

Iteration and Fractals on the TI-82 Calculator

Thomas W. Shilgalis 4520 Mathematics Department
 Illinois State University
 Normal, IL 61790-4520 Phone (309) 438-7302

Initial exercises

1. Set the calculator to radian MODE.
 - (i) Press COS, then any number of your choice, then ENTER.
 - (ii) Now press COS, then ANS, then ENTER. (The *last entry* key, ENTRY, can be used instead; replace the number by ANS.)
 - (iii) Repeat (ii) three or four times. What do you observe?
 - (iv) Press ENTER another twenty or so times. What happens?

2.
 - (i) Enter the expression $1+(1+4)$ and press ENTER.
 - (ii) Press ENTRY and replace 4 by ANS.
 - (iii) Press ENTER several more times. What do you observe?

3.
 - (i) Enter the expression $(10 + 36\div 10)\div 2$ and press ENTER.
 - (ii) Press ENTRY and replace 10 by ANS.
 - (iii) Press ENTER several more times. What do you observe?

You have been solving (approximately) the equation $f(x) = x$ by *iteration* (or *feedback*), a process in which each output value is used as the next input value. In example 1 $f(x) = \cos(x)$, while in examples 2 and 3 $f(x) = 1/(1+x)$ and $f(x) = (x + 36/x)/2$, respectively. The numbers 10 in example 3, 4 in example 2, and "any number of your choice" in example 1, step (i), are called *initial values* or *seeds*. (You may have recognized example 3 as the familiar "divide and average" method of approximating square roots.) A value x^* which satisfies $f(x^*) = x^*$ is called a *fixed point* of f . Some fixed points are *stable* (or *attractors*), while others are *unstable* (or *repellers*).

More iterating

4. Now iterate the function f defined by $f(x) = \sqrt{1+x}$. What is the solution to $f(x) = x$?

In examples 2, 3 and 4 it is possible to solve the equation $f(x) = x$ explicitly by ordinary algebra; not so with example 1, however.

5. Solve $f(x) = x$ by iteration when $f(x) = e^x - 2$. There are two solutions, one positive and one negative. Only one solution can be found by iterating f , and your choice of an initial value is important. To find the other solution, iterate the function g defined by $g(x) = \ln(x+2)$. (Recall that when $x > -2$ the equation $\ln(x+2) = x$ is equivalent to $e^x - 2 = x$.) The two solutions are $-1.84140566\dots$ and $1.146193221\dots$. The former is an attractor for f and a repeller for g ; the latter is an attractor for g and a repeller for f . Verify this by using -1.841 as an initial value when iterating g and by using 1.146 as an initial value when iterating f .
6. Now try to solve $2 \sin(x) = x$ by iteration. The solutions are 0 (unstable) and $\pm 1.89549426\dots$ (both stable). Can you get them all?
7. Now try to solve $\sin(x) = x$ by iteration. The stable solution 0 is approached very slowly.
8. Now try to solve $\cos^{-1}(x) = x$. Are you having trouble? Perhaps you chose a bad initial value. Try $.7390851332$, which was the solution to example 1. Be patient and press ENTER several times. Now try $.739$. The solution to $\cos^{-1}(x) = x$ certainly exists, but it is unstable.
9. The equation $x^2 = 2^x$ (which is a special case of $x^y = y^x$) has the obvious solutions 2 and 4 and a non-obvious solution $-.766\dots$. One way to solve this equation is to find a function to iterate by "solving for x ." Taking logs of both sides we get $\ln(x^2) = x \ln 2$, from which $x = \ln(x^2)/\ln 2$. Iterate $f(x) = \ln(x^2)/\ln 2$, trying several different initial values. What do you observe? See what happens if you iterate the nearly equivalent function $g(x) = (2 \ln x)/\ln 2$.

A different way to "solve for x " produces $x = \pm\sqrt{2^x}$. Iterate $r(x) = \sqrt{2^x}$. Then iterate $s(x) = -\sqrt{2^x}$. Make note of your results.
10. Iterate $f(x) = 4x(1-x)$. What do you observe? Try the exact solution $.75$ as your initial value. Then try $.750000001$ and $.749999999$. Is the solution $.75$ stable?

Example 10 is a special case of the *quadratic iterator* $f(x) = ax(1-x)$. When $1 \leq a \leq 4$ and $0 \leq x \leq 1$ the function values are confined to the interval $[0,1]$, so the feedback process does not produce values that "get away" to infinity. Since we want to examine the behavior of this family of quadratic functions for several values of a , it will pay us to enter the following programs in the calculator:

<pre> Prog: QUADSLOW :Disp "ENTER A" :Input A :Disp "ENTER X" :Input X :Lbl 1 :AX(1-X)→X :Disp X :Pause :Goto 1 </pre>	<pre> Prog: QUADFAST :Disp "ENTER A" :Input A :Disp "ENTER X" :Input X :0→C :Lbl 1 :AX(1-X)→X :C+1→C :If C < 100: Goto 1 :Disp X :Pause :Goto 1 </pre>
--	---

Run the program QUADSLOW with one or two of the values of A given in the table. Then run QUADFAST on the others. Record your observations.

A Initial value Long-term output

2		
2.75		
3		
3.01		
3.3		
3.5		
3.55		
3.84		

In your table you should notice a single fixed point for values of A between 1 and 3. But after 3 a phenomenon called *bifurcation* and period-doubling begins and continues as A increases toward the so-called Feigenbaum point s_∞ ^a 3.5699456.... See Figure 11.5 below ([4], p. 104). Compare it with the Feigenbaum diagram on the next page. The first bifurcation value is $b_1 = 3$. The next few are b_2 ^a 3.449489..., b_3 ^a 3.544090..., b_4 ^a 3.564407..., b_5 ^a 3.568759... ([2], p. 224). You should have observed the following 8-cycle with $A = 3.55$: .50603..., .88737..., .35480..., .81265..., .54047..., .88168..., .37032..., .82780.... With $A = 3.84$, however, you should get a 3-cycle: .14940..., .48800..., .95944....

The portion of the final-state diagram to the left of the Feigenbaum point s_∞ is a self-similar fractal tree (see figure 11.5). It describes the *period-doubling scenario* of the quadratic iterator, which leads from a very simple and orderly behavior of the dynamics right to the beginning of the chaotic region. Let us try to understand the mechanism lying at the base of its generation and leading to the self-similarity of the tree.

