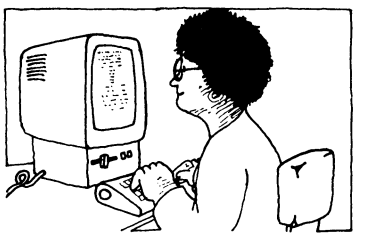
COMPUTER CORNER

EDITOR

Eugene A. Herman
Department of Mathematics
Grinnell College
Grinnell, IA 50112



In this column, readers are encouraged to share their expertise and experiences with computers as they relate to college-level mathematics. Articles that illustrate how computers can be used to enhance pedagogy, solve problems, and model real-life situations are especially welcome.

Classroom Computer Capsules features new examples of using the computer to enhance teaching. These short articles demonstrate the use of readily available computing resources to present or elucidate familiar topics in ways that can have an immediate and beneficial effect in the classroom.

Send submissions for both columns to Eugene A. Herman.

The Root-Finding Route to Chaos

Richard Parris

Richard Parris received his mathematics degrees from Tufts University (B.A.) and Princeton University (Ph.D.). He now lives at Phillips Exeter Academy in New Hampshire, where he teaches mathematics, coaches, and supervises a dormitory. If he had spare time, it would be spent at the harpsichord, not at the computer.

The current excitement over fractal images and the dynamical systems that produce them is a boon for mathematics teachers. This enticing subject is so accessible that we can not help but feel that our students will finally find the inspiration that has heretofore been lacking in their approach to mathematics.

The central theme of dynamical systems is the iteration of functions. One studies the mathematics of feedback loops, which incessantly turn output back into input. Exploration would therefore seem to be a simple matter: Select a favorite function and a seed value, then iterate and wait for rich patterns to appear.

Alas, it is not that simple. For one thing, *nothing* might happen. To see extraordinary sights, after all, one usually has to know where to look. For another,

those obnoxious questions may still arise: *What is the purpose? Is this stuff really good for anything? Why iterate functions in the first place?* Indeed, *is* there any more to the mathematics than just the production of complicated images? Even a teacher who believes that this is reason enough to be interested cannot expect very many students to commit themselves to a project whose purpose may seem to be just another attempt to trick them into doing some mathematics.

It is fortunate that the opportunity presented by this new subject does not reside solely (and superficially) in the beauty and complexity of the graphics that now flood the marketplace. Rather, it is that these fascinating patterns are locked up in the simplest of mathematical formulas. One can be learning the basics as one explores this territory, which, although breathtakingly new, lies remarkably close to home. This will encourage those teachers who are wary of any attempts to squeeze more new applications into an already crowded curriculum. Instead of displacing traditional topics, the study of dynamical systems gives us new ways to develop fundamental concepts, strengthening the traditional curriculum while enriching it. Moreover, the novelty will breathe life into our dry, dusty subject. Most significant, perhaps, is the abundance of easily asked, open-ended questions. Our budding scientists can go as far with the material as they want. A few of them may even feel that they have a chance to make original contributions.

This article, which is organized around a single, well-known algorithm for root extraction, presents an effective way of incorporating dynamical systems into the teaching of mathematics. As the sample exercises show, students have the opportunity to do some purposeful analysis on this algorithm. The opening item might be posed as a follow-up to a student-initiated question about how calculating machines obtain their answers:

1. Calculate the positive square root of 5, accurate to (say) eight decimal places, using only the operations of elementary arithmetic.

Unless someone in the class already knows the Babylonian [**1**] divide-and-average process $x \to R(x) = (x + 5/x)/2$, the teacher may have to orchestrate its rediscovery. At the very least, the class should learn how this venerable and marvelous method works; it is rich in mathematical themes. The target number is trapped between two known values, x and $5/x$, where x initially is a guess. The target is, in fact, the geometric mean of these values. The arithmetic mean beckons as an easily calculated and improved approximation to the target; one therefore replaces x by $R(x)$, hence $5/x$ by $5/R(x)$, too. In this way, x and $5/x$ are brought closer together (see item 3 below); iteration *ad infinitum* brings x and $5/x$ arbitrarily close together. The sequence $\{x_n\}$ defined recursively by $x_{n+1} = R(x_n)$ must therefore converge to the desired square root, given any positive x_1. As will be seen, the rate of convergence is astonishingly rapid.

