EXCERPTED FROM

STEPHEN WOLFRAM

A NEW KIND OF SCIENCE

CHAPTER 4

Systems Based on Numbers

4

Systems Based on Numbers

The Notion of Numbers

Much of science has in the past ultimately been concerned with trying to find ways to describe natural systems in terms of numbers.

Yet so far in this book I have said almost nothing about numbers. The purpose of this chapter, however, is to investigate a range of systems that are based on numbers, and to see how their behavior compares with what we have found in other kinds of systems.

The main reason that systems based on numbers have been so popular in traditional science is that so much mathematics has been developed for dealing with them. Indeed, there are certain kinds of systems based on numbers whose behavior has been analyzed almost completely using mathematical methods such as calculus.

Inevitably, however, when such complete analysis is possible, the final behavior that is found is fairly simple.

So can systems that are based on numbers ever in fact yield complex behavior? Looking at most textbooks of science and mathematics, one might well conclude that they cannot. But what one must realize is that the systems discussed in these textbooks are usually ones that are specifically chosen to be amenable to fairly complete analysis, and whose behavior is therefore necessarily quite simple.

And indeed, as we shall see in this chapter, if one ignores the need for analysis and instead just looks at the results of computer

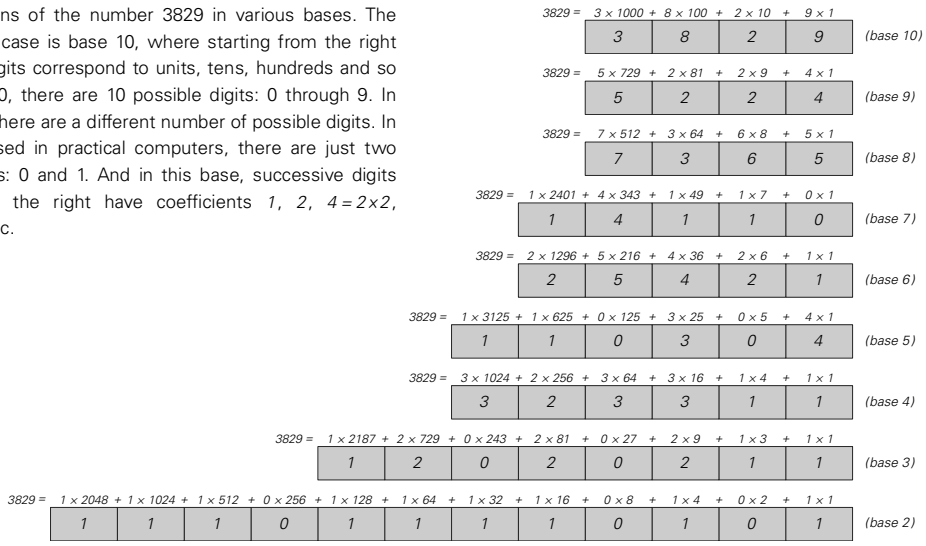
experiments, then one quickly finds that even rather simple systems based on numbers can lead to highly complex behavior.

But what is the origin of this complexity? And how does it relate to the complexity we have seen in systems like cellular automata?

One might think that with all the mathematics developed for studying systems based on numbers it would be easy to answer these kinds of questions. But in fact traditional mathematics seems for the most part to lead to more confusion than help.

One basic problem is that numbers are handled very differently in traditional mathematics from the way they are handled in computers and computer programs. For in a sense, traditional mathematics makes a fundamental idealization: it assumes that numbers are elementary objects whose only relevant attribute is their size. But in a computer, numbers are not elementary objects. Instead, they must be represented explicitly, typically by giving a sequence of digits.

The idea of representing a number by a sequence of digits is familiar from everyday life: indeed, our standard way of writing numbers corresponds exactly to giving their digit sequences in base 10. What base 10 means is that for each digit there are 10 possible choices:



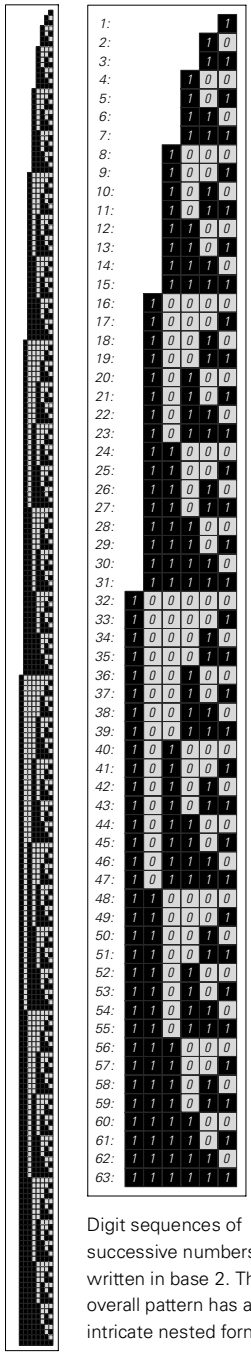
Representations of the number 3829 in various bases. The most familiar case is base 10, where starting from the right successive digits correspond to units, tens, hundreds and so on. In base 10, there are 10 possible digits: 0 through 9. In other bases, there are a different number of possible digits. In base 2, as used in practical computers, there are just two possible digits: 0 and 1. And in this base, successive digits starting from the right have coefficients *1*, *2*, *4 = 2×2*, *8 = 2×2×2*, etc.

0 through 9. But as the picture at the bottom of the facing page shows, one can equally well use other bases. And in practical computers, for example, base 2 is almost always what is used.

So what this means is that in a computer numbers are represented by sequences of 0's and 1's, much like sequences of white and black cells in systems like cellular automata. And operations on numbers then correspond to ways of updating sequences of 0's and 1's.

In traditional mathematics, the details of how operations performed on numbers affect sequences of digits are usually considered quite irrelevant. But what we will find in this chapter is that precisely by looking at such details, we will be able to see more clearly how complexity develops in systems based on numbers.

In many cases, the behavior we find looks remarkably similar to what we saw in the previous chapter. Indeed, in the end, despite some confusing suggestions from traditional mathematics, we will discover that the general behavior of systems based on numbers is very similar to the general behavior of simple programs that we have already discussed.



Digit sequences of successive numbers written in base 2. The overall pattern has an intricate nested form.

Elementary Arithmetic

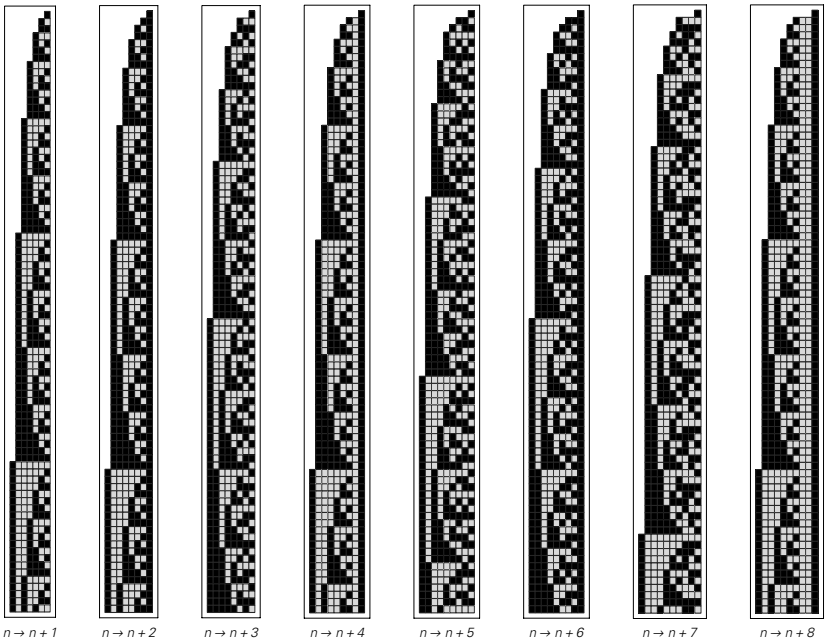
The operations of elementary arithmetic are so simple that it seems impossible that they could ever lead to behavior of any great complexity. But what we will find in this section is that in fact they can.

To begin, consider what is perhaps the simplest conceivable arithmetic process: start with the number 1 and then just progressively add 1 at each of a sequence of steps.

The result of this process is to generate the successive numbers 1, 2, 3, 4, 5, 6, 7, 8, … The sizes of these numbers obviously form a very simple progression.

But if one looks not at these overall sizes, but rather at digit sequences, then what one sees is considerably more complicated. And in fact, as the picture on the right demonstrates, these successive digit sequences form a pattern that shows an intricate nested structure.

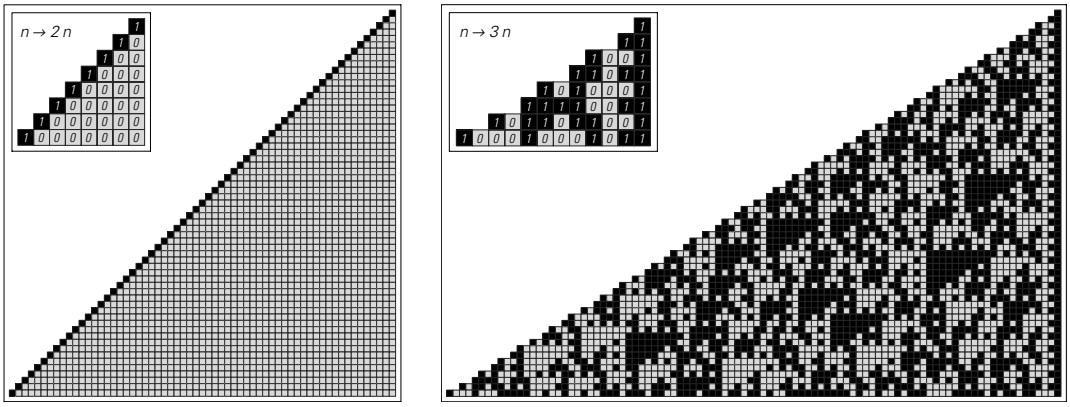
The pictures below show what happens if one adds a number other than 1 at each step. Near the right-hand edge, each pattern is somewhat different. But at an overall level, all the patterns have exactly the same basic nested structure.



Digit sequences in base 2 of numbers obtained by starting with 1 and then successively adding a constant at each step. All these patterns ultimately have the same overall nested form.

If instead of addition one uses multiplication, however, then the results one gets can be very different. The first picture at the top of the facing page shows what happens if one starts with 1 and then successively multiplies by 2 at each step.

It turns out that if one represents numbers as digit sequences in base 2, then the operation of multiplying by 2 has a very simple effect: it just shifts the digit sequence one place to the left, adding a 0 digit on the right. And as a result, the overall pattern obtained by successive multiplication by 2 has a very simple form.



Patterns produced by starting with the number 1, and then successively multiplying by a factor of 2, and a factor of 3. In each case, the digit sequence of the number obtained at each step is shown in base 2. Multiplication by 2 turns out to correspond just to shifting all digits in base 2 one position to the left, so that the overall pattern produced in this case is very simple. But multiplication by 3 yields a much more complicated pattern, as the picture on the right shows. Note that in these pictures the complete numbers obtained at each step correspond respectively to the successive integer powers of 2 and of 3.

But if the multiplication factor at each step is 3, rather than 2, then the pattern obtained is quite different, as the second picture above shows. Indeed, even though the only operation used was just simple multiplication, the final pattern obtained in this case is highly complex.

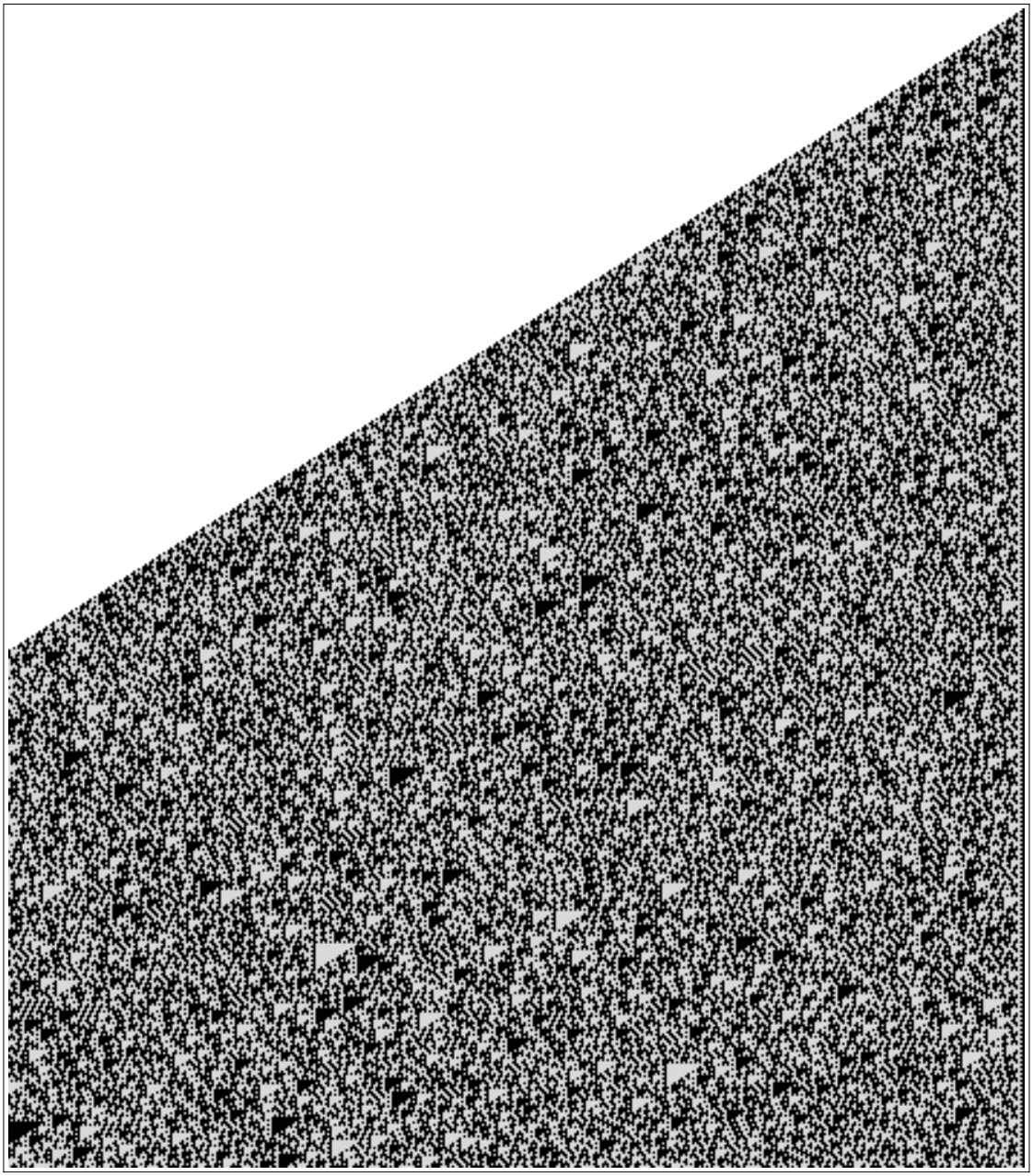
The picture on the next page shows more steps in the evolution of the system. At a small scale, there are some obvious triangular and other structures, but beyond these the pattern looks essentially random.