The Period-Doubling Tree

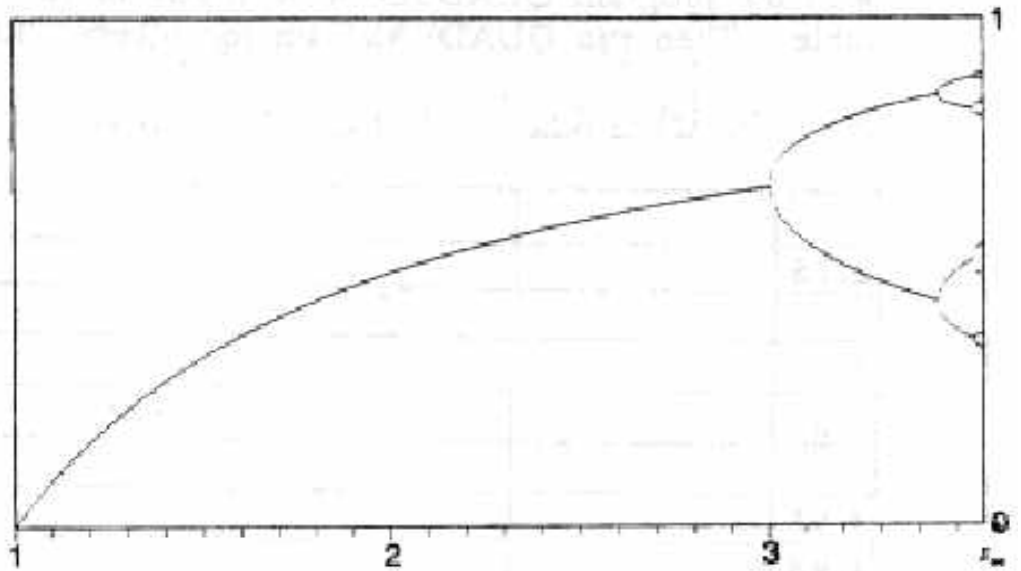


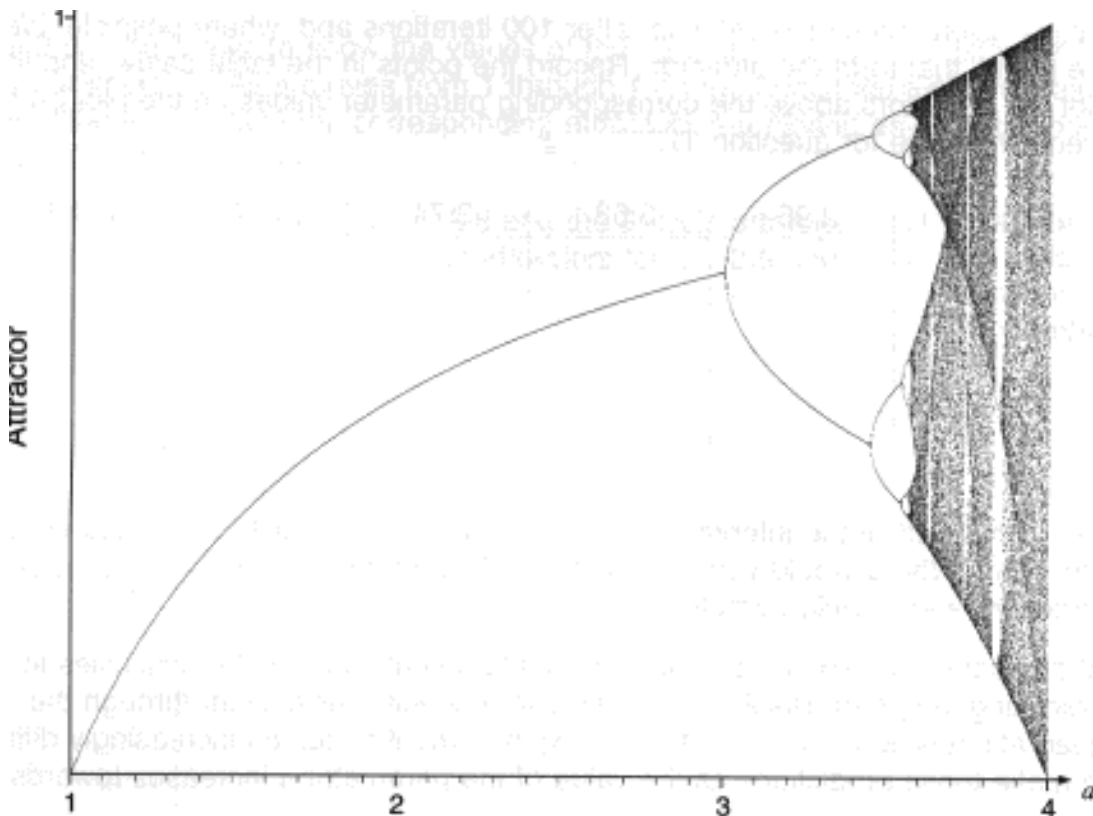
Figure 11.5: The first portion of the final-state diagram - the period-doubling tree.

5.1 3D

The quadratic function $f(x) = ax(1 - x)$ clearly exhibits an extraordinary variety of changing behaviors as the parameter a increases in value from 1 through 4. The most significant connection is with predictability. A stable, predictable long term behavior exists in the form of a fixed point attractor for low values of a . As a increases, predictability becomes more complex as the number of points in the attractors increases. As a approaches 4, the behavior moves quickly into an erratic, chaotic, unpredictable state.

The consequence of this observation is profound and disturbing. Natural phenomena governed by such a quadratic relationship may appear to be in or out of control solely because of the functional relationship among the parameters and variables. Under one set of conditions they can be totally predictable yet, for only slight changes in conditions, they become completely chaotic and unpredictable.

A completed plot of attractors for all functions of the form $f(x) = ax(1 - x)$ for every parameter value from 1 through 4 is shown below. This complete plot is named after the American physicist Mitchell J. Feigenbaum. His work during the mid-1 970's at Los Alamos Laboratory highlighted important properties associated with plots of this type.



Graphical approach to iteration

You have observed that iteration sometimes yields fixed points of a function, i.e., solutions to $f(x) = x$, and that sometimes it fails. Let us see what happens when a function is iterated by looking at the graphs of $y = f(x)$ and $y = x$. The graphs on the next few pages were produced by the software Analyzer*3.2. A TI-82 (and TI-81) program QUADGRAF ([4], p. 91) which does iteration graphically is listed below. In words and symbols, here is the story.

The initial value x_0 is operated on to produce the point $(x_0, f(x_0))$. This point is projected horizontally onto the line $y = x$ to produce the point $(f(x_0), f(x_0))$, or (x_1, x_1) . Evaluation of f at x_1 produces the point $(x_1, f(x_1))$, which is projected horizontally onto the line $y = x$ to produce the point $(f(x_1), f(x_1))$, or (x_2, x_2) . As you can see from the following graphs, the sequence of points sometimes converges to a single point, the intersection of $y = f(x)$ and $y = x$. But sometimes it produces a long-term cycling between two values, or four, or eight, and so on. And sometimes the sequence diverges to $\pm \infty$.

The program QUADGRAF is easy to run, but the TI-82 provides an alternative way to get the same graphical and numerical results. Page 14-9 of the TI-82 manual is reproduced here for convenience. The symbols U_n and U_{n-1} are terms of the U sequence. U_{n-1} is located at the 7 key (2nd function). The equation $U_n = K U_{n-1} (1 - U_{n-1})$ corresponds to the line :AI-AII→J in QUADGRAF, which corresponds to the line :AX(1-X)→X in QUADSLOW and QUADFAST. To iterate a different function, e.g., $\cos(x)$, use the Y= key to replace $U_n = K U_{n-1} (1 - U_{n-1})$ by $U_n = \cos(U_{n-1})$ and change the window appropriately. Try both of

Prog:QUADGRAF ([4], p. 91)

```

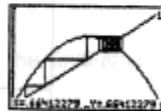
:ClrDraw
:0→R
:0→Xmin      :Lbl 1
:1→Xmax      :AI-AII→J
:0→Ymin      :Line(I,I,I,J)
:1→Ymax      :Lbl 2
:Disp "A="    :Line(I,J,J,J)
:Input A      :Pause
:Disp "I="    :Disp J
:Input I      :Pause
:DrawF AX(1-X) :R+1→R
:DrawF X      :J→I
:AI-AII→J     :If R<8
:Line(I,0,I,J) :Goto 1
:Goto 2       :End