2. Prove that $\sqrt{5}$ lies between x and $5/x$, for any positive number x.

3. Given that y is between x and $5/x$, show that $5/y$ is, too.

At an elementary level, this bracketing theme can serve as an introduction to our project. I present such a class with an extensive table of pairs (x, y) of positive solutions to $xy = 5$, then ask for the number that separates every such pair.

The student also has opportunities to write meaningful computer programs to test a growing theory. I find it expedient to begin computer work by giving the class a program fragment with which everyone can begin to experiment. As the material

evolves, so does the program. In pseudocode:

```
input b                    { the number whose root we seek }
input x                    { a seed value, or initial guess }
for k := 1 to 20
    x := (x + b/x)/2
    print x
```

This program is a simple example of a *dynamical system*. It was designed to seek out solutions to a specified problem; we need only provide a *seed value* to initiate the process each time.

4. Does the outcome of a root-finding search depend on which seed value we feed the program? Are there seed values that do not lead to a root? [Except for zero, each seed value leads to the nearer root.]

5. Study the rate of convergence of the square-root process. In particular, how many steps are necessary to obtain 1000-place accuracy for $\sqrt{5}$? Does the 20-step program above suffice, assuming that the computer would print out 1000 places if we wanted to see them? [Yes; ten steps would suffice.]

For item 5, it is desirable for the software to deliver as many decimal places as it is capable of. The quadratic rate of convergence is then easy to see. Roughly put, the number of correct significant digits doubles with each iteration, once x is sufficiently close to the target. Here is why: Let m be the positive square root of b, so that $2(R(x) - m) = (x - m)^2/x$. If $m < x$ (which will be true after just one step of the process, if the initial approximation is positive), then $(R(x) - m)/(x - m) < 1/2$ shows that the error shrinks to zero. Once $x - m$ drops below 1, $R(x) - m < (\frac{1}{2}m)(x - m)^2$ ensures the accuracy-doubling.

The Newton-Raphson method $x \to R(x) = [(n-1)x + b/x^{n-1}]/n$ extends the Babylonian method to the problem of calculating n^{th} roots of b. The cube-root case will be of interest to us below. Computer-generated data reveal the same accuracy-doubling phenomenon, and it can be proved as above. Incidentally, it is possible to devise algorithms, such as Hutton's [2], that deliver even greater rates of convergence.

Complex Numbers

It is well known that the real number system is inadequate when it comes to providing solutions to equations. We must look within the larger system of imaginaries to find everything we need. These nonreal quantities have been invisibly exerting their influence all along.

6. Consider the square-root-finding program again. What happens when we provide a negative value for b (and seed the process with a nonzero real x-value)?

The computer provides a chaotic data stream for us to puzzle over. It is easy to dismiss the example by simply observing that negative numbers do not have real square roots (thus there is nothing to which the process *can* converge), but this is only a partial explanation, and it would miss a thematic point of this investigation. The time has come to include imaginary numbers in our discussion. Depending on

the resident computer language, the following transition may not be necessary:

7. Modify the program that finds square roots so that it will be able to solve nonreal problems as well. This means that it should be able to find square roots for $3 + 4i$, for example. [$\pm(2 + i)$]

Because every complex number is a *pair* of real numbers, we now have twice as many variables. It is worth emphasizing, however, that the function being iterated is still the familiar $R(z) = (z + b/z)/2$, where z stands for $x + yi$ and b stands for $c + di$. The next exercises are essential:

8. Apply the square-root finder to a few simple examples, including some for which you already know the two roots. In particular, what happens when we ask for a square root of -1 and seed the process with a nonreal value?

9. Different seed values can produce different results when they are fed to a root-finding program. Explore this phenomenon for the square-root finder above. Let us consider, for example, a search for the two square roots of $3 + 4i$, using (in turn) the values $z = 1$, $z = -1 + 3i$, $z = 2 - 5i$, and $z = -2 + 4i$ to seed the process.