So just as in simple programs like cellular automata, it seems that simple systems based on numbers can also yield behavior that is highly complex and apparently random.

But we might imagine that the complexity we see in pictures like the one on the next page must somehow be a consequence of the fact that we are looking at numbers in terms of their digit sequences—and would not occur if we just looked at numbers in terms of their overall size.

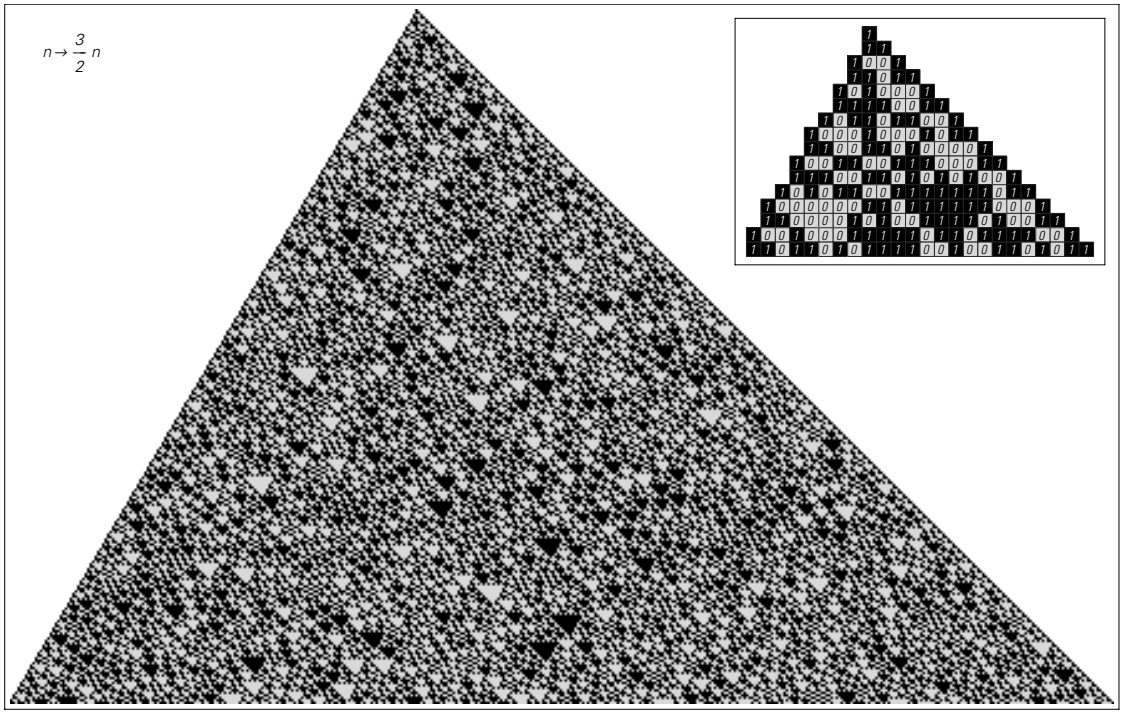
A few examples, however, will show that this is not the case.

To begin the first example, consider what happens if one multiplies by $3/2$, or 1.5, at each step. Starting with 1, the successive numbers that one obtains in this way are 1, $3/2 = 1.5$, $9/4 = 2.25$, $27/8 = 3.375$, $81/16 = 5.0625$, $243/32 = 7.59375$, $729/64 = 11.390625$, ...



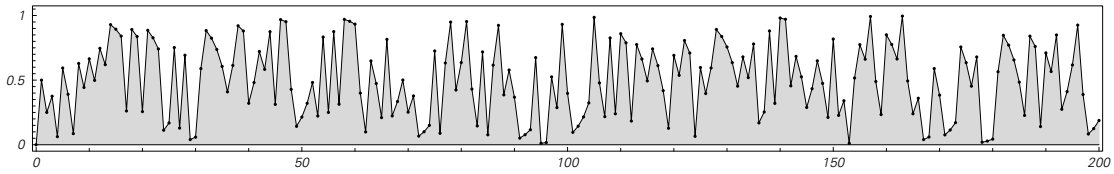
The first 500 powers of 3, shown in base 2. Some small-scale structure is visible, but on a larger scale the pattern seems for all practical purposes random. Note that the pattern shown here has been truncated at the edge of the page on the left, although in fact the whole pattern continues to expand to the left forever with an average slope of $Log[2, 3] \approx 1.58$.

The picture below shows the digit sequences for these numbers given in base 2. The digits that lie directly below and to the left of the original 1 at the top of the pattern correspond to the whole number part of each successive number (e.g. 3 in 3.375), while the digits that lie to the right correspond to the fractional part (e.g. 0.375 in 3.375).



Successive powers of 3/2, shown in base 2. Multiplication by 3/2 can be thought of as multiplication by 3 combined with division by 2. But division by 2 just does the opposite of multiplication by 2, so in base 2 it simply shifts all digits one position to the right. The overall pattern is thus a shifted version of the pattern shown on the facing page.

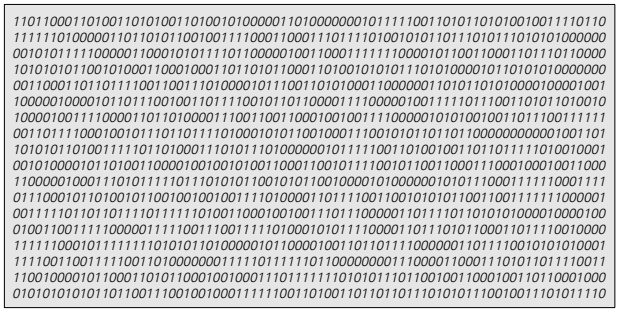
And instead of looking explicitly at the complete pattern of digits, one can consider just finding the size of the fractional part of each successive number. These sizes are plotted at the top of the next page. And the picture shows that they too exhibit the kind of complexity and apparent randomness that is evident at the level of digits.



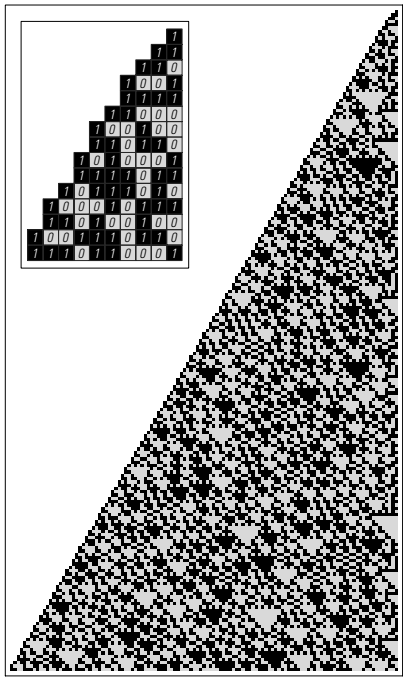
Sizes of the fractional parts of successive powers of 3/2. These sizes are completely independent of what base is used to represent the numbers. Only the dots are significant; the shading and lines between them are just included to make the plot easier to read.

The example just given involves numbers with fractional parts. But it turns out that similar phenomena can also be found in systems that involve only whole numbers.

As a first example, consider a slight variation on the operation of multiplying by $3/2$ used above: if the number at a particular step is even (divisible by 2), then simply multiply that number by $3/2$, getting a whole number as the result. But if the number is odd, then first add 1—so as to get an even number—and only then multiply by $3/2$.



Results of starting with the number 1, then applying the following rule: if the number at a particular step is even, multiply by 3/2; otherwise, add 1, then multiply by 3/2. This procedure yields a succession of whole numbers whose digit sequences in base 2 are shown at the right. The rightmost digits obtained at each step are shown above. The digit is 0 when the number is even and 1 when it is odd, and, as shown, the digits alternate in a seemingly random way. It turns out that the system described here is closely related to one that arose in studying the register machine shown on page 100. The system here can be represented by the rule *n → If[EvenQ[n], 3 n/2, 3 (n + 1)/2]*, while the one on page 100 follows the rule *n → If[EvenQ[n], 3 n/2, (3 n + 1)/2]*. After the first step these systems give the same sequence of numbers, except for an overall factor of 3.



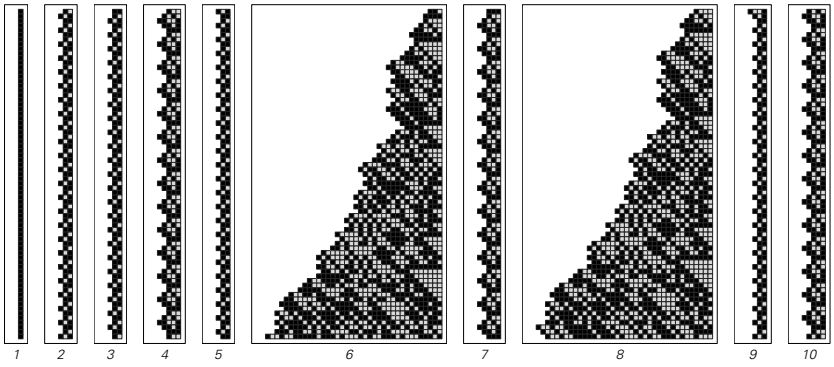
This procedure is always guaranteed to give a whole number. And starting with 1, the sequence of numbers one gets is 1, 3, 6, 9, 15, 24, 36, 54, 81, 123, 186, 279, 420, 630, 945, 1419, 2130, 3195, 4794, …

Some of these numbers are even, while some are odd. But as the results at the bottom of the facing page illustrate, the sequence of which numbers are even and which are odd seems to be completely random.

Despite this randomness, however, the overall sizes of the numbers obtained still grow in a rather regular way. But by changing the procedure just slightly, one can get much less regular growth.

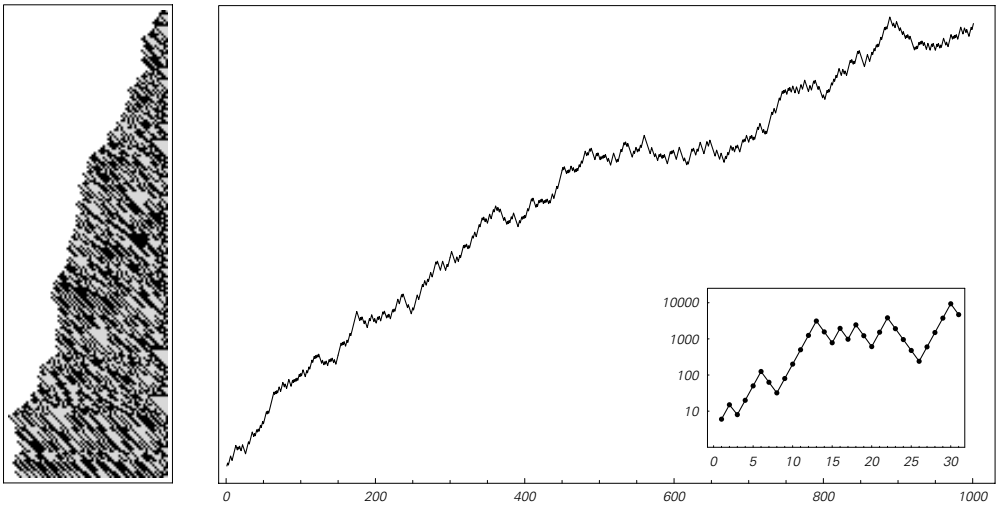
As an example, consider the following procedure: if the number obtained at a particular step is even, then multiply this number by $5/2$; otherwise, add 1 and then multiply the result by $1/2$.

If one starts with 1, then this procedure simply gives 1 at every step. And indeed with many starting numbers, the procedure yields purely repetitive behavior. But as the picture below shows, it can also give more complicated behavior.



Results of applying the rule $n \to$ *If[EvenQ[n], 5 n/2, (n + 1)/2]*, starting with different initial choices of n. In many cases, the behavior obtained is purely repetitive. But in some cases it is not.