```

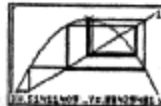
Procedure

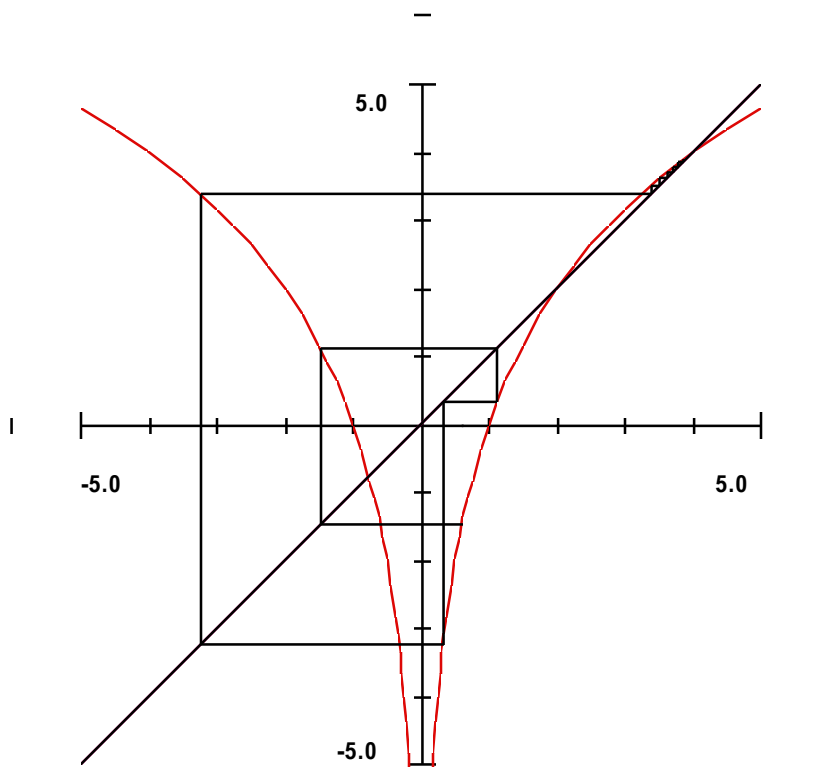
1. Press **2ND** [Y=] Select Seq. Press **WINDOW** [▢] Select Web FORMAT and the defaults. Press **STAT** [STAT PLOT] and turn off all stat plots.
2. Press **Y=**. Enter the sequence. (U_{n-1} is on the keyboard.)
 $U_n = K U_{n-1} (1 - U_{n-1})$
3. Press **2ND** [QUIT] to return to the Home screen and store 2.9 to K.
4. Press **WINDOW**. Set the WINDOW variables.

$U_{nStart} = .01$	$Xmin = 0$	$Ymin = 0$
$VnStart = 0$	$Xmax = 1$	$Ymax = 1$
$nStart = 0$	$Xscl = 1$	$Yscl = 1$
$nMin = 0$		
$nMax = 10$		
5. Press **TRACE** to display the graph, and then press **▢** to trace the cobweb. This is a cobweb with one attractor.

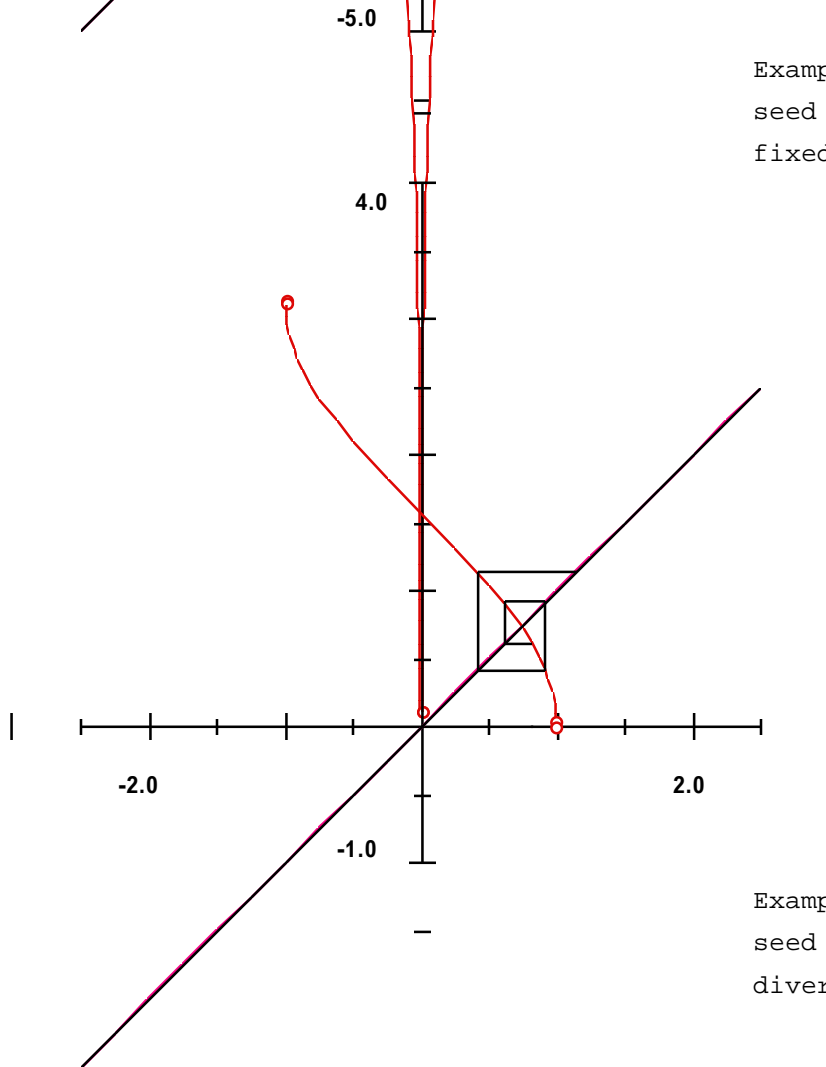


6. Change K to 3.44 and TRACE to show a cobweb with two attractors.
7. Change K to 3.54 and TRACE to show a cobweb with four attractors.

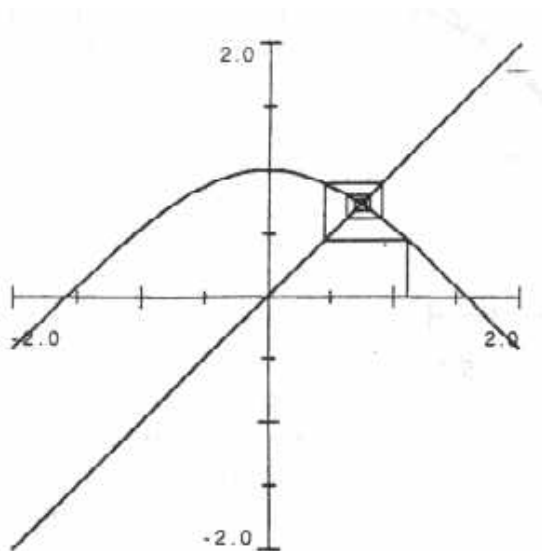




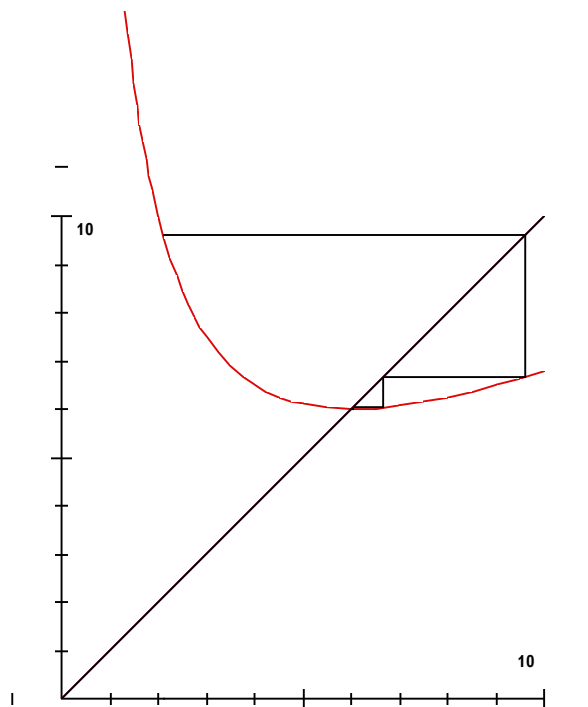
Example 9: $f(x) = \ln(x^2)/\ln 2$
 seed = .6
 fixed point = 4