10. Given a square-root-finding program and a choice of seed value, we anticipate three possible outcomes: the process will converge to one of the two roots, or else it will do neither, instead wandering aimlessly forever. Try to discover a simple rule that predicts accurately which of the three outcomes will occur. Can you prove that your rule will work correctly every time? [The final question is difficult; see item 14 and the ensuing discussion.]

Exercise 10 makes a good classroom activity. Every student can participate in color-coding the complex plane. Each seed value receives its color according to the destination of its orbit—the sequence of values produced by the process. If nothing else, this activity will afford some practice plotting complex numbers; the real goal, however, is the discovery of the square-root rule: *Seed values are attracted to the nearer root, unless there is no nearer root. Those seed values that lie on the perpendicular bisector of the segment joining the roots do not lead anywhere.* This principle explains the behavior of the real square-root finder when it is given the job of finding the square roots of a negative real number (see item 6). It also explains the chaotic results when $-2 + 4i$ is used to seed the search for a square root of $3 + 4i$ (see item 9; see item 12 also, however).

11. Suppose that $m^2 = b$, and that z is nonzero and equidistant from m and $-m$. Prove that $R(z) = (z + b/z)/2$ is also equidistant from m and $-m$. [The hypothesis is $|z - m| = |z + m|$, which implies that $|z - m|^2 = |z + m|^2$. In turn, this may be rewritten in the form $|z^2 - 2mz + b| = |z^2 + 2mz + b|$. Now divide by $2|z|$.]

12. If, while searching for a square root of $3 + 4i$, the computer is allowed to calculate too many terms of the orbit of $-2 + 4i$, it is quite possible that the resulting sequence will approach one of the roots after all. Does this contradict the result of the preceding problem? [Roundoff errors cause the orbit to drift away from the true line of chaos.]

13. Although we still use the Babylonian formula when we search for imaginary square roots, it is no longer correct to think of the target value

as being trapped between two known values as these values are brought closer together. Use the example $b = 3 + 4i$ and the seed value $z = 1$ to explain this remark.

The preceding exercise casts considerable doubt on the reliability of the complex square-root finder. In other words, it is not at all obvious that the process will converge to whichever root is closer to the seed value. Although this is an elementary result, it is remarkably difficult to establish without some sort of guidance; it may be necessary to leave this theorem unproved. On the other hand, one's intuition says that seed values that are sufficiently close to a target ought to lead quickly to that target, as in the following:

14. If $|z - m| < 2|m|/3$, then $|R(z) - m| < |z - m| < 2|m|/3$ must hold as well. The sequence of z-values approaches m as a limit.

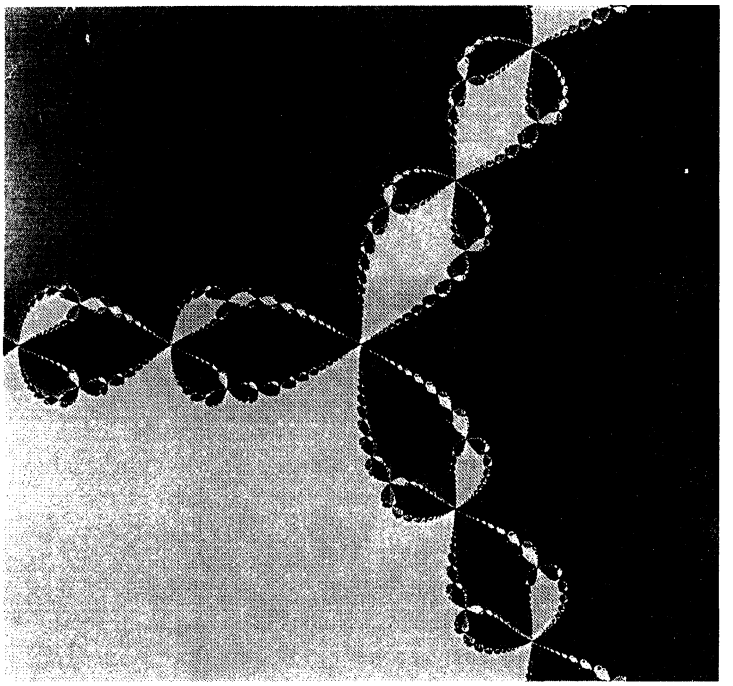
This is proved by first writing $|R(z) - m|$ in the form $|z - m|^2/(2|z|)$ and then noting that $|m|/3 < |z|$ is implied by the triangle inequality and the hypothesis. Both $|z - m|/(2|z|) < 1$ and $|z - m|^2/(2|z|) < |z - m|$ now follow. Establishing convergence requires more careful inequalities: Let $|z - m| = k|m|$, where $k < 2/3$. As above, one deduces from the triangle inequality that $(1 - k)|m| \leq |z|$, from which $|R(z) - m| \leq k|z - m|/(2 - 2k)$ follows. Because $k/(2 - 2k) < 1$, one deduces that $|z - m|$ converges geometrically to zero.