Starting for example with the number 6, the sizes of the numbers obtained on successive steps show a generally increasing trend, but there are considerable fluctuations, and these fluctuations seem to be essentially random. Indeed, even after a million steps, when the



The results of following the same rule as on the previous page, starting from the value 6. Plotted on the right are the overall sizes of the numbers obtained for the first thousand steps. The plot is on a logarithmic scale, so the height of each point is essentially the length of the digit sequence for the number that it represents—or the width of the row on the left.

number obtained has 48,554 (base 10) digits, there is still no sign of repetition or of any other significant regularity.

So even if one just looks at overall sizes of whole numbers it is still possible to get great complexity in systems based on numbers.

But while complexity is visible at this level, it is usually necessary to go to a more detailed level in order to get any real idea of why it occurs. And indeed what we have found in this section is that if one looks at digit sequences, then one sees complex patterns that are remarkably similar to those produced by systems like cellular automata.

The underlying rules for systems like cellular automata are however usually rather different from those for systems based on numbers. The main point is that the rules for cellular automata are always local: the new color of any particular cell depends only on the previous color of that cell and its immediate neighbors. But in systems based on numbers there is usually no such locality.

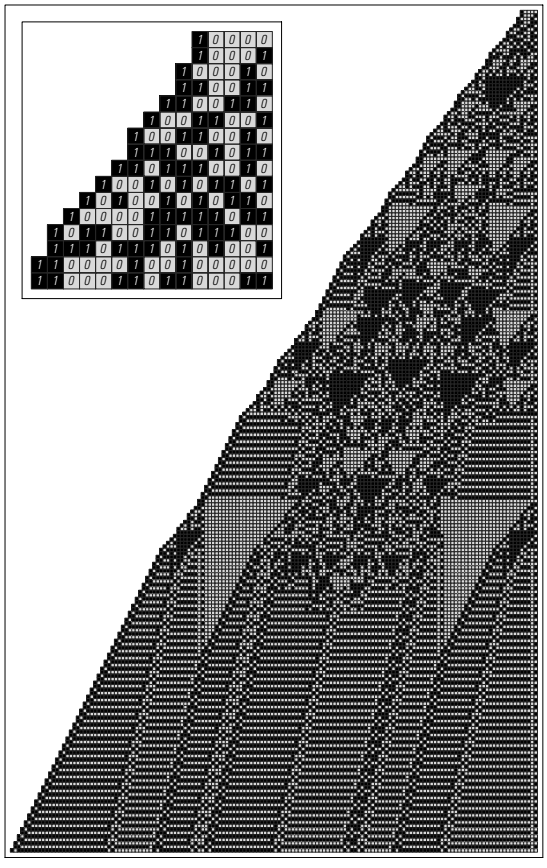
One knows from hand calculation that even an operation such as addition can lead to "carry" digits which propagate arbitrarily far to the left. And in fact most simple arithmetic operations have the property

that a digit which appears at a particular position in their result can depend on digits that were originally far away from it.

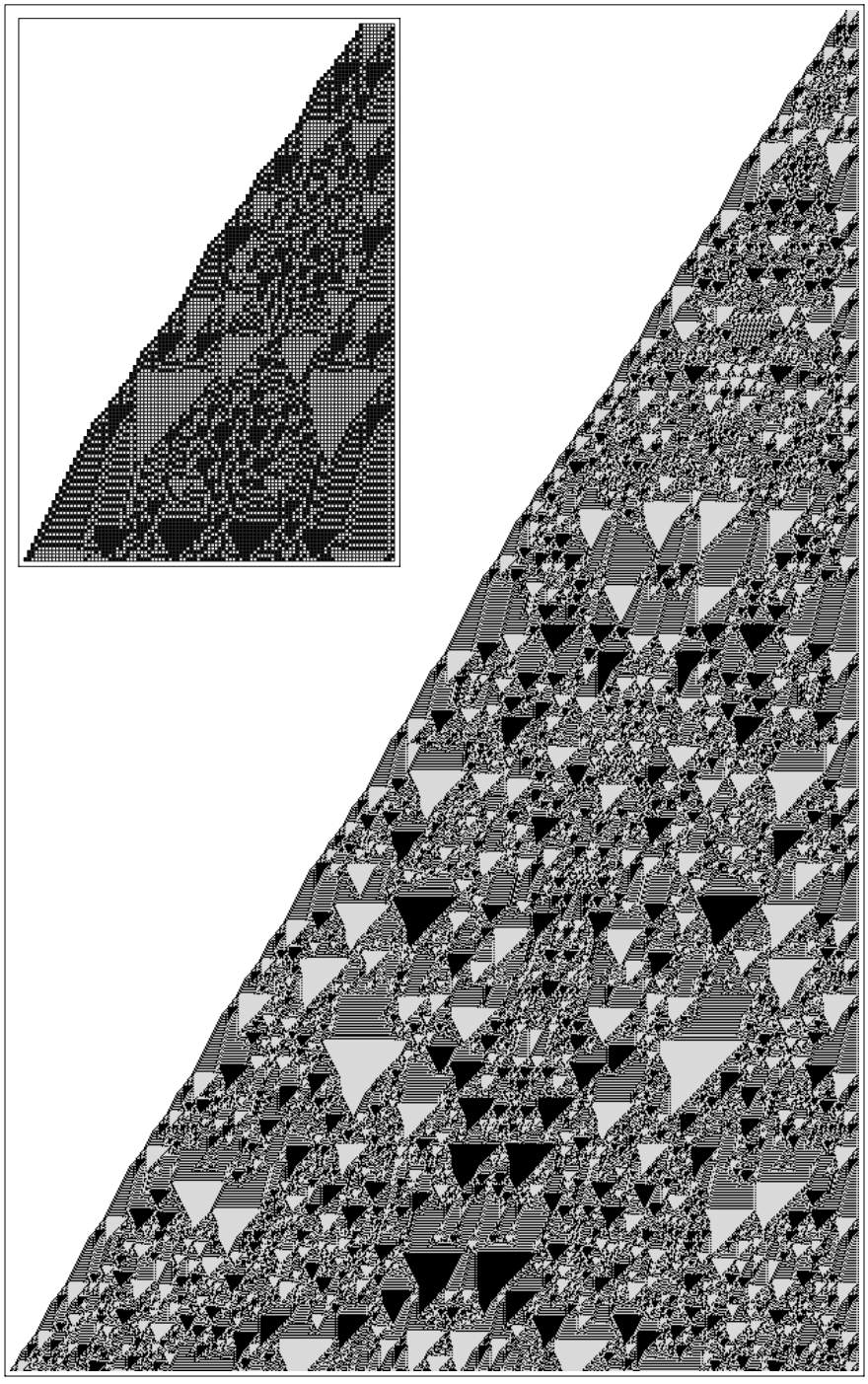
But despite fundamental differences like this in underlying rules, the overall behavior produced by systems based on numbers is still very similar to what one sees for example in cellular automata.

So just like for the various kinds of programs that we discussed in the previous chapter, the details of underlying rules again do not seem to have a crucial effect on the kinds of behavior that can occur.

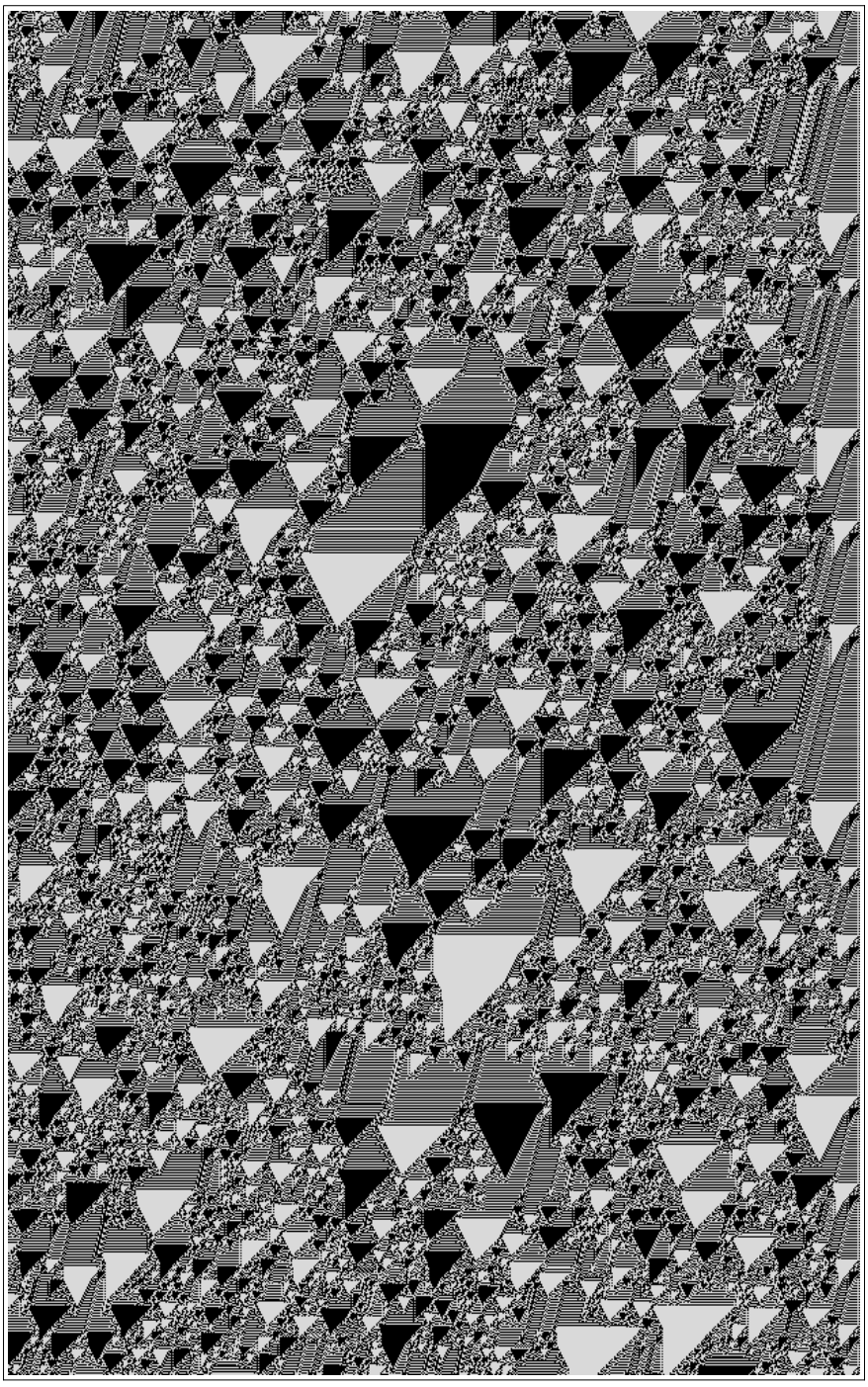
Indeed, despite the lack of locality in their underlying rules, the pictures below and on the pages that follow show that it is even possible to find systems based on numbers that exhibit something like the localized structures that we saw in cellular automata on page 32.



An example of a system defined by the following rule: at each step, take the number obtained at that step and write its base 2 digits in reverse order, then add the resulting number to the original one. For many possible starting numbers, the behavior obtained is very simple. This picture shows what happens when one starts with the number 16. After 180 steps, it turns out that all that survives are a few objects that one can view as localized structures.



A thousand steps in the evolution of a system with the same rule as on the previous page, but now starting with the number 512. Localized structures are visible, but the overall pattern never seems to take on any kind of simple repetitive form.



Continuation of the pattern on the facing page, starting at the millionth step. The picture shows the right-hand edge of the pattern; the complete pattern extends about 700 times the width of the page to the left.

Recursive Sequences

In the previous section, we saw that it is possible to get behavior of considerable complexity just by applying a variety of operations based on simple arithmetic. In this section what I will show is that with the appropriate setup just addition and subtraction turn out to be in a sense the only operations that one needs.

The basic idea is to consider a sequence of numbers in which there is a definite rule for getting the next number in the sequence from previous ones. It is convenient to refer to the first number in each sequence as $f[1]$, the second as $f[2]$, and so on, so that the n^{th} number is denoted $f[n]$. And with this notation, what the rule does is to specify how $f[n]$ should be calculated from previous numbers in the sequence.

In the simplest cases, $f[n]$ depends only on the number immediately before it in the sequence, denoted $f[n-1]$. But it is also possible to set up rules in which $f[n]$ depends not only on $f[n-1]$, but also on $f[n-2]$, as well as on numbers still earlier in the sequence.

The table below gives results obtained with a few specific rules. In all the cases shown, these results are quite simple, consisting of sequences that increase uniformly or fluctuate in a purely repetitive way.

`f[n] = 1 + f[n - 1], f[1] = 1`

(a) 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 ...

`f[n] = 1 - f[n - 1], f[1] = 1`

(b) 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 ...

`f[n] = 2 f[n - 1], f[1] = 1`

(c) 1 2 4 8 16 32 64 128 256 512 1024 2048 4096 8192 16384 32768 65536 131072 262144 524288 1048576 2097152 ...

`f[n] = f[n - 1] + f[n - 2], f[1] = 1, f[2] = 1`

(d) 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 ...

`f[n] = f[n - 1] - f[n - 2], f[1] = 1, f[2] = 1`

(e) 1 1 0 -1 -1 0 1 1 0 -1 -1 0 1 1 0 -1 -1 0 1 1 0 -1 -1 0 1 1 0 -1 -1 0 1 1 0 -1 -1 0 1 1 0 -1 -1 0 1 1 ...