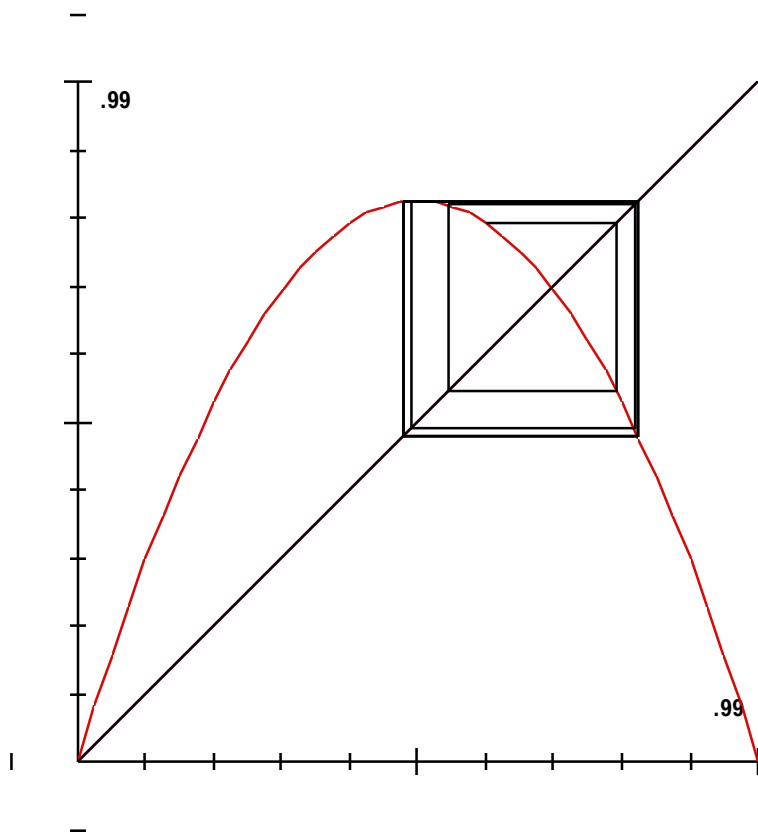
Example 8: $f(x) = \cos^{-1}(x)$
 seed = .82
 diverges; $x_4 > 1$



Example 1: $f(x) = \cos(x)$
 seed = 1.1
 fixed point $\approx .739$



Example 3: $f(x) = (x + 36/x)/2$
 seed = 2.1
 fixed point = 6



$f(x) = 3.3x(1-x)$
 seed = .6
 two-cycle: $x_{13} = .8236 = x_{15}$
 $x_{12} = .4794 = x_{14}$

The Cantor Set

From the interval $[0,1]$ delete the open interval $(1/3,2/3)$. From $[0,1/3]$ delete $(1/9,2/9)$ and from $[2/3,1]$ delete $(7/9,8/9)$. Continue deleting the middle third of the intervals that remain after a given stage. If the process is continued infinitely often, the points that remain constitute the Cantor set, one of the earliest (1883) fractals, though that term did not arise until much later ([1], pp. 79-90). See Fig 2.5 in the Appendix, where the Cantor set is discussed more thoroughly, for the first few stages of the interval-deletion process. Though the connection between this geometric description of the Cantor set and the following analytic construction is not obvious, we must content ourselves here with the latter*. To run the program CANTOR listed below, enter a value of X and get an output. Respond to the prompt with ANS and continue enough times to see whether the sequence of values escapes to infinity or becomes trapped as a single constant or an alternating set of values. Initial values that lead to trapped sequences are called *prisoners* and are members of the Cantor set. Others are called *escapees*. Use these initial values and label them as prisoners (P) or escapees (E):

```
Prgm:CANTOR
:Lbl 1          1_3          .33333      4_27          2_9          .22222
:Input X       8_9          5_9          5_27          8_81          7_81
:If X < .5     5_81          1_243 4_243 3_10          6_10
:3X→A         7_10          8_10
:Disp A       3_10 + 2_3
:If X < .5     7_10 - 2_3
:Goto 1        8_10 - 2_3
:3-3X→A
:Disp A
:Goto 1
```

*The connection between the geometric definition of the Cantor set and the analytic construction can be seen as follows. Running the program CANTOR and feeding the outputs back in is iteration of the function f defined by $f(x) = 3x$ if $0 \leq x \leq 0.5$, $3 - 3x$ if $0.5 < x \leq 1$. As shown in the graphs below, the seed $7/81$ converges to 0 and is a prisoner, whereas the seed 0.8 diverges to $-\infty$. Consider values of x in the middle third $(1/3, 2/3)$ of $[0, 1]$. The function f maps this open interval onto the open interval $(1, 3/2)$, which is then mapped onto the open interval $(-3/2, 0)$. As soon as one iterate becomes negative, all subsequent iterates are negative and their magnitudes become indefinitely large. Thus the interval $(1/3, 2/3)$ consists entirely of escapees. Furthermore, both of the intervals $(1/9, 2/9)$ and $(7/9, 8/9)$, the middle thirds of $[0, 1/3]$ and $[2/3, 1]$, respectively, are mapped by f onto the interval $(1/3, 2/3)$, and subsequent iterations send those iterates off to $-\infty$. We see a pattern, then, which shows that intervals which are removed during the geometric construction of the Cantor set are intervals of escapees under the iteration of f . The following paragraph characterizes members of the Cantor set. The Appendix gives a more extensive discussion.

It can be shown ([1], pp. 85-87) that numbers in $[0,1]$ which possess a triadic (base three) representation without 1's (and only those numbers) are prisoners under this iterative process and thus members of the Cantor set. For example, the prisoner $2/9$ has the triadic representation $.02$, whereas the escapee $5/9 (= 1/3 + 2/9)$ has the representation $.12$ which contains a 1. Note, however, that while the prisoner $1/3$ has the representation $.1$, it also has the representation $.02$ which has no 1. A 1 in the rightmost position can be gotten rid of by replacing it with 02, but no such replacement is possible for a 1 that is not on the right end. (Recall, by way of comparison, that $.9 = 1$ and that $.59 = .6$ in the base ten numeration system.)

Exercise: give triadic representations for the prisoners $8/9$, $7/81$ and $3/10$ and the escapees $5/81$ and $8/10$.

Answers: $.22$, $.00202$, $.0220022$, $.0012$, $.210121$.

