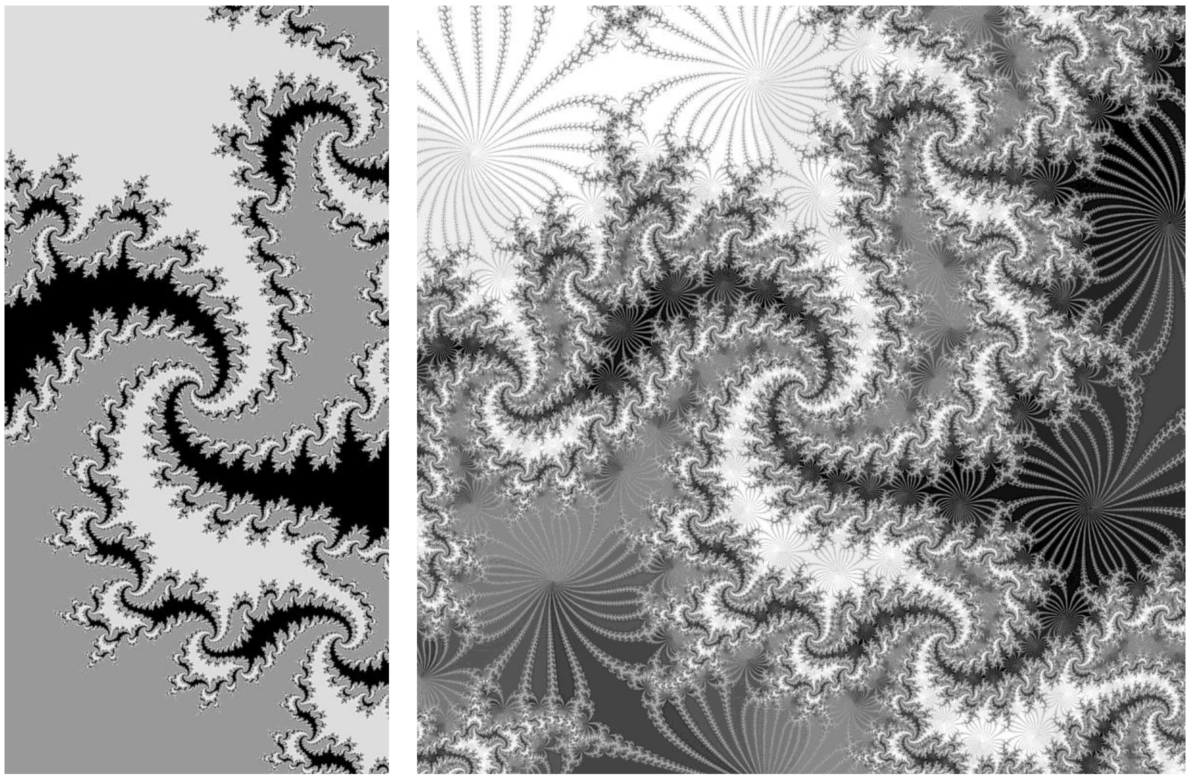

(a) (b)

Figure 8: *rotating c each iteration in solving $z^3 + c = 0$: (a) 85°, (b) 87°.*

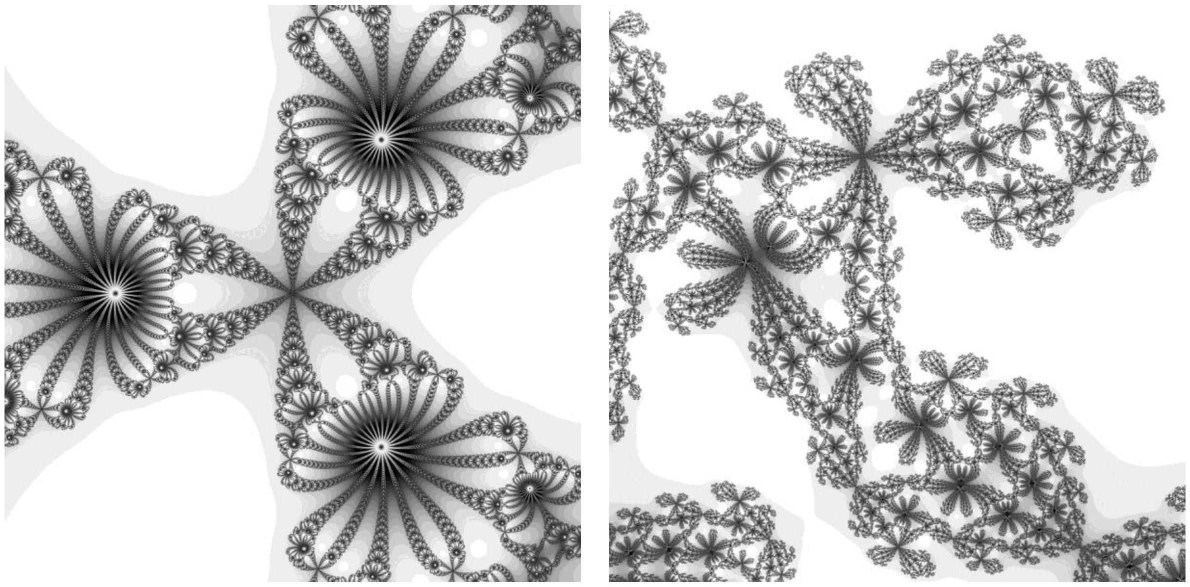
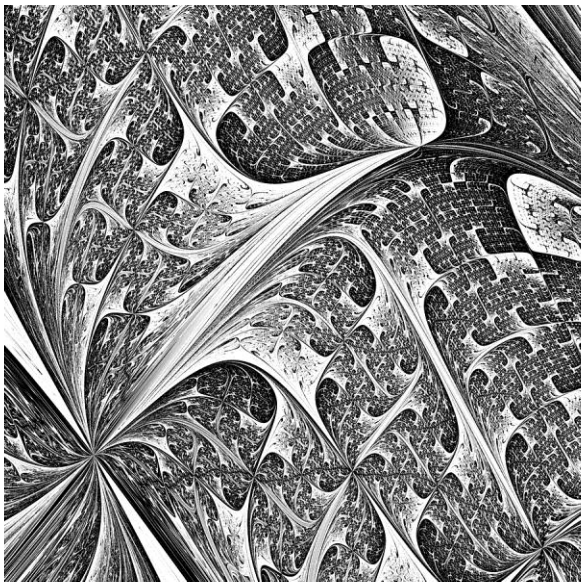