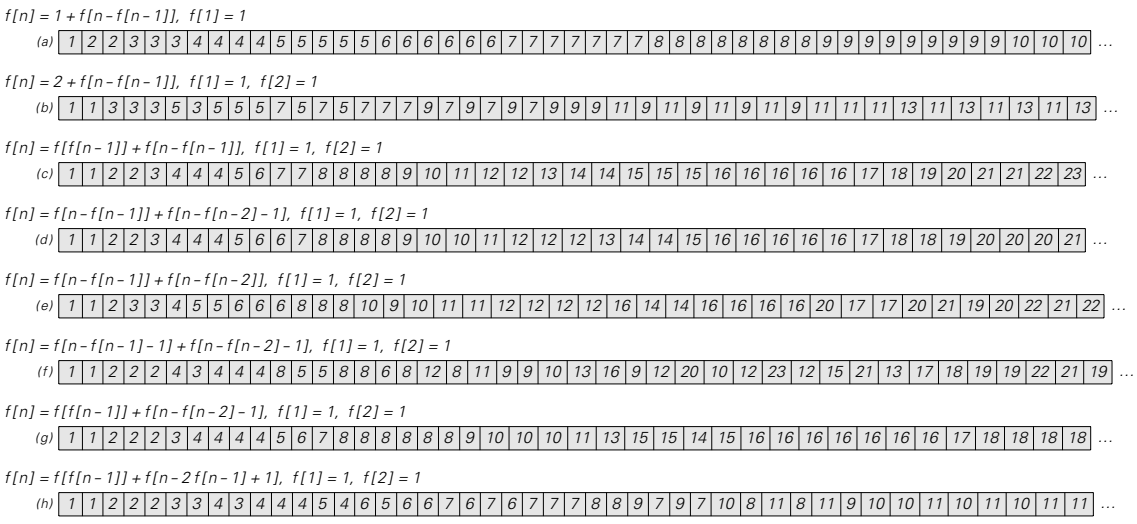
`f[n] = -f[n - 1] + f[n - 2], f[1] = 1, f[2] = 1`

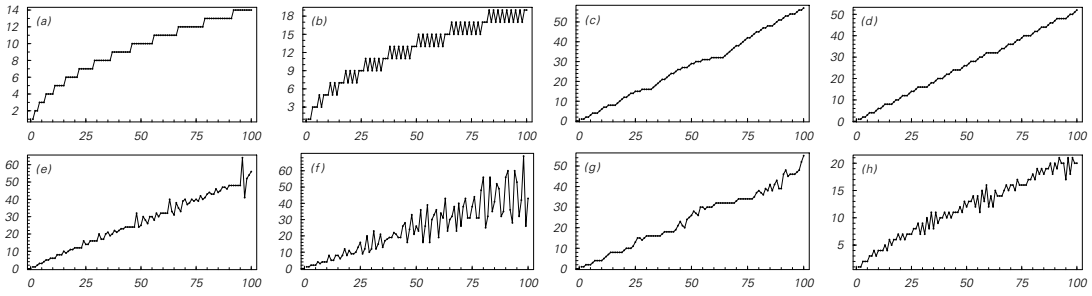
(f) 1 1 0 1 -1 2 -3 5 -8 13 -21 34 -55 89 -144 233 -377 610 -987 1597 -2584 4181 -6765 10946 -17711 28657 -46368 ...

Examples of some simple recursive sequences. The n^{th} element in each sequence is denoted `f[n]`, and the rule specifies how this element is determined from previous ones. With all the rules shown here, successive elements either increase smoothly or fluctuate in a purely repetitive way. Sequence (c) is the powers of two; (d) is the so-called Fibonacci sequence, related to powers of the golden ratio $(1+\sqrt{5})/2 \simeq 1.618$. All rules of the kind shown here lead to sequences where `f[n]` can be expressed in terms of a simple sum of powers of the form a^n.

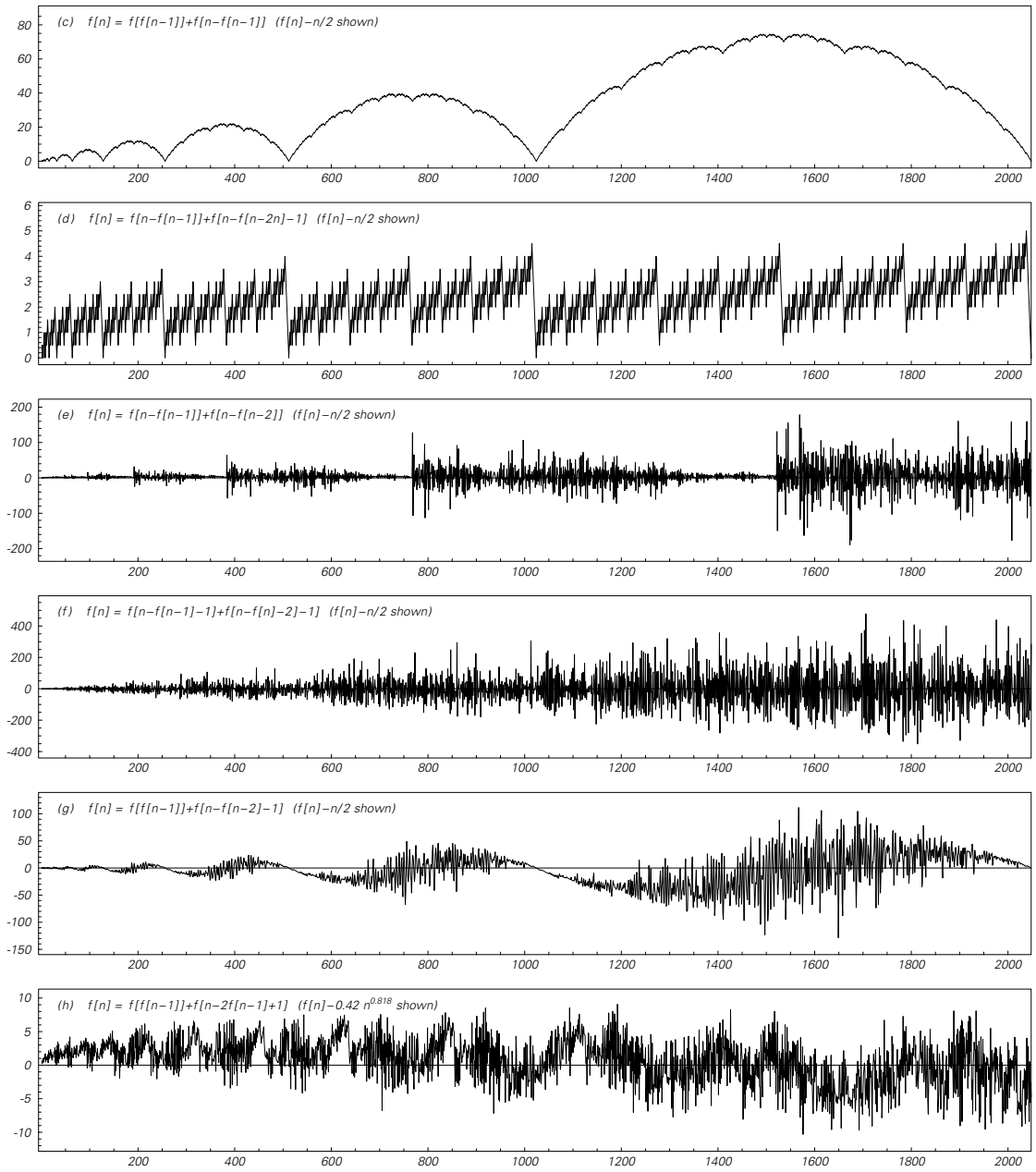
But it turns out that with slightly more complicated rules it is possible to get much more complicated behavior. The key idea is to consider rules which look at numbers that are not just a fixed distance back in the sequence. And what this means is that instead of depending only on quantities like $f[n-1]$ and $f[n-2]$, the rule for $f[n]$ can also for example depend on a quantity like $f[n-f[n-1]]$.

There is some subtlety here because in the abstract nothing guarantees that $n-f[n-1]$ will necessarily be a positive number. And if it is not, then results obtained by applying the rule can involve meaningless quantities such as $f[0]$, $f[-1]$ and $f[-2]$.





Examples of sequences generated by rules that do not depend only on elements a fixed distance back. Most such rules eventually end up involving meaningless quantities such as *f[0]* and *f[-1]*, but the particular rules shown here all avoid this problem.



Fluctuations in the overall increase of sequences from the previous page. In cases (c) and (d), the fluctuations have a regular nested form, and turn out to be directly related to the base 2 digit sequence of n. In the other cases, the fluctuations are more complicated, and seem in many respects random. All the rules shown start with $f[1] = f[2] = 1$.

For the vast majority of rules written down at random, such problems do indeed occur. But it is possible to find rules in which they do not, and the pictures on the previous two pages show a few examples I have found of such rules. In cases (a) and (b), the behavior is fairly simple. But in the other cases, it is considerably more complicated.

There is a steady overall increase, but superimposed on this increase are fluctuations, as shown in the pictures on the facing page.

In cases (c) and (d), these fluctuations turn out to have a very regular nested form. But in the other cases, the fluctuations seem instead in many respects random. Thus in case (f), for example, the number of positive and negative fluctuations appears on average to be equal even after a million steps.

But in a sense one of the most surprising features of the facing page is that the fluctuations it shows are so violent. One might have thought that in going say from $f[2000]$ to $f[2001]$ there would only ever be a small change. After all, between $n = 2000$ and 2001 there is only a 0.05% change in the size of n.

But much as we saw in the previous section it turns out that it is not so much the size of n that seems to matter as various aspects of its representation. And indeed, in cases (c) and (d), for example, it so happens that there is a direct relationship between the fluctuations in $f[n]$ and the base 2 digit sequence of n.

In case (d), the fluctuation in each $f[n]$ turns out to be essentially just the number of 1's that occur in the base 2 digit sequence for n. And in case (c), the fluctuations are determined by the total number of 1's that occur in the digit sequences of all numbers less than n.

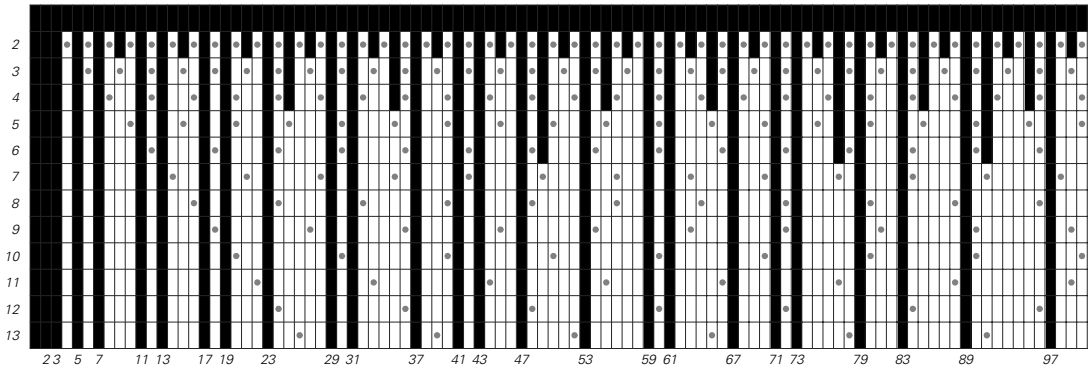
There are no such simple relationships for the other rules shown on the facing page. But in general one suspects that all these rules can be thought of as being like simple computer programs that take some representation of n as their input.

And what we have discovered in this section is that even though the rules ultimately involve only addition and subtraction, they nevertheless correspond to programs that are capable of producing behavior of great complexity.

The Sequence of Primes

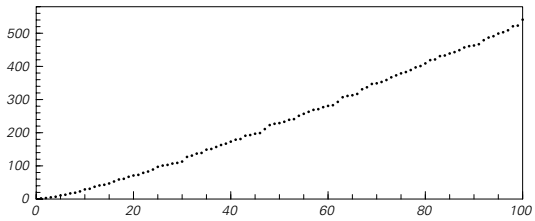
In the sequence of all possible numbers 1, 2, 3, 4, 5, 6, 7, 8, … most are divisible by others—so that for example 6 is divisible by 2 and 3. But this is not true of every number. And so for example 5 and 7 are not divisible by any other numbers (except trivially by 1). And in fact it has been known for more than two thousand years that there are an infinite sequence of so-called prime numbers which are not divisible by other numbers, the first few being 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, …

The picture below shows a simple rule by which such primes can be obtained. The idea is to start out on the top line with all possible numbers. Then on the second line, one removes all numbers larger than 2 that are divisible by 2. On the third line one removes numbers divisible by 3, and so on. As one goes on, fewer and fewer numbers remain. But some numbers always remain, and these numbers are exactly the primes.

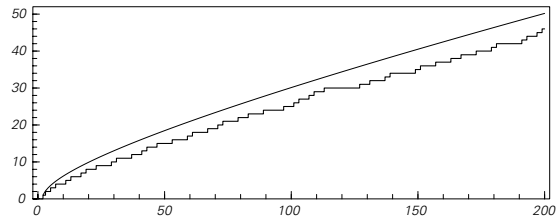


A filtering process that yields the prime numbers. One starts on the top line with all numbers between 1 and 100. Then on the second line, one removes numbers larger than 2 that are divisible by 2—as indicated by the gray dots. On the third line, one removes numbers larger than 3 that are divisible by 3. If one then continues forever, there are some numbers that always remain, and these are exactly the primes. The process shown is essentially the sieve of Eratosthenes, already known in 200 BC.

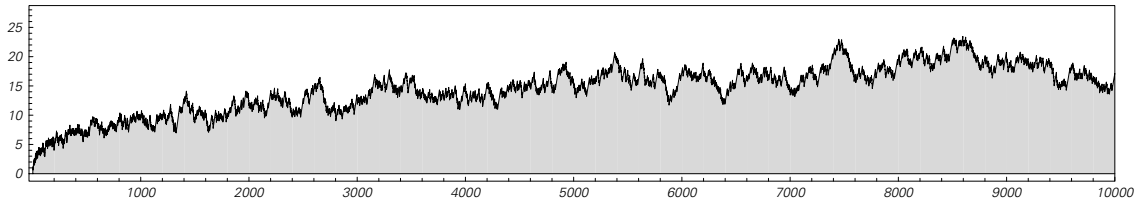
Given the simplicity of this rule, one might imagine that the sequence of primes it generates would also be correspondingly simple. But just as in so many other examples in this book, in fact it is not. And indeed the plots on the facing page show various features of this sequence which indicate that it is in many respects quite random.