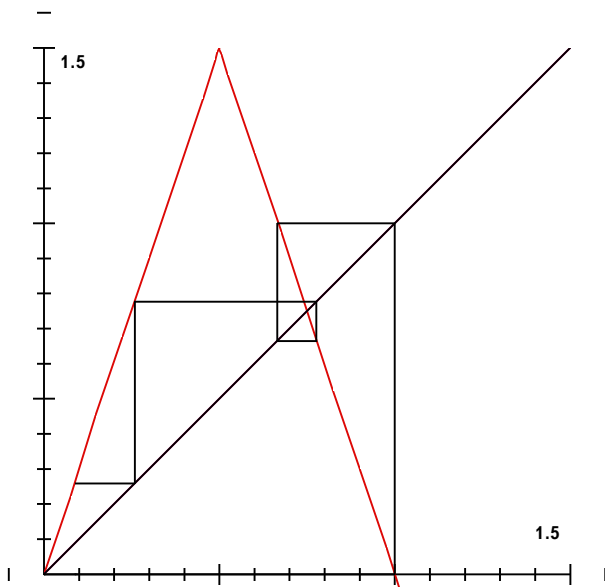
Note that $.0220022 = 2/3^2 + 2/3^3 + 2/3^6 + 2/3^7 + 2/3^{10} + 2/3^{11} + 2/3^{14} + 2/3^{15} + \dots$

$$= 2/3^2 + 2/3^3 + (2/3^6 + 2/3^{10} + 2/3^{14} + \dots) + (2/3^7 + 2/3^{11} + 2/3^{15} + \dots)$$

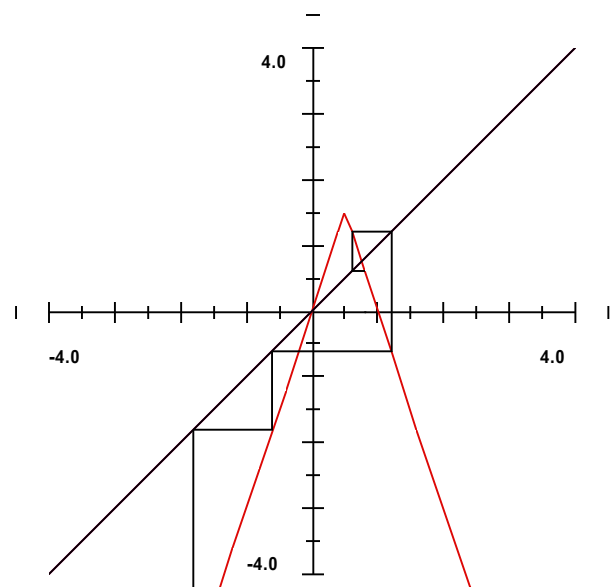
$$= 2/3^2 + 2/3^3 + \frac{2/3^6}{1 - 1/3^4} + \frac{2/3^7}{1 - 1/3^4} \quad (\text{by the geometric series formula})$$

$$= 2/9 + 2/27 + 1/360 + 1/1080 = 3/10.$$

A similar analysis may be used to verify that $8/10 = .210121$.



Cantor set tent:
 $f(x) = 3x$, for $0 \leq x \leq .5$
 $3 - 3x$, for $.5 < x \leq 1$
 seed = $7/81$
 converges to 0



Cantor set tent:
 $f(x) = 3x$, $0 \leq x \leq .5$
 $3 - 3x$, for $.5 < x \leq 1$
 seed = $.8$
 iterates diverge to $-\infty$

Sierpinski triangle

Run the program CHAOS ([3], p. 49) below. Then delete the five lines indicated by ***** and run it again. Then investigate the effect of changing .3333 to .2 and .6666 to .5.

Prog: CHAOS

```

:ClrDraw
:0→Xmin
:1→Xmax
:0→Ymin
:1→Ymax
:Disp "X="
:Input X
:Disp "Y="
:Input Y
:0→C
:X→A      ***** (don't type the *'s)
:Y→B      *****
:Lbl 1
:C+1→C
:Pt-On(X,Y)
:Line(A,B,X,Y)      *****
:X→A      *****
:Y→B      *****
:If C>1500
:End
:Rand→N
:If N<.3333
Goto 2
:If N>.6666
:Goto 3
:0.5X→X
:0.5Y→Y
:Goto 1
:Lbl 2
:0.5(X+1)→X
:0.5Y→Y
:Goto 1
:Lbl 3
:0.5(X+.5)→X
:0.5(Y+1)→Y
:Goto 1

```

The Chaos Game

Preparations: Take a sheet of paper and a pencil and mark three points on the sheet, label them 1, 2, and 3, and call them *bases*. Have a die which allows you to pick the numbers 1, 2, and 3 randomly. It is obvious how to manufacture such a die. Take an ordinary die and just identify the faces 6 with 1, 5 with 2, and 4 with 3.

Rules: Start the game by picking an arbitrary point on the sheet of paper and mark it by a small dot. Call it the *game point*. Now role the die. If number 2, for example, comes up, consider the line between the game point and base 2 and mark a dot exactly in the middle, i.e. halfway between the game point and base 2. This dot will be the new game point, and we have completed the first cycle of the game. Now repeat (i.e. role the die again) to randomly get the number 1, 2, or 3; and depending on the result, mark a dot halfway between the last game point and the randomly chosen base.

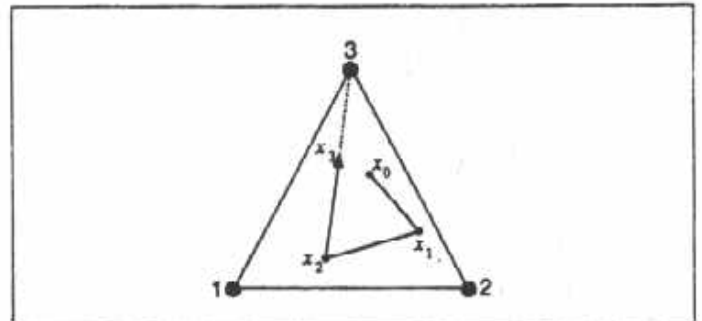


Figure 1.20 : The three base points (vertices of a triangle) and a few iterations of the game point.

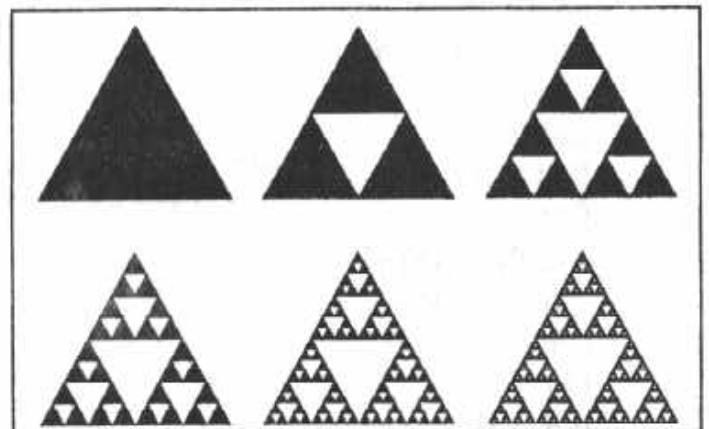


Figure 2.16 : The basic construction steps of the Sierpinski gasket.

The Fern - a fractal curve

Here is a program adapted for the TI-81 and TI-82 from a BASIC program in [1], p. 376. The output is not as good as the picture shown, but it does look like a fern. Run the program.

```
Prog:FERN
:ClrDraw
:0→Xmin
:50→Xmax
:0→Ymin
:60→Ymax
:1→I
:10→L
:50→W
:W+L→V
: .5W→E
: .57W→F
: .408W→G
: .1075W→H
:0W→M
: -.036W→N
: .0893W→P
: .27W→Q
:E→X
:0→Y
:Lbl 1
:Rand→R
:If R > .02
:Goto 2
:0X+0Y+E→X
:0X+.27Y+M→Y
:Goto 5
:Lbl 2
:If R > .17
:Goto 3
: -.139X+.263Y+F→X
: .246X+.224Y+N→Y
:Goto 5
:Lbl 3
:If R > .3
:Goto 4
: .17X-.215Y+G→X
: .222X+.176Y+P→Y
:Goto 5
:Lbl 4
: .781X+.034Y+H→X
: -.032X+.739Y+Q→Y
:Lbl 5
:Pt-On(X+L,V-Y)
:I+1→I
:If I > 1000
:End
:Goto 1
```

Screen Image of Chaos
Fern



Programming the Mandelbrot set and enlargements

The Mandelbrot set is used -- or misused -- worldwide to drive computers crazy. Home Computers and PC's do it, workstations do it, and even Super Computers are regularly pushed to their limits with the Mandelbrot set. Clearly, programmable, hand-held pocket calculators do not stand a chance competing against their big brothers. Given their limitations it is astonishing that they are at all capable of rendering images of the Mandelbrot set. Furthermore, it is amazing how much they can achieve with so little programming effort. On the right is a list of the complete TI-81 code consisting of mere 48 lines. Yet it provides the main features that one expects (except for color display).