For the sake of completeness, I include the following outline of how one may establish the general result: The most efficient way to proceed is to change variables. The dynamical system $z \to R(z)$ is transformed into a new system by replacing each z by $m(1 + w)/(1 - w)$, where $m^2 = b$ and w is the new variable to watch. The dynamical system that results from this transformation is simply $w \to w^2$! Moreover, the target $z = m$ corresponds to $w = 0$, the dividing line $|z - m| = |z + m|$ corresponds to the unit circle in the w-plane, and the attractive region $|z - m| < |z + m|$ corresponds to the interior of the unit circle. Now, instead of iterating the process $z \to R(z)$ and watching to see whether z approaches m, one iterates the process $w \to w^2$ and watches to see whether w approaches 0. So long as the squaring process starts inside the unit circle, this is certain to happen.

15. The preceding discussion of how the Babylonian square-root process works dynamically has revealed a very simple geometric principle. It is natural, therefore, for us to try to extend this understanding to the cube-root process. Modify the program that finds square roots of complex numbers so that it will find cube roots instead. Try your program on the cube roots of 8.

As before, the class now can engage in a color-coding activity. Seed values are colored according to the behavior of their orbits. Three colors are needed for the three cube roots of 8, and a fourth color reserved for those seed values whose orbits exhibit indecisive behavior. (It is not easy this time to give examples of such indecisive points, however.) A class that has studied the theorem of de Moivre will no doubt expect some sort of three-fold symmetry to be present in the finished diagram, and even those students who have had no experience extracting roots via polar coordinates will recognize the equiangular placement of $z = 2$, $z = -1 + i\sqrt{3}$, and $z = -1 - i\sqrt{3}$. Someone, remembering that (almost) every real seed value leads to the real cube root, will draw the (almost) correct conclusion that the real axis should receive only one color. A few students may be willing to conjecture what the correct arrangement of colors is, but it is unlikely

that anyone will anticipate the staggering detail:



Seed values for the $\sqrt[3]{8}$ algorithm are colored according to the behavior of their orbits. The black seeds converge to 2.

The indecisive points form the complicated boundary that separates one color from another. One of them is the origin, which is the central point in the figure. As we have seen on occasion, the behavior of these boundary points during the root-finding process is just as confusing as is their appearance. What makes the boundary configuration so complicated visually is that *each boundary point borders on all three colors*. Unlike the square-root diagram, this figure does require a computer program! In pseudocode:

```
for each pixel P
  begin
  Z := user_coordinates(P)          {seed root search}
  color := background
  count := 0
  repeat
    Z := (2 * Z + 8/(Z * Z))/3
    count := count + 1
    if |Z - root1| < error then color := color1 else
      if |Z - root2| < error then color := color2 else
        if |Z - root3| < error then color := color3
  until (color <> background) or (enough < count)
  apply color to P
  end
```

Convergence is assured once the orbit of Z wanders within *error* of *rooti*, at which point calculation for pixel P halts and *colori* is assigned. However, it is impractical to wait for this outcome if an astronomical number of iterations is required, or if the orbit has no limit at all. Thus calculations must be halted once more than *enough* iterations have been tried. See the Appendix for a BASIC version of this program.