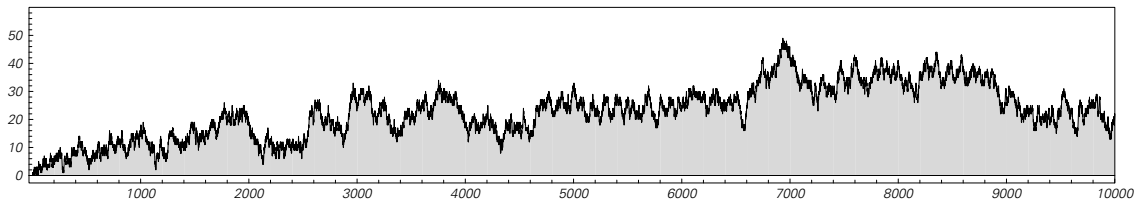
(a) The sequence of primes (Prime[n])



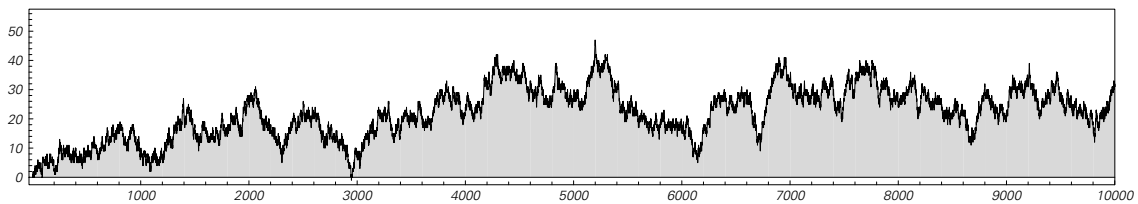
(b) The number of primes smaller than n (PrimePi[n]), together with the estimate LogIntegral[n]



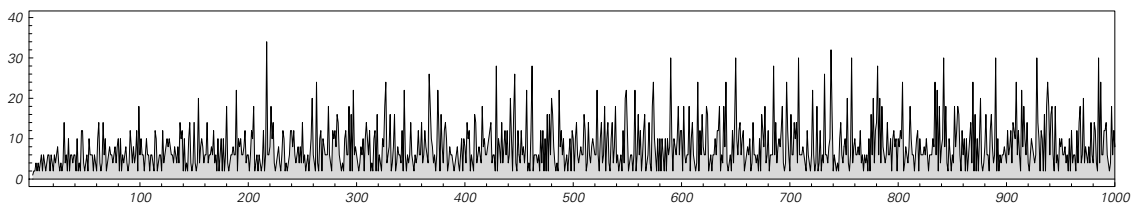
(c) The difference LogIntegral[n] – PrimePi[n]



(d) The excess of primes of the form 3 k – 1 over ones of the form 3 k + 1



(e) The excess of primes of the form 4 k – 1 over ones of the form 4 k + 1



(f) Gaps between successive primes

Features of the sequence of primes. Despite the simplicity of the rule on the facing page that generates the primes, the actual sequence of primes that is obtained seems in many respects remarkably random.

The examples of complexity that I have shown so far in this book are almost all completely new. But the first few hundred primes were no doubt known even in antiquity, and it must have been evident that there was at least some complexity in their distribution.

However, without the whole intellectual structure that I have developed in this book, the implications of this observation—and its potential connection, for example, to phenomena in nature—were not recognized. And even though there has been a vast amount of mathematical work done on the sequence of primes over the course of many centuries, almost without exception it has been concerned not with basic issues of complexity but instead with trying to find specific kinds of regularities.

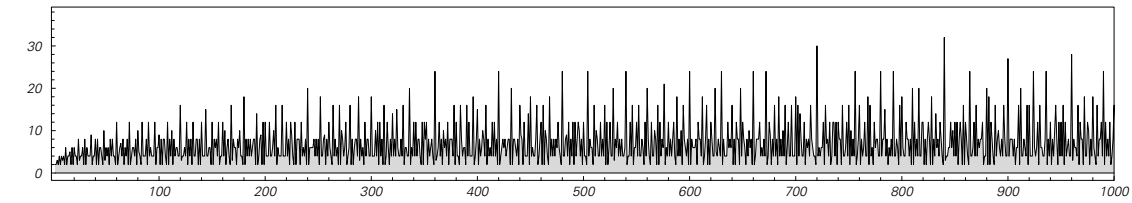
Yet as it turns out, few regularities have in fact been found, and often the results that have been established tend only to support the idea that the sequence has many features of randomness. And so, as one example, it might appear from the pictures on the previous page that (c), (d) and (e) always stay systematically above the axis. But in fact with considerable effort it has been proved that all of them are in a sense more random—and eventually cross the axis an infinite number of times, and indeed go any distance up or down.

So is the complexity that we have seen in the sequence of primes somehow unusual among sequences based on numbers? The pictures on the facing page show a few other examples of sequences generated according to simple rules based on properties of numbers.

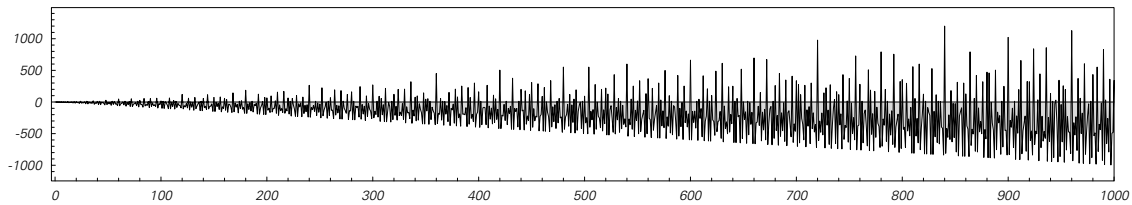
And in each case we again see a remarkable level of complexity.

Some of this complexity can be understood if we look at each number not in terms of its overall size, but rather in terms of its digit sequence or set of possible divisors. But in most cases—often despite centuries of work in number theory—considerable complexity remains.

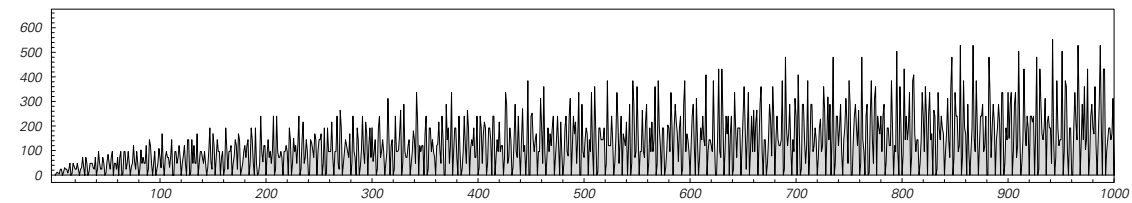
And indeed the only reasonable conclusion seems to be that just as in so many other systems in this book, such sequences of numbers exhibit complexity that somehow arises as a fundamental consequence of the rules by which the sequences are generated.



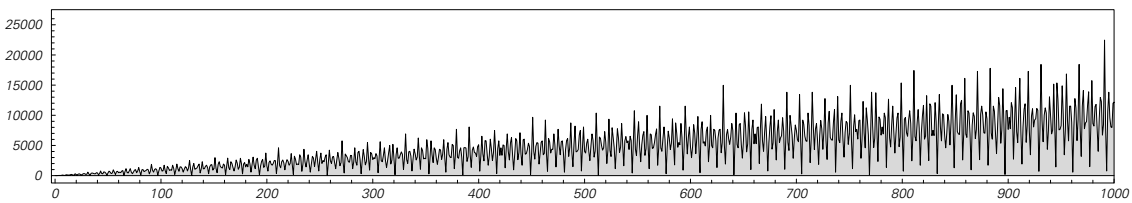
(a) The number of divisors of n (including n)



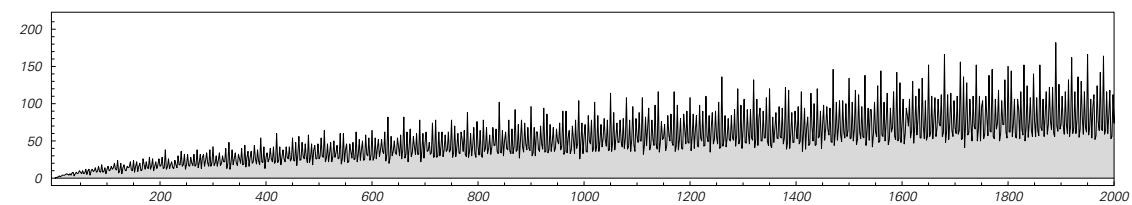
(b) The sum of the divisors of n (excluding n) minus n



(c) The number of ways of expressing n as a sum of three squares



(d) The number of ways of expressing n as a sum of four squares



(e) The number of ways of expressing an even number n as the sum of two primes

Sequences based on various simple properties of numbers. Extensive work in number theory has managed to establish only a few properties of these. It is for example known that (d) never reaches zero, while curve (c) reaches zero only for numbers of the form $4^r(8s+7)$. Sequence (b) is zero at so-called perfect numbers. Even perfect numbers always have a known form, but whether any odd perfect numbers exist is a question that has remained unresolved for more than two thousand years. The claim that sequence (e) never reaches zero is known as Goldbach's Conjecture. It was made in 1742 but no proof or counterexample has ever been found.

Mathematical Constants

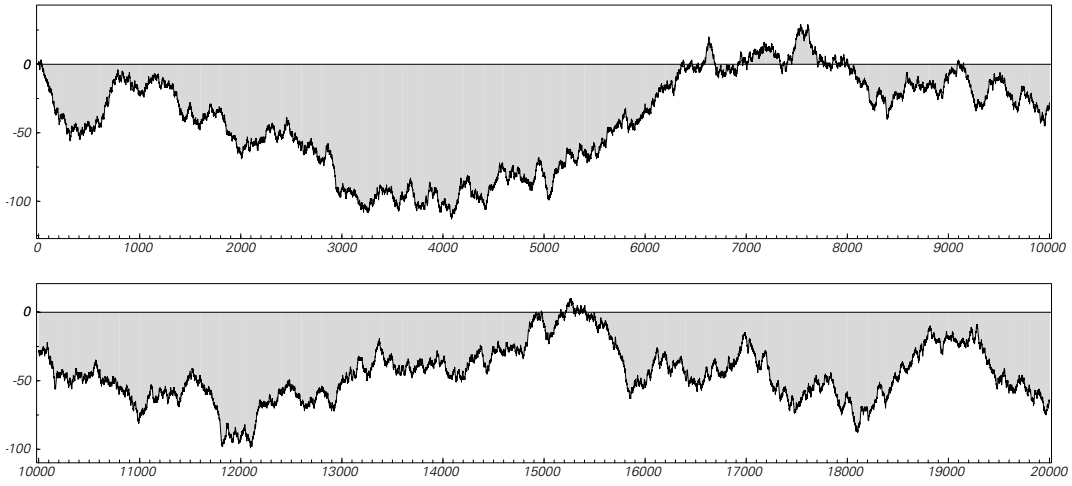
The last few sections have shown that one can set up all sorts of systems based on numbers in which great complexity can occur. But it turns out that the possibility of such complexity is already suggested by some well-known facts in elementary mathematics.

The facts in question concern the sequences of digits in numbers like π (pi). To a very rough approximation, π is 3.14. A more accurate approximation is 3.14159265358979323846264338327950288.

But how does this sequence of digits continue?

One might suppose that at some level it must be quite simple and regular. For the value of π is specified by the simple definition of being the ratio of the circumference of any circle to its diameter.

But it turns out that even though this definition is simple, the digit sequence of π is not simple at all. The facing page shows the first 4000 digits in the sequence, both in the usual case of base 10, and in base 2. And the picture below shows a pictorial representation of the first 20,000 digits in the sequence.



A pictorial representation of the first 20,000 digits of π in base 2. The curve drawn goes up every time a digit is 1, and down every time it is 0. Great complexity is evident. If the curve were continued further, it would spend more time above the axis, and no aspect of what is seen provides any evidence that the digit sequence is anything but perfectly random.