- * Computation of a global view of the set
- * Interactive positioning of a zoom window.
- * Computation and display of the selected enlargement.

The last two steps may be repeated. Here are the operating instructions:

1. Type in the program and run it.
2. Type "0" when prompted "ZOOM? (0,1)".
3. Enter the number N of maximal iterations upon "MAX ITER?"
4. Now the program starts to compute (this may take a while).
5. After termination of the program a picture of the Mandelbrot set (or a section of it) is on the display. Now, a window for an enlargement can be chosen. First, push the ZOOM button.
6. Select the item "1 :Box" from the displayed zoom menu.
7. Using the four cursor control buttons position the cursor to the upper left corner of the desired zoom window. The coordinates of the point are displayed. Push ENTER.
8. Repeat for the lower right hand corner. The outline of the window is dragged across the display. Push ENTER and CLEAR.
9. Start the program again. This time, enter "1" when responding to "ZOOM? (0,1)". Continue with step 3.

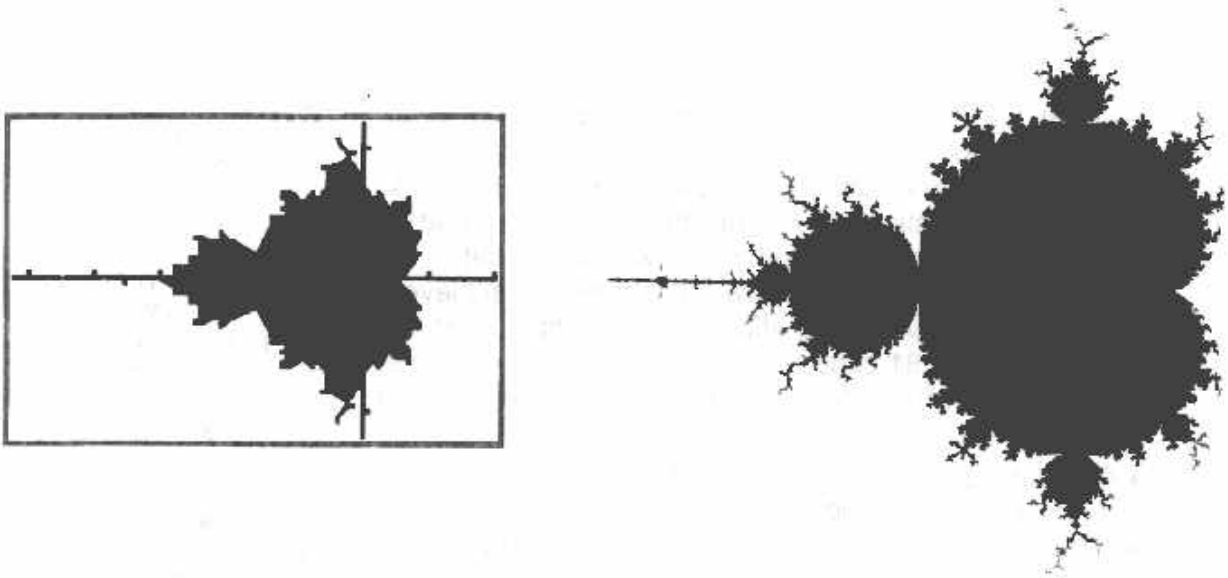
Explanations: When setting $N = 15$ the overview picture takes 25 minutes to compute. Enlargements require longer times. Setting the zoom box in steps 6 -- 8 defines the variables $Xmin$ to $Ymax$ which, when not zooming, are defined in lines 9-17 of the program. The display consists of 64 lines of 96 pixels each. When computing the default overview the symmetry of the Mandelbrot set about the real axis is exploited and only 32 lines are computed. The program variables J and I denote the line and pixel number. K is the iteration counter. The loops in the program structure are indicated by the arrows. A similar program can be written for the CASIO graphing calculator.

```

1 All-off
2 Clr-Draw
3 Disp "ZOOM? (0,1)"
4 Input Q
5 Disp "MAX ITER?"
6 Input N
7 63 → Z
8 If Q
9 Goto 0
10 | -1.2 → Ymin
11 | 1.2 → Ymax
12 | 0.5 → Yscl
13 | -2.6 → Xmin
14 | 1.0 → Xmax
15 | 0.5 → Xscl
16 | 31 → Z
17 Lbl 0
18 DispGraph
19 (Xmax - Xmin) / 95 → C
20 (Ymax - Ymin) / 63 → D
21 0 → J
22 Lbl J
23 ↑ Ymin + JD → B
24 0 → I
25 Lbl I
26 ↑ Xmin + IC → A
27 A → X
28 B → Y
29 1 → K
30 Lbl 1
31 ↑ XX → U
32 YY → V
33 2XY + B → Y
34 U - V + A → X
35 K + 1 → K
36 If U + V > 100
37 Goto 2
38 If K < N
39 Goto 1
40 Pnt-On (A,B)
41 If Z=31
42 Pnt-On (A,-B)
43 Lbl 2
44 IS > (I,95)
45 Goto I
46 IS > (J,Z)
47 Goto J
48 End

```

The program on the previous page ([4], p. 154) produced the left version of the Mandelbrot set shown below. A better picture is on the right.



References and Software

- [1] Peitgen, Heinz-Otto; Jürgens, Hartmut; Saupe, Dietmar. Fractals for the Classroom' Part One: Introduction to Fractals and Chaos. Springer-Verlag, New York, 1992. (Available through NCTM)
- [2] Peitgen, Heinz-Otto; Jürgens, Hartmut; Saupe, Dietmar. Fractals for the Classroom, Part Two: Complex Systems and Mandelbrot Set. Springer-Verlag, New York, 1992. (Available through NCTM)
- [3] Peitgen, Heinz-Otto; Jürgens, Hartmut; Saupe, Dietmar; Maletsky, Evan; Perciante, Terence, Yunker, Lee. Fractals for the Classroom, Strategic Activities Volume One. Springer-Verlag, New York, 1991. (Available through NCTM)
- [4] Peitgen, Heinz-Otto; Jürgens, Hartmut; Saupe, Dietmar; Maletsky, Evan; Perciante, Terence; Yunker, Lee. Fractals for the Classroom, Strategic Activities Volume Two. Springer-Verlag, New York, 1992. (Available through NCTM)
- [5] Analyzer, Version 3.2.0. Software written by Beverly H. West and Douglas S. Alfors. Available from Addison-Wesley Publishing Co.

2.1 The Cantor Set

Cantor (1845–1918) was a German mathematician at the University of Halle where he carried out his fundamental work in the foundations of mathematics, which we now call *set theory*.

Georg Cantor



Figure 2.3 : Georg Cantor, 1845–1918.

The Cantor set was first published⁴ in 1883 and emerged as an example of certain exceptional sets.⁵ It is probably fair to say that in the zoo of mathematical monsters — or early fractals — the Cantor set is by far the most important, though it is less visually appealing and more distant to an immediate natural interpretation than some of the others. It is now understood that the Cantor set plays a role in many branches of mathematics; and in fact, in a very deep sense, in chaotic dynamical systems (we will touch upon this property at least a bit) and is somehow hidden as the essential skeleton or model behind many other fractals (for example Julia sets, as we will see in chapter 12).