Root-Finding Wrap-Up

This startling picture is an unexpected conclusion to our investigation of the root-finding process. What makes this process even more interesting is its history, for this represents perhaps the first well-studied problem whose solutions form what is now called a *fractal*. In 1879, the British mathematician Arthur Cayley proposed to study the dynamical behavior of complex numbers when the Newton-Raphson cube-root method is applied to them [**3**]. In effect, his goal was to understand and describe the pattern we have just seen. Having already disposed easily of square roots, Cayley may have felt confident that he would be successful with this problem; however, he never did publish any results. It was not until some forty years later that two French mathematicians, Pierre Fatou [**4**] and Gaston Julia [**5**], were able to resolve Cayley's problem. It is sad that neither man lived long enough to see these images. Julia even lived until 1978. (In his honor, the chaotic sets of indecisive points that typically arise in dynamical systems are now called *Julia sets*.) The recent advances in computer graphics, which make such images possible, have thereby reawakened interest in what is an old subject.

What sort of mathematics course accommodates such exploration? On several occasions, I have introduced this material during a ten-week course (forty meetings), which has the following syllabus: vectors, polar coordinates and complex numbers (including de Moivre's theorem, of course), a review of graphing techniques, elementary sequences and series, and an introduction to limits. This course is the last one in our curriculum before the onset of calculus. The teacher can therefore draw on all of precalculus mathematics. In particular, one can approach the cube-roots-of-8 problem in at least three *fundamentally different* ways: via factoring and solving a cubic equation, via trigonometry and the theorem of de Moivre, and—lastly—via the dynamic method of iteration.

It is worth emphasizing that much good mathematics is done before production of the final picture. Among other things, this short study of dynamical systems provides meaningful examples of sequences, whereby certain irrational quantities are exhibited as limits of rationals. Students can have hands-on experience in the devising, coding, and testing of algorithms. Moreover, these dynamic algorithms stand in marked contrast to traditional methods of solving mathematical problems.

Appendix

```
100 screen 1 : color 0,1 : cls : hpix = 319 : vpix = 199
110 left = -2 : right = 3 : low = -2 : high = 2
120 window (left, high) - (right, low)
130 hdel = (right-left)/hpix : vdel = (high-low)/vpix
200 for j = 0 to hpix
205    p = left + j * hdel
210    for i = 0 to vpix
215       q = low + i * vdel
```

```
220      x = p : y = q : col = 0
300      for k = 1 to 20
310        denom = (x * x + y * y)*(x * x + y * y)
312        if denom < 0.000001 then 400
320        newx = (2* x  + 8* x * x - y * y)/denom)/3
322        newy = (2* y - 16* x * y/denom)/3
330        x = newx : y = newy
340        if 0.5 < (x - 2)*(x - 2) + y * y then 350
345          col = 1 : goto 400
350        if 0.5 < (x + 1)*(x + 1) + (y - 1.73)*(y - 1.73) then 360
355          col = 2 : goto 400
360        if 0.5 < (x + 1)*(x + 1) + (y + 1.73)*(y + 1.73) then 370
365          col = 3 : goto 400
370      next k
400      pset(p, q), col
500    next i
510  next j
600  end
```

Lines 200 and 210 instruct the computer to examine every pixel on the screen, which, according to the minimal CGA specifications of line 100, measures 320 horizontally by 200 vertically. Lines 110 and 120 associate these pixels with the region $-2 \le x \le 3$, $-2 \le y \le 2$. This viewing window contains all three roots with a little room to spare. At any instant, the pixel being examined represents the point (p, q). Initially, the pixel is assigned the color black (color 0); this assignment is changed to a different color if the orbit of (p, q) wanders close enough to one of the three target points within twenty iterations (line 300). The active point on the orbit is named (x, y), although it is temporarily named (newx, newy) during the calculation stage. Color decisions are made in lines 340 through 365, and the actual coloring of the pixel takes place at line 400.

References

1. Dirk Struik, *A Concise History of Mathematics*, Vol. I, Dover, 1948, p. 29.
2. D. F. Bailey, A historical survey of solution by functional iteration, *Mathematics Magazine* 62 (1989) 159.
3. Arthur Cayley, The Newton-Fourier imaginary problem, *American Journal of Mathematics* 2 (1879) 97.
4. Pierre Fatou, Sur les équations fonctionelles, *Bulletin de la Société Mathématique Francaise* 47 (1919) 161–271; 48 (1920) 33–94, 208–314.
5. Gaston Julia, Sur l'iteration des fonctions rationelles, *Journal de Mathématiques Pures et Appliquées* 8 (1918) 47–245.