3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706798214808651328230664709384
46095505822317253594081284811174502841027019385211055596446229489549303819644288109756659334461284756482337867831652712019091
45648566923460348610454326648213393607260249141273724587006606315588174881520920962829254091715364367892590360011330530548820
46652138414695194151160943305727036575959195309218611738193261179310511854807446237996274956735188575272489122793818301194912
98336733624406566430860213949463952247371907021798609437027705392171762931767523846748184676694051320005681271452635608277857
71342757789609173637178721468440901224953430146549585371050792279689258923542019956112129021960864034418159813629774771309960
51870721134999999837297804995105973173281609631859502445945534690830264252230825334468503526193118817101000313783875288658753
32083814206171776691473035982534904287554687311595628638823537875937519577818577805321712268066130019278766111959092164201989
38095257201065485863278865936153381827968230301952035301852968995773622599413891249721775283479131515574857242454150695950829
53311686172785588907509838175463746493931925506040092770167113900984882401285836160356370766010471018194295559619894676783744
94482553797747268471040475346462080466842590694912933136770289891521047521620569660240580381501935112533824300355876402474964
73263914199272604269922796782354781636009341721641219924586315030286182974555706749838505494588586926995690927210797509302955
32116534498720275596023648066549911988183479775356636980742654252786255181841757467289097777279380008164706001614524919217321
72147723501414419735685481613611573525521334757418494684385233239073941433345477624168625189835694855620992192221842725502542
56887671790494601653466804988627232791786085784383827967976681454100953883786360950680064225125205117392984896084128488626945
60424196528502221066118630674427862203919494504712371378696095636437191728746776465757396241389086583264599581339047802759009
94657640789512694683983525957098258226205224894077267194782684826014769909026401363944374553050682034962524517493996514314298
09190659250937221696461515709858387410597885959772975498930161753928468138268683868942774155991855925245953959431049972524680
84598727364469584865383673622262609912460805124388439045124413654976278079771569143599770012961608944169486855584840635342207
22258284886481584560285060168427394522674676788952521385225499546667278239864565961163548862305774564980355936345681743241125
15076069479451096596094025228879710893145669136867228748940560101503308617928680920874760917824938589009714909675985261365549
78189312978482168299894872265880485756401427047755513237964145152374623436454285844479526586782105114135473573952311342716610
21359695362314429524849371871101457654035902799344037420073105785390621983874478084784896833214457138687519435064302184531910
48481005370614680674919278191197939952061419663428754440643745123718192179998391015919561814675142691239748940907186494231961
56794520809514655022523160388193014209376213785595663893778708303906979207734672218256259966150142150306803844773454920260541
46659252014974428507325186660021324340881907104863317346496514539057962685610055081066587969981635747363840525714591028970641
40110971206280439039759515677157700420337869936007230558763176359421873125147120532928191826186125867321579198414848829164470
60957527069572209175671167229109816909152801735067127485832228718352093539657251210835791513698820914442100675103346711031412
67111369908658516398315019701651511685171437657618351556508849099898599823873455283316355076479185358932261854896321329330898
57064204675259070915481416549859461637180270981994309924488957571282890592323326097299712084433573265489382391193259746366730
58360414281388303203824903758985243744170291327656180937734440307074692112019130203303801976211011004492932151608424448596376
69838952286847831235526582131449576857262433441893039686426243410773226978028073189154411010446823252716201052652272111660397

11.00100100001111110110101010001000100001011010001100001000110100110001001100011001100010100010111000000011011100000111001101000
10010100100000001001001110000010001000101001100111110011000111010000000001000001011101111101010011000111011000100111001101100 10
00100101000101001010000010000111100110001110001101000000001001101110111101111100101010001100110110011110011010011101001000001 10
00110110011000000010101100001010011011011111001001011111000101000011011101001111111000010011010101101101011011010101000111100 00
10010001011110010010000101101101010111011001100010010111100111111011000110111101000100110001000010111010011010011000110111111 1
01101011010110000101111111111010111001011011011110100000001101011011111101101111011100011100001101011111110110101101010100010 01
10011111101001011010111010011111001001000001000101111100010010110001111111100110010010010010100001100110010100011110110011100
10001011011001111011100001000000000011111001011100010100001011000111011111110000010110011000110110100100100000110110000111000 1
01010111010011100110100110100100010110001111111010100011111101001001001100111101011111100000110110010101011101001000111101111 0
01010001110101101100101100001110001100010111100110101011000100000100001010101001010111011100111101101010100101001000001110111 1
00001001011010010110011011010110011100001100001101010100111001001010101111001001100000000010011110001011101000110110000001000 1
10010100001100000010000101111100001100101001000000101111001000110001011100011011011001110001110111110001110011110011101110010 11
00000110000000111010000110000000111001101100100111100000111010001011101100000001111010001010001111101101011100010101011101111
10000011011110100110001010010110010011101111000101011110010111111011010010101010110000001011100011000001110011001010101010010 01
01111100111010101001010101101010111001010001010111010010001001100001100010011000111110100000010100010000000010101011100101000 1
11001011010100010101010101011000100001011011010110100110011000101110000110100000100010100000111101000110011101010000101010100
10000110101011110111110001110010111010011001001110110011111011100001010000010001011000110110111110111100001010100010101110101
00111000101010111010111010000011000001100011111011011001110010111000011111000010110100110111000011110010011000111101010111111
01011010111010001100110110110000100100110011110101110001111010001100100101001110000000100101000100101011000011001110111001110 1
11000111101001000100110000110101101001011101110011010111111000100101111111110100000011011011001100010100000100001100100110110
00011101100000001001110011001111101100100001101010011001000101001000011111001010110001100000010111011110110010000000000110010 1
11011111100001000101110101011101111010011000010101110101101100011101110000100110001000110000001011101011011001010001101110001 0
00001000111000100100111110100000001110100111001011010101100110001010000111101101101011011111111001110000011111101000100001000 1
11001001011100000101101000100100000101010010010000100001000000000001000110100111001000111100000100101010011110000111111001101 1
01011110001000011100011001101000010000101111011011101001011011001001101001100111000011001001110001100001101010111101001110001
00011110000011010100101000110100000110100101101100001010100001011110110100010010110000011111010011100101000101010110101000100
11001110100011011011101111011110000101101101100000100110111101000111011111001001011101000111011111100000101000001111110111110 1
10010101010011000101000011111000101100101000111010011100110101111000000010111011001100110110010100101100100111101000001001001 00
00110000111010001000100011001110111010000110000110010100010101101111110011111101101000111110110000100101001011100001100111011 1
00010110101111010111110111000000011011110111010111011000100001011100000010010000000111001101000000000011010010001001001111101010 1
10110000001011010101010011001001110110100111010101001100010001101100011111101110111000001100001101111111101101111101110010010
00010100110110000000100011110100110111110100001101011100100100110100000000010100001001001001000110110110000111111110101011010011

The first 4000 digits of π in bases 10 and 2. Despite the simple definition of π as the ratio of the circumference to the diameter of a circle, its digit sequence is sufficiently complicated as to seem for practical purposes random.

In no case are there any obvious regularities. Indeed, in all the more than two hundred billion digits of π that have so far been computed, no significant regularity of any kind has ever been found. Despite the simplicity of its definition, the digit sequence of π seems for practical purposes completely random.

But what about other numbers? Is π a special case, or are there other familiar mathematical constants that have complicated digit sequences? There are some numbers whose digit sequences effectively have limited length. Thus, for example, the digit sequence of 3/8 in base 10 is 0.375. (Strictly, the digit sequence is 0.3750000000…, but the 0's do not affect the value of the number, so are normally suppressed.)

It is however easy to find numbers whose digit sequences do not terminate. Thus, for example, the exact value of 1/3 in base 10 is 0.3333333333333…, where the 3's repeat forever. And similarly, 1/7 is 0.142857142857142857142857142857…, where now the block of digits 142857 repeats forever. The table below gives the digit sequences for several rational numbers obtained by dividing pairs of whole numbers. In all cases what we see is that the digit sequences of such numbers have a simple repetitive form. And in fact, it turns out that absolutely all rational numbers have digit sequences that eventually repeat.

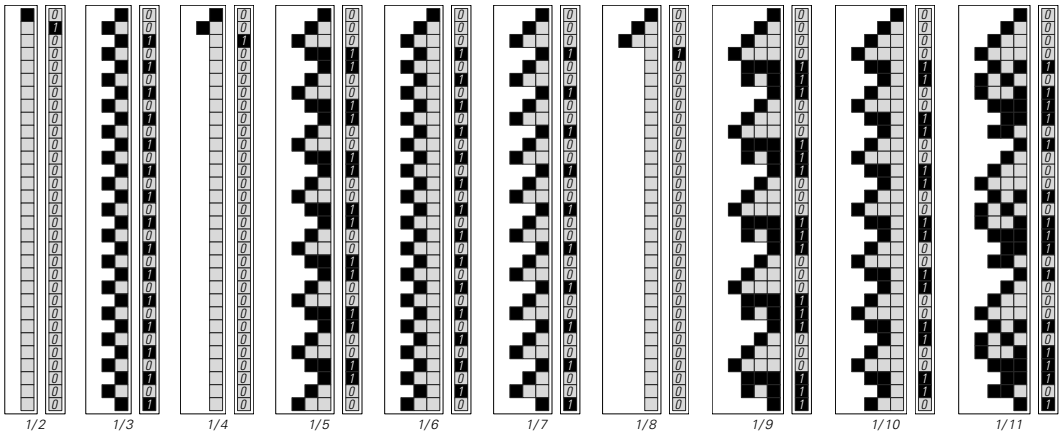
1/3 = 0.33333333333333333333333333333333333333333333333333333333333333333333333333333333…
1/7 = 0.14285714285714285714285714285714285714285714285714285714285714285714285714285142…
1/9 = 0.11111111111111111111111111111111111111111111111111111111111111111111111111111111…
1/11 = 0.09090909090909090909090909090909090909090909090909090909090909090909090909090909…
1/81 = 0.01234567901234567901234567901234567901234567901234567901234567901234567901234567…

1/3 = 0.01010101010101010101010101010101010101010101010101010101010101010101010101010101…
1/7 = 0.00100100100100100100100100100100100100100100100100100100100100100100100100100100…
1/9 = 0.00011100011100011100011100011100011100011100011100011100011100011100011100011000…
1/11 = 0.00010111010001011101000101110100010111010001011101000101110100010111010001011010…
1/81 = 0.00000011001010010001011000011111100110101101110100111100000011001010010001011000…

Digit sequences for various rational numbers, given in base 10 (above) and base 2 (below). For a number of the form p/q, the digit sequence always repeats with a period of at most $q - 1$ steps.

We can get some understanding of why this is so by looking at the details of how processes for performing division work. The pictures

below show successive steps in a particular method for computing the base 2 digit sequence for the rational numbers p/q.



Successive steps in the computation of various rational numbers. In each case, the column on the right shows the sequence of base 2 digits in the number, while the box on the left shows the remainder at each of the steps in the computation.

The method is essentially standard long division, although it is somewhat simpler in base 2 than in the usual case of base 10. The idea is to have a number r which essentially keeps track of the remainder at each step in the division. One starts by setting r equal to p. Then at each step, one compares the values of $2\,r$ and q. If $2\,r$ is less than q, the digit generated at that step is 0, and r is replaced by $2\,r$. Otherwise, r is replaced by $2\,r - q$. With this procedure, the value of r is always less than q. And as a result, the digit sequence obtained always repeats at most every $q - 1$ steps.

It turns out, however, that rational numbers are very unusual in having such simple digit sequences. And indeed, if one looks for example at square roots the story is completely different.

Perfect squares such as $4 = 2\times 2$ and $9 = 3\times 3$ are specifically set up to have square roots that are just whole numbers. But as the table at the top of the next page shows, other square roots have much more complicated digit sequences. In fact, so far as one can tell, all whole numbers other than perfect squares have square roots whose digit sequences appear completely random.

$\sqrt{2}$ = 1.41421356237309504880168872420969807856967187537694807317667973799073247846210703 9...

$\sqrt{3}$ = 1.73205080756887729352744634150587236694280525381038062805580697945193301690880003 7...

$\sqrt{5}$ = 2.23606797749978969640917366873127623544061835961152572427089724541052092563780489 9...

$\sqrt{6}$ = 2.44948974278317809819728407470589139196594748065667012843269256725096037745731502 7...

$\sqrt{7}$ = 2.64575131106459059050161575363926042571025918308245018036833445920106882323028362 8...

$\sqrt{8}$ = 2.82842712474619009760337744841939615713934375075389614635335947598146495692421407 8...

$\sqrt{10}$ = 3.16227766016837933199889354443271853371955513932521682685750485279259443863923822 1...

$\sqrt{11}$ = 3.31662479035539984911493273667068668392708854558935359705868214611648464260904384 7...

$\sqrt{2}$ = 1.01101010000010011110011001100111111100111011110011001001000010001011001011111011 0...

$\sqrt{3}$ = 1.10111011011001111010111010000101100001001100101010100111001110110010010101110100 0...

$\sqrt{5}$ = 10.00111100011011101111001101110010111111101001010011111000001010111110011100111001...

$\sqrt{6}$ = 10.01110011000100011100001010000001001001000010010111001111101000000110010000110010...

$\sqrt{7}$ = 10.10100101010011111111101010011101001011111000111010011011011110001110011101010011 1...

$\sqrt{8}$ = 10.11010100000100111100110011001111111001110111100110010010000100010110010111110110...

$\sqrt{10}$ = 11.00101001100010110000011101011011010010110110101001010010000010010100010101111...

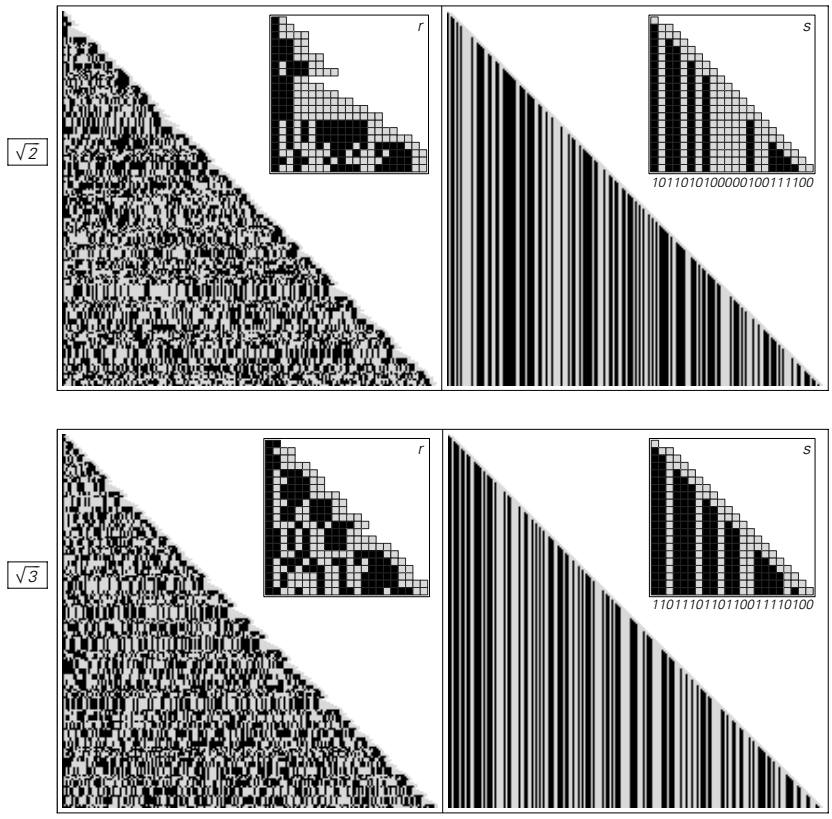
$\sqrt{11}$ = 11.01010001000011100101001001111111101011011110011010000010110100011101111001001001...