The basic Cantor set is an infinite set of points in the unit interval $[0,1]$. That is, it can be interpreted as a set of certain numbers, as for example $0, 1, 1/3, 2/3, 1/9, 2/9, 7/9, 8/9, 1/27, 2/27, \dots$ Plotting these and all other points (assuming we could know what they were) would not make much of a picture at all. Therefore, we use a common little trick. Rather than plotting

⁴G. Cantor, *Über unendliche, lineare Punktmengenfolgen* V, *Mathematische Annalen* 21 (1883) 545–591.

⁵The Cantor set is an example of a perfect, nowhere dense subset.

The Cantor Set



Figure 2.4 : The Cantor set represented by vertical lines whose base points are exactly at all the different points belonging to the set.

just points we plot vertical lines all of the same length whose base points are exactly at all the different points belonging to the Cantor set. By so doing, we are able to see the distribution of these points a bit better. Figure 2.4 gives a first impression. Rather than being able to actually see the Cantor set, it is probably much more important to remember its classical construction.

Start with the interval $[0,1]$. Now take away the (open) interval $(1/3, 2/3)$, i.e., remove the middle third from $[0,1]$, but not the numbers $1/3$ and $2/3$. This leaves two intervals $[0, 1/3]$ and $[2/3, 1]$ of length $1/3$ each and completes a basic construction step. Now we repeat, we look at the remaining intervals $[0, 1/3]$ and $[2/3, 1]$ and remove their middle thirds, which yields four intervals of length $1/9$. Continue on in this way. In other words, there is a feedback process in which a sequence of (closed) intervals is generated — one after the first step, two after the second step, four after the third step, eight after the fourth step, etc. (i.e. 2^n intervals of length $1/3^n$ after the n th step). Figure 2.5 visualizes the construction.

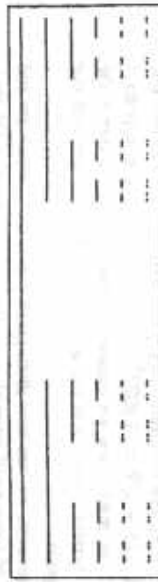


Figure 2.5 : Some initial steps of the construction.

What is the Cantor set? It is the set of points which remain if we carry out the removal steps infinitely often. How do we explain infinitely often? Let us try. A point, say x , is in the Cantor set if we can guarantee that no matter how often we carry out the removal process, the point x will not be taken out. Obviously $0, 1, 1/3, 2/3, 1/9, 2/9, 7/9, 8/9, 1/27, 2/27, \dots$ are examples of such points

Interval End Points
Are in the Cantor Set

because they are the end points of the intervals which are created in the steps; and therefore, they must remain. All these points have one thing in common. Namely, they are related to powers of 3 — or rather, to powers of $1/3$. This is an important observation which we will exploit later to understand the Cantor set. One is tempted to believe that any point in the Cantor set is of this kind, i.e. an end point of one of the small intervals generated in the process. This conclusion is categorically wrong. We will not give the complete argument but at least discuss the fact to some extent.

... But That's Not All

If the Cantor set were just the end points of the intervals of the generating process, we could easily enumerate them as shown in figure 2.6.



Figure 2.6 : Counting end points of intervals from the Cantor set construction. In stage k , $k > 0$ of the construction process 2^k new end points are added and enumerated as shown.

16

That means, the Cantor set would be a countable set, but it is known to be uncountable.⁶ That is, there is no way to enumerate the points in the Cantor set. Thus, there must be many more points which are not end points. Can we give examples which are not end points? To name such examples, we will use a simple, but far reaching characterization of the Cantor set, namely by triadic numbers.

Let us write some of the points of the Cantor set as triadic numbers: $1/3$ is 0.1 in the triadic system, $2/3$ is 0.2 , $1/9$ is 0.01 , and $2/9$ is 0.02 . In general we can characterize any point of the Cantor set in the following way.

84

Fact. *The Cantor set C is the set of points in $[0, 1]$ for which there is a triadic expansion that does not contain the digit '1'.*

This number theoretic characterization eliminates the problem of the existence of a limit for the geometric construction of the Cantor set.

The above examples $2/3$ and $2/9$ are points in the Cantor set according to this statement, since their triadic expansions 0.2 and 0.02 do not contain any digits '1'. However, the other two examples $1/3$ and $1/9$ seem to contradict the rule, their expansions

0.1 and 0.01 clearly show a digit '1'. Yes, that is correct, but remember that we have ambiguity in our representations, and $1/3$ can also be written as $0.0222\bar{2}$. Therefore, it belongs. But then, you may ask, what about $1/3 + 1/9$? This is a number from the middle third interval which is discarded in the first construction step of the Cantor set. It has a triadic expansion 0.11 , and can't we also write this in different form and thus get into trouble? Yes, indeed, $1/3 + 1/9 = 0.1022\bar{2}$; but as you see, there appears a digit '1' no matter how we choose to represent that number in the triadic system. Thus, it is out and there is no problem with our description.

Moreover, we can now distinguish points in C which are end points of some small interval occurring in the process of the feedback construction from those points which are definitely not. Indeed, end points in this sense just correspond to numbers which have triadic expansion ending with infinitely many consecutive 2's or 0's. All other possibilities, as for example

0.020022000222000022220000022222...

or a number in which we pick digits 0 and 2 at random will belong to the Cantor set but are not end points, and those are, in fact, more typical for the Cantor set. In other words, if one picks a number from C at random, then with probability 1, it will not be an end point. By this characterization of the Cantor set we can understand that, in fact, any point in C can be approximated arbitrarily closely by other points from C , and yet C itself is a dust of points. In other words, there is nothing like an interval in C (which is obvious if we recall the geometric construction, namely the removal of intervals).

Distinguishing End Points from Others

The Cardinality of the Cantor Set

We can now see that the cardinality of the Cantor set must be the same as the cardinality of the unit interval $[0, 1]$. We start with the interval $[0, 1]$ and show how each point in it corresponds to one point in the Cantor set.

- Each point in the interval has a binary expansion.
- Each binary expansion corresponds to a path in the binary tree for binary decimals.
- Each such path has a corresponding path in the ternary tree for the Cantor set.
- Each path in the ternary tree of the Cantor set identifies a unique point in the Cantor set by an address in triadic expansion.

Therefore, for each number in the interval, there is a corresponding point in the Cantor set. For different numbers there are different points. Thus, the cardinality of the Cantor set must be at least as large as the cardinality of the interval. On the other hand, it cannot exceed this cardinality, because the Cantor set is a subset of the interval. Therefore, both cardinalities must be the same.

17

The Cantor set is truly complex, but is it also self-similar? Yes, indeed, if one takes the part of C which lies in the interval $[0, 1/3]$, for example, we can regard that part as a scaled-down version of the entire set. How do we see that? Let us take the definition of the Cantor set collecting all points in $[0, 1]$ which admit a triadic representation not containing digit 1. Now for every point, say

$$\xi = \alpha_1 \times 3^{-1} + \alpha_2 \times 3^{-2} + \alpha_3 \times 3^{-3} + \alpha_4 \times 3^{-4} + \dots$$

(with $\alpha_i \in \{0, 2\}$) in the Cantor set we find a corresponding one in $[0, 1/3]$ by dividing ξ by 3, i.e.

$$\frac{\xi}{3} = 0 \times 3^{-1} + \alpha_1 \times 3^{-2} + \alpha_2 \times 3^{-3} + \alpha_3 \times 3^{-4} + \dots$$

Indeed, if $x = 0.200220\dots$ and we multiply by $1/3 = 0.1$, that means that we just shift the decimal point one place to the left (i.e., we obtain $0.0200220\dots$), which is in C again. Thus, the part of the Cantor set present in $[0, 1/3]$ is an exact copy of the entire Cantor set scaled down by the factor $1/3$ (see figure 2.13). For the part of C which lies in the interval $[2/3, 1]$, essentially we can do the same calculation (we only have to include the addition of $2/3 = 0.2$). In the same way, any subinterval in the geometric Cantor set construction contains the entire Cantor set scaled down by an appropriate factor of $1/3^k$. In other words, the Cantor set can be seen as a collection of arbitrarily small pieces, each of which is an exact scaled-down version of the entire Cantor set. This is what we mean when we say the Cantor set is self-similar. Thus, taking self-similarity as an intuitive property means that self-similarity here

is absolutely perfect and is true for an infinite range. Note that in our discussion of self-similarity we have carefully avoided the geometrical model of the Cantor set. Instead, we have exploited the number theoretic representation.