(a) (b)

Figure 9: *alternating equations to be solved every iteration: (a) $z^3 - 1 = 0$ and $z^9 - 1 = 0$, (b) $z^2 - 1 = 0$, $z^3 - 1 = 0$, $z^5 - 1 = 0$.*

Summary and Conclusions

The simplicity of Newton's method makes it easy to expand upon and this paper illustrates several possible variations. When the goal is to enhance the method's chaotic dynamics, as opposed to improving its solution-finding properties, then many changes can be made to the routine itself and how it is employed, to prevent a solution from being found. Then, the orbit of the iterates becomes a rich field for visual exploration. Two additional examples are shown in Figure 10. Figure 10(a) uses a pair of equations, $x^2 + 1 = 0$ (which uses an adjustment factor of 0.9) and $y^2 + 1 = 0$ ($\alpha = 1.1$). The real-valued x and y are combined into z, which is multiplied by the pixel coordinates each iteration. In Figure 10(b), the equation to be solved in $z^3 - c = 0$, where c is the complex pixel coordinates. The iterate and the adjustment factor are both multiplied each iteration, the iterate by $1 + i$ and the adjustment factor by $0.893 + 0.125i$. In both cases, the grayscale levels are determined from the polar angles of z.


(a)

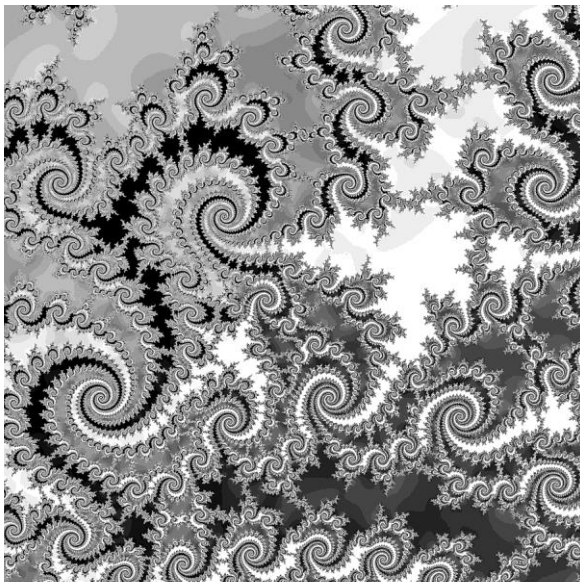

(b)

Figure 10: *two additional examples: (a) two real equations and multiplying the iterate each iteration, (b) $z^3 - c = 0$ and multiplying both the iterate and adjustment factor each iteration*.

References

[1] Weisstein, Eric W. "Newton's Method." From MathWorld--A Wolfram Web Resource. http://mathworld.wolfram.com/NewtonsMethod.html.

[2] K. R. Meadows, J. Casper, and D. A. Caughey. "A Numerical Investigation of Sound Amplification by a Shock Wave." *AMSE Publications: FED*, 147, p. 47.

[3] P. Bourke. "Gallery of Fractals Created using the Newton Raphson Method." http://paulbourke.net/fractals/newtonraphson/.

[4] Kalantari, B. Polynomial Root-Finding and Polynomiography. World Scientific, 2009.

[5] K. Mitchell. "Newton's Tracery." http://kerrymitchellart.com/gallery56/newtons-tracery.html.

[6] P. Bourke. "Julia Set Fractal (2D)." http://paulbourke.net/fractals/juliaset/.

[7] "Newton's Method." https://en.wikipedia.org/wiki/Newton%27s_method#Nonlinear_systems_of_equations.