Digit sequences for various square roots, given at the top in base 10 and at the bottom in base 2. Despite their simple definition, all these sequences seem for practical purposes random.

But how is such randomness produced? The picture at the top of the facing page shows an example of a procedure for generating the base 2 digit sequence for the square root of a given number n.

The procedure is only slightly more complicated than the one for division discussed above. It involves two numbers r and s, which are initially set to be n and 0, respectively. At each step it compares the values of r and s, and if r is larger than s it replaces r and s by $4\,(r - s - 1)$ and $2\,(s + 2)$ respectively; otherwise it replaces them just by $4\,r$ and $2\,s$. And it then turns out that the base 2 digits of s correspond exactly to the base 2 digits of $\sqrt{n}$ —with one new digit being generated at each step.

As the picture shows, the results of the procedure exhibit considerable complexity. And indeed, it seems that just like so many other examples that we have discussed in this book, the procedure for generating square roots is based on simple rules but nevertheless yields behavior of great complexity.



A procedure for generating the digit sequences of square roots. Two numbers, r and s, are involved. To find $\sqrt{n}$ one starts by setting $r = n$ and $s = 0$. Then at each step one applies the rule $\{r, s\} \to If[r > s, \{4(r - s - 1), 2(s + 2)\}, \{4r, 2s\}]$. The result is that the digits of s in base 2 turn out to correspond exactly to the digits of $\sqrt{n}$. Note that if n is not between 1 and 4, it must be multiplied or divided by an appropriate power of 4 before starting this procedure.

It turns out that square roots are certainly not alone in having apparently random digit sequences. As an example, the table on the next page gives the digit sequences for some cube roots and fourth roots, as well as for some logarithms and exponentials. And so far as one can tell, almost all these kinds of numbers also have apparently random digit sequences.

In fact, rational numbers turn out to be the only kinds of numbers that have repetitive digit sequences. And at least in square roots, cube roots, and so on, it is known that no nested digit sequences

$\sqrt[3]{2} = 1.25992104989487316476721060727822835057025146470150798008197511215529967651395948372939656243625509415431025 60\ldots$

$\sqrt[3]{3} = 1.44224957030740838232163831078010958839186925349935057754641619454168759682999733985475547970564525668683508 08\ldots$

$\sqrt[4]{2} = 1.18920711500272106671749997056047591529297209246381741301900222471946666822691715987078134453813767371603739 47\ldots$

$\sqrt[4]{3} = 1.31607401295249246081921890179699905516006859020582217673192265859586679519730213305074315024660193152004774 23\ldots$

$\text{Log}[2] = 0.69314718055994530941723212145817656807550013436025525412068000949339362196969471560586332699641868754200148 10\ldots$

$\text{Log}[3] = 1.09861228866810969139524523692252570464749055782274945173469433363749429321860896687361575481373208878797002 90\ldots$

$e = 2.71828182845904523536028747135266249775724709369995957496696762772407663035354759457138217852516642742746639 19\ldots$

$e^2 = 7.38905609893065022723042746057500781318031557055184732408712782252257379607905776338431248507912179477375316 12\ldots$

$\sqrt[3]{2} = 1.01000010100010100010111110011000110101110010100010101110001000100011110111011010101101110001010110111110001 00\ldots$

$\sqrt[3]{3} = 1.01110001001101110100010010010001001000111110111011001011100110111011100111111100010110110001010110111000110\ldots$

$\sqrt[4]{2} = 1.00110000011011111110000010100011000110110111000101010010110111101000110101011010010001100011000001011100100 00\ldots$

$\sqrt[4]{3} = 1.01010000111010100011100111111100101111110001011001100101111101101101110000110011001111110001010000011010011 01\ldots$

$\text{Log}[2] = 0.10110001011100100001011111110111101000111001111011110011010101111001001111000111011001110011000000000111111 00\ldots$

$\text{Log}[3] = 1.00011001001111101010011110101010110100000011000010100101111011010100100000110011000110101010101000001010011 1\ldots$

$e = 10.10110111111000010101000101100010100010101110110100101010011010101011111101110001010110001000000010011100111 1\ldots$

$e^2 = 111.01100011100110010010111000110101001101110110101101110011000011001110100011101110100010000001101011011010001\ldots$

Digit sequences for cube roots, fourth roots, logarithms and exponentials, given at the top in base 10 and the bottom in base 2. Once again, these sequences seem for practical purposes random.

ever occur. It is straightforward to construct a nested digit sequence using for example the substitution systems on page 83, but the point is that such a digit sequence never corresponds to a number that can be obtained by the mathematical operation of taking roots.

So far in this chapter we have always used digit sequences as our way of representing numbers. But one might imagine that perhaps this representation is somehow perverse, and that if we were just to choose another one, then numbers generated by simple mathematical operations would no longer seem complex.

Any representation for a number can in a sense be thought of as specifying a procedure for constructing that number. Thus, for example, the pictures at the top of the facing page show how the base 10 and base 2 digit sequence representations of π can be used to construct the number π.

$$3.141592653\ldots = 3 + \frac{1}{10}\left(1 + \frac{1}{10}\left(4 + \frac{1}{10}\left(1 + \frac{1}{10}\left(5 + \frac{1}{10}\left(9 + \frac{1}{10}\left(2 + \frac{1}{10}\left(6 + \frac{1}{10}\left(5 + \frac{1}{10}\left(3 + \ldots\right)\right)\right)\right)\right)\right)\right)\right)$$

$$11.001001000\ldots = 2 + 1 + \frac{1}{2}\left(0 + \frac{1}{2}\left(0 + \frac{1}{2}\left(1 + \frac{1}{2}\left(0 + \frac{1}{2}\left(0 + \frac{1}{2}\left(1 + \frac{1}{2}\left(0 + \frac{1}{2}\left(0 + \frac{1}{2}\left(0 + \ldots\right)\right)\right)\right)\right)\right)\right)\right)$$

Procedures for building up π from its base 10 and base 2 digit sequence representations.

By replacing the addition and multiplication that appear above by other operations one can then get other representations for numbers. A common example are so-called continued fraction representations, in which the operations of addition and division are used, as shown below.

$$3 + 1/(7 + 1/(15 + 1/(1 + 1/(292 + 1/(1 + 1/(1 + 1/(1 + 1/(2 + 1/(1 + 1/(3 + 1/(1 + 1/(14 + \ldots)))))))))))$$

$$\{3, 7, 15, 1, 292, 1, 1, 1, 2, 1, 3, 1, 14, 2, 1, 1, 2, 2, 2, 2, 1, 84, 2, 1, 1, 15, 3, 13, 1, 4, 2, 6, 6, 99, 1, 2, 2, 6, 3, 5, 1, \ldots\}$$

The continued fraction representation of π. In this representation the value of π is built up by successive additions and divisions, rather than successive additions and multiplications.

The table on the next page gives the continued fraction representations for various numbers. In the case of rational numbers, the results are always of limited length. But for other numbers, they go on forever. Square roots turn out to have purely repetitive continued fraction representations. And the representations of $e \simeq 2.718$ and all its roots also show definite regularity. But for π, as well as for cube roots, fourth roots, and so on, the continued fraction representations one gets seem essentially random.

What about other representations of numbers? At some level, one can always use symbolic expressions like $\sqrt{2} + e^{\sqrt{3}}$ to represent numbers. And almost by definition, numbers that can be obtained by simple mathematical operations will correspond to simple such expressions. But the problem is that there is no telling how difficult it may be to compute the actual value of a number from the symbolic expression that is used to represent it.

And in thinking about representations of numbers, it seems appropriate to restrict oneself to cases where the effort required to find the value of a number from its representation is essentially the same for

$1/7 = \{0, 7]$
$7/11 = \{0, 1, 1, 1, 3]$
$\sqrt{2} = \{1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, \ldots\}$
$\sqrt{3} = \{1, 1, 2, 1, \ldots\}$
$\sqrt{5} = \{2, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, \ldots\}$
$\sqrt{7} = \{2, 1, 1, 1, 4, 1, 1, 1, 4, 1, 1, 1, 4, 1, 1, 1, 4, 1, 1, 1, 4, 1, 1, 1, 4, 1, 1, 1, 4, 1, 1, 1, 4, 1, 1, 1, 4, 1, 1, 1, 4, 1, 1, 1, 4, 1, 1, 1, 4, 1, 1, 1, 4, 1, \ldots\}$
$(1 + \sqrt{5})/2 = \{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, \ldots\}$
$\sqrt[3]{2} = \{1, 3, 1, 5, 1, 1, 4, 1, 1, 8, 1, 14, 1, 10, 2, 1, 4, 12, 2, 3, 2, 1, 3, 4, 1, 1, 2, 14, 3, 12, 1, 15, 3, 1, 4, 534, 1, 1, 5, 1, 1, 121, 1, 2, 2, 4, 10, 3, 2, 2, \ldots\}$
$\sqrt[3]{3} = \{1, 2, 3, 1, 4, 1, 5, 1, 1, 6, 2, 5, 8, 3, 3, 4, 2, 6, 4, 4, 1, 3, 2, 3, 4, 1, 4, 9, 1, 8, 4, 3, 1, 3, 2, 6, 1, 6, 1, 3, 1, 1, 1, 1, 12, 3, 1, 3, 1, 1, 4, 1, 6, 1, 5, \ldots\}$
$\sqrt[4]{2} = \{1, 5, 3, 1, 1, 40, 5, 1, 1, 25, 2, 3, 1, 6, 2, 1, 1, 2, 1, 2, 1, 1, 1, 2, 2, 1, 7, 2, 7, 1, 1, 1, 2, 1, 1, 32, 4, 1, 6, 2, 1, 1, 1, 15, 1, 5, 1, 4, 1, 1, 1, 3, 1, 3, \ldots\}$
$\sqrt[4]{3} = \{1, 3, 6, 9, 1, 1, 2, 1, 2, 1, 2, 5, 1, 12, 5, 1, 4, 1, 13, 1, 6, 1, 22, 1, 8, 21, 3, 142, 1, 1, 2, 1, 2, 2, 7, 1, 2, 1, 1, 1, 5, 3, 1, 1, 2, 1, 1, 3, 1, 1, 1, 1, \ldots\}$
$Log[2] = \{0, 1, 2, 3, 1, 6, 3, 1, 1, 2, 1, 1, 1, 1, 3, 10, 1, 1, 1, 2, 1, 1, 1, 1, 3, 2, 3, 1, 13, 7, 4, 1, 1, 1, 7, 2, 4, 1, 1, 2, 5, 14, 1, 10, 1, 4, 2, 18, 3, 1, 4, 1, 6, \ldots\}$
$Log[3] = \{1, 10, 7, 9, 2, 2, 1, 3, 1, 32, 2, 17, 1, 15, 1, 1, 7, 3, 1, 35, 1, 1, 1, 2, 5, 3, 2, 1, 4, 2, 1, 3, 1, 5, 3, 13, 1, 1, 1, 6, 2, 3, 1, 152, 1, 2, 3, 1, 7, 9, 2, \ldots\}$
$e = \{2, 1, 2, 1, 1, 4, 1, 1, 6, 1, 1, 8, 1, 1, 10, 1, 1, 12, 1, 1, 14, 1, 1, 16, 1, 1, 18, 1, 1, 20, 1, 1, 22, 1, 1, 24, 1, 1, 26, 1, 1, 28, 1, 1, 30, 1, 1, 32, 1, \ldots\}$
$\sqrt{e} = \{1, 1, 1, 1, 5, 1, 1, 9, 1, 1, 13, 1, 1, 17, 1, 1, 21, 1, 1, 25, 1, 1, 29, 1, 1, 33, 1, 1, 37, 1, 1, 41, 1, 1, 45, 1, 1, 49, 1, 1, 53, 1, 1, 57, 1, 1, 61, 1, 1, \ldots\}$
$\sqrt[3]{e} = \{1, 2, 1, 1, 8, 1, 1, 14, 1, 1, 20, 1, 1, 26, 1, 1, 32, 1, 1, 38, 1, 1, 44, 1, 1, 50, 1, 1, 56, 1, 1, 62, 1, 1, 68, 1, 1, 74, 1, 1, 80, 1, 1, 86, 1, 1, 92, 1, 1, \ldots\}$
$e^2 = \{7, 2, 1, 1, 3, 18, 5, 1, 1, 6, 30, 8, 1, 1, 9, 42, 11, 1, 1, 12, 54, 14, 1, 1, 15, 66, 17, 1, 1, 18, 78, 20, 1, 1, 21, 90, 23, 1, 1, 24, 102, 26, 1, 1, 27, \ldots\}$
$e^3 = \{20, 11, 1, 2, 4, 3, 1, 5, 1, 2, 16, 1, 1, 16, 2, 13, 14, 4, 6, 2, 1, 1, 2, 2, 2, 3, 5, 1, 3, 1, 1, 68, 7, 5, 1, 4, 2, 1, 1, 1, 1, 1, 7, 3, 1, 6, 1, 2, 5, 4, 7, \ldots\}$
$\pi = \{3, 7, 15, 1, 292, 1, 1, 1, 2, 1, 3, 1, 14, 2, 1, 1, 2, 2, 2, 2, 1, 84, 2, 1, 1, 15, 3, 13, 1, 4, 2, 6, 6, 99, 1, 2, 2, 6, 3, 5, 1, 1, 6, 8, 1, 7, 1, 2, 3, 7, 1, 2, \ldots\}$
$\pi^2 = \{9, 1, 6, 1, 2, 47, 1, 8, 1, 1, 2, 2, 1, 1, 8, 3, 1, 10, 5, 1, 3, 1, 2, 1, 1, 3, 15, 1, 1, 2, 2, 1, 3, 2, 7, 1, 9, 18, 30, 2, 145, 1, 1, 17, 9, 1, 1, 1, 1, 7, 12, 1, \ldots\}$
$Sinh[1] = \{1, 5, 1, 2, 2, 2, 1, 2, 7, 5, 1, 1, 1, 2, 2, 19, 1, 2, 1, 7, 1, 1, 9, 1, 3, 1, 1, 2, 1, 1, 1, 1, 1, 3, 1, 2, 4, 5, 3, 5, 1, 3, 1, 1, 1, 2, 7, 1, 9, 1, 1, 2, 1, 21, 1, \ldots\}$
$Tanh[1] = \{0, 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, \ldots\}$

Continued fraction representations for several numbers. Square roots yield repetitive sequences in this representation, but cube roots and higher roots yield seemingly random sequences.

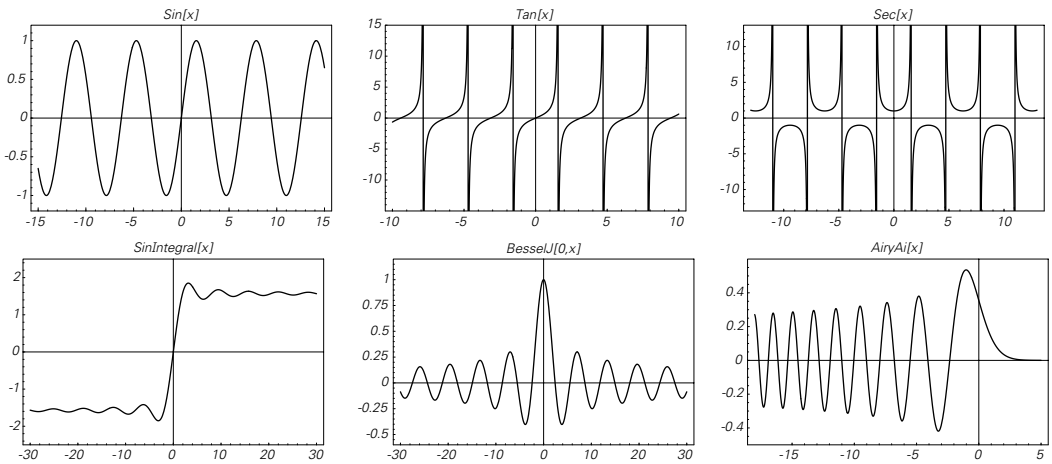
all numbers. If one does this, then the typical experience is that in any particular representation, some class of numbers will have simple forms. But other numbers, even though they may be the result of simple mathematical operations, tend to have seemingly random forms.

And from this it seems appropriate to conclude that numbers generated by simple mathematical operations are often in some intrinsic sense complex, independent of the particular representation that one uses to look at them.

Mathematical Functions

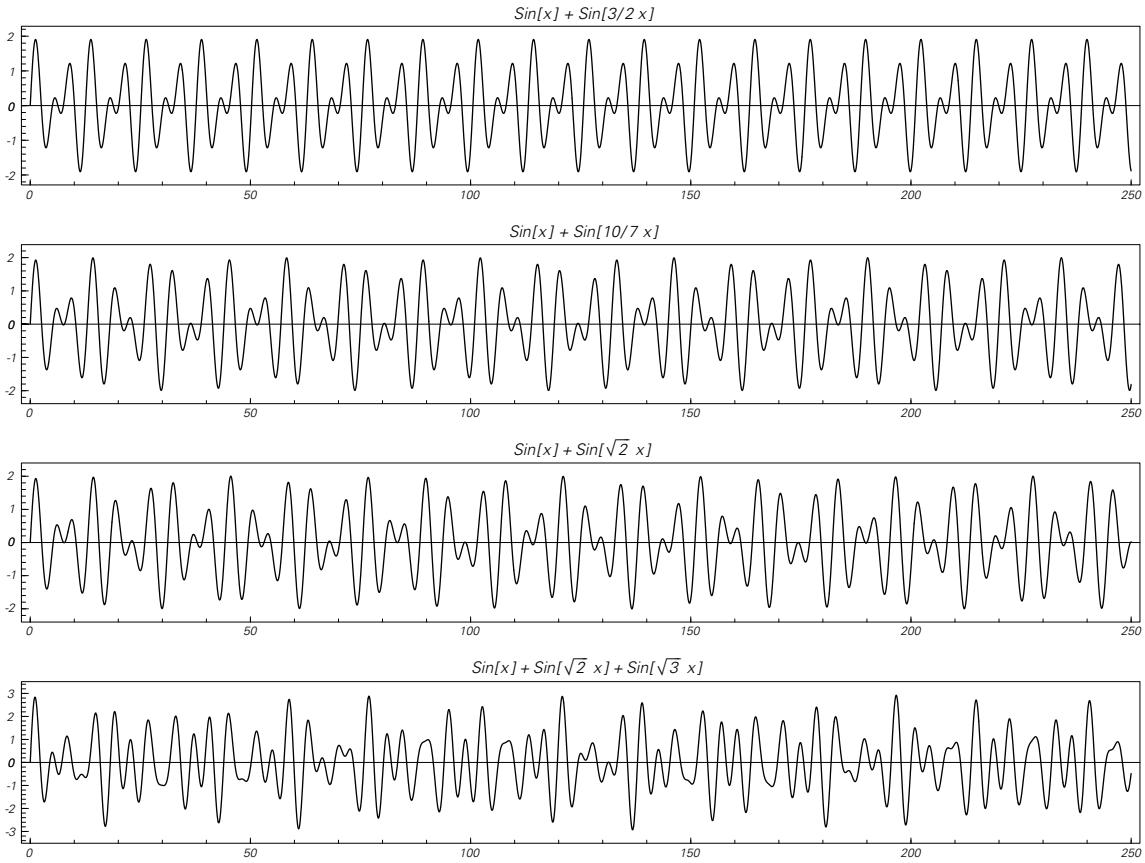
The last section showed that individual numbers obtained by applying various simple mathematical functions can have features that are quite complex. But what about the functions themselves?

The pictures below show curves obtained by plotting standard mathematical functions. All of these curves have fairly simple, essentially repetitive forms. And indeed it turns out that almost all the standard mathematical functions that are defined, for example, in *Mathematica*, yield similarly simple curves.



Plots of some standard mathematical functions. The top row shows three trigonometric functions. The bottom row shows three so-called special functions that are commonly encountered in mathematical physics and other areas of traditional science. In all cases the curves shown have fairly simple repetitive forms.

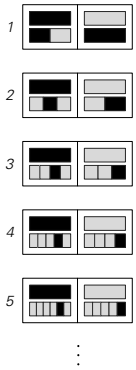
But if one looks at combinations of these standard functions, it is fairly easy to get more complicated results. The pictures on the next page show what happens, for example, if one adds together various sine functions. In the first picture, the curve one gets has a fairly simple repetitive structure. In the second picture, the curve is more complicated, but still has an overall repetitive structure. But in the third and fourth pictures, there is no such repetitive structure, and indeed the curves look in many respects random.

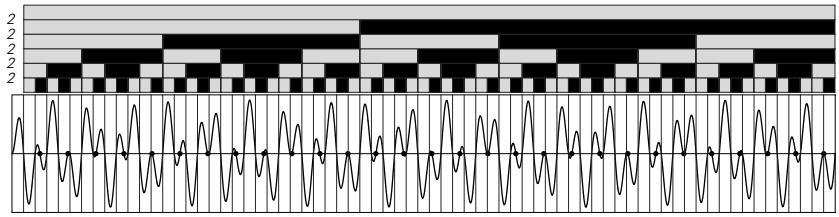


Curves obtained by adding together various sine functions. In the first two cases, the curves are ultimately repetitive; in the second two cases they are not. If viewed as waveforms for sounds, then these curves correspond to chords. The first curve yields a perfect fifth, while the third curve yields a diminished fifth (or tritone) in an equal temperament scale.

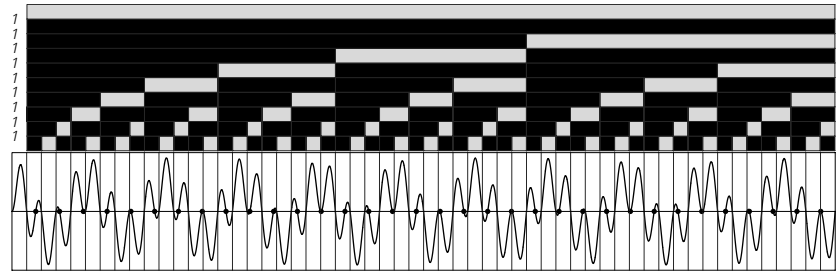
In the third picture, however, the points where the curve crosses the axis come in two regularly spaced families. And as the pictures on the facing page indicate, for any curve like $\text{Sin}[x] + \text{Sin}[\alpha\, x]$ the relative arrangements of these crossing points turn out to be related to the output of a generalized substitution system in which the rule at each step is obtained from a term in the continued fraction representation of $(\alpha - 1)/(\alpha + 1)$.

When α is a square root, then as discussed in the previous section, the continued fraction representation is purely repetitive,

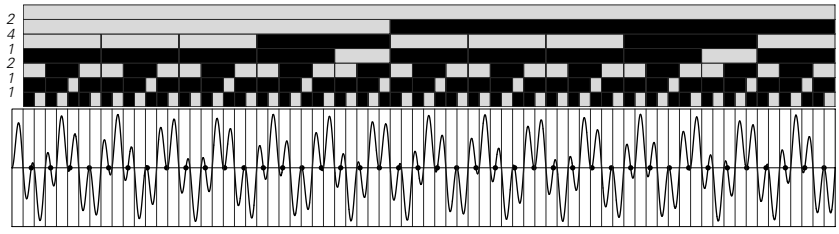




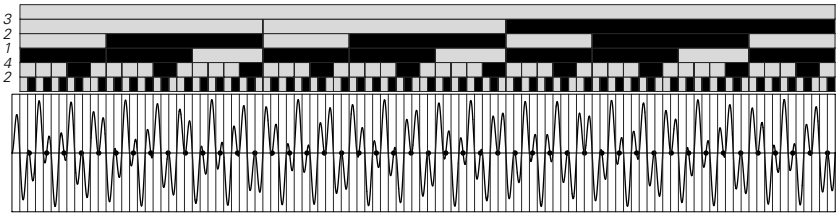
$Cos[x] - Cos[(1 + \sqrt{2})\,x]$



$Cos[x] - Cos[(2 + \sqrt{5})\,x]$



$Cos[x] - Cos[(2 + \sqrt[3]{5})\,x]$



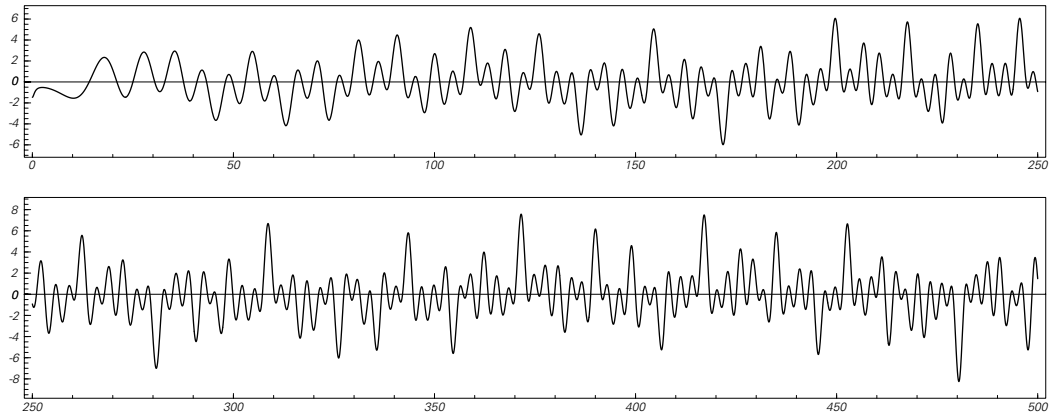
$Cos[x] - Cos[(1 + \sqrt{e})\,x]$

Curves obtained by adding or subtracting exactly two sine or cosine functions turn out to have a pattern of axis crossings that can be reproduced by a generalized substitution system. In general there is an axis crossing within an interval when the corresponding element in the generalized substitution system is black, and there is not when the element is white. In the case of $Cos[x] - Cos[\alpha\,x]$ each step in the generalized substitution system has a rule determined as shown on the left from a term in the continued fraction representation of $(\alpha - 1)/(\alpha + 1)$. In the first two examples shown α is a quadratic irrational, so that the continued fraction is repetitive, and the pattern obtained is purely nested. (The second example is analogous to the Fibonacci substitution system on page 83.) In the last two examples, however, there is no such regularity. Note that successive terms in each continued fraction are shown alongside successive