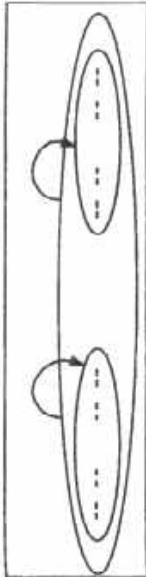


Figure 2.13 : The Cantor set is a collection of two exact copies of the entire Cantor set scaled down by the factor $1/3$.

Note that the scaling property of the Cantor set corresponds to the following invariance property. Take a point from C and multiply it by $1/3$. The result will be in C again. The same is true if we first multiply by $1/3$ and then add $2/3$. This is apparent from the triadic characterization, and it will be the key observation for chapter 5.

Before we continue our introduction to classical fractals with some other examples, let us touch one more property of the Cantor set which reveals an important dynamic interpretation and an amazing link with chaos.

Let us look at a mathematical feedback system defined in the following way. If x is an input number, then the output number is determined by the following conditional formula (2.3).

$$x \rightarrow \begin{cases} 3x & \text{if } x \leq 0.5 \\ -3x + 3 & \text{if } x > 0.5 \end{cases} \quad (2.3)$$

In other words, the output is evaluated to be $3x$ if $x < 0.5$ and is $-3x + 3$ if $x \geq 0.5$.

Starting with an initial point x_0 , the feedback process defines a sequence

$$x_0, x_1, x_2, x_3, \dots$$

The interesting question then is: what is the long-term behavior of such sequences? For many initial points x_0 the answer is very easy to derive. Take for example a number $x_0 < 0$. Then $x_1 = 3x_0$, and $x_1 < 0$. By induction it follows that all numbers x_k from this sequence are negative, and, thus

$$x_k = 3^k x_0.$$

This sequence then grows negatively without any bound, it tends to negative infinity, $-\infty$. Let us call a sequence with such a long-term behavior an *escaping sequence* and the initial point x_0 an *escaping point*.

Let us now take $x_0 > 1$. Then $x_1 = -3x_0 - 3 < 0$, and again the sequence escapes to $-\infty$. But not all points are escaping points. For example, for $x_0 = 0$ we have that all succeeding numbers in the sequence are also equal to zero. We conclude, that any initial point x_0 , which at some stage goes to zero will remain there forever, and thus is not an escaping point. Such points we call *prisoners*. So far we have found that all prisoner points must be in the unit interval $[0, 1]$. This leads to the interesting question: which points in the unit interval will remain and which will escape? Let us look at some examples.

x_0	x_1	x_2	x_3	x_4	...	P/E
0	0	0	0	0	...	prisoner
$1/3$	1	0	0	0	...	prisoner
$1/2$	$3/2$	$-3/2$	$-9/2$	$-27/2$...	escapee
$1/5$	$3/5$	$6/5$	$-3/5$	$-9/5$...	escapee

Clearly the entire (open) interval $(1/3, 2/3)$ escapes because when $1/3 < x_0 < 2/3$ we have $x_1 > 1$ and $x_2 < 0$. But then every point which eventually lands in that interval will also escape under iteration. Figure 2.14 illustrates these points and reveals the Cantor set construction for the points which will remain.

Fact. The prisoner set P for the feedback system given by eqn. (2.3) is the Cantor set, while all points in $[0, 1]$ which are outside the Cantor set belong to the escape set E .

This is a remarkable result and shows that the study of the dynamics of feedback systems can provide an interpretation for the Cantor set. This close relation between chaos and fractals will be continued in chapter 12.

Escaping Intervals

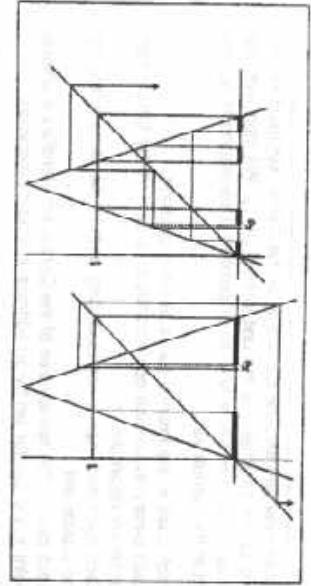


Figure 2.14 : Mechanism for escaping points.

An application of iteration

This application deals with chlorine concentration in a swimming pool. It was originally presented at the NCTM 1991 Annual Meeting in New Orleans by Joan Reinthaler, Sidwell Friends School, Washington, DC, who adapted it for publication in the Fall 1991 issue of COMAP Consortium. I am grateful to Sally Dodge, Evanston (IL) Township High School, for this reference.

Preliminary facts

1. Chlorine dissipates in reaction to bacteria and to the sun at a rate of about 15% of the amount present per day. When the concentration is high, it dissipates much more quickly.
2. The ideal concentration of chlorine in a pool is 1 to 2 parts per million (ppm); 3 ppm is safe though uncomfortable to the eyes.
3. Pools need to be "shocked" every once in a while with a very high dose to clean out pumps and filters as well as the pool itself. After "shocking" the concentration drops quickly and reaches 3 ppm in about two hours.
4. It is normal practice to add small amounts of chlorine every day to maintain the ideal 1 to 2 ppm concentration.

Problem

Suppose that a pool has been "shocked" and the chlorine concentration has dropped to 3 ppm. Let $t = 0$ at this point in time and let $C(t)$ be the concentration at

time t , where t is measured in days. If no chlorine is added, $C(1) = 3 \cdot (0.85)^1 = 2.55$, $C(2) = 3 \cdot (0.85)^2 = 2.1675$, $C(3) = 3 \cdot (0.85)^3 = 1.84$, $C(4) = 3 \cdot (0.85)^4 = 1.566$, $C(5) = 3 \cdot (0.85)^5 = 1.33$, $C(6) = 3 \cdot (0.85)^6 = 1.13$, $C(7) = 3 \cdot (0.85)^7 = 0.96 < 1$.

Let us experiment with various amounts A (in ppm) to be added to the pool at the same time each morning to maintain the chlorine concentration at an acceptable level, where, as above, $C(0) = 3$. We will first try $A = 0.2$. Perform the following TI-81 or 82 keystrokes:

```
.85  $\times$  3 + .2 ENTER      Output is 2.75  
.85 ANS + .2 ENTER      Output is 2.5375  
ENTER                  Output is 2.356875  
ENTER                  Output is 2.20334375
```

Continue to press ENTER and observe the stabilization at approximately 1.333 ppm.

Repeat this sequence of steps using $A = 0.1$. What do you observe?

Use other values of A and record your observations. Can you find a value of A that produces a stabilization at 1.5 ppm?

Now let $C(0) = 100$ and $A = 0.2$. How many days does it take for the concentration to drop below 2.0 ppm?

With $A = 0.2$ you have been iterating the function f defined by $f(x) = .85x + .2$. For such a simple function the fixed point x^* can be found by solving the equation $x^* = .85x^* + .2$ to get $x^* = .2/.15 = 1.333\dots$. You can then verify that, if it is desired to have $x^* = 1.5$, A should be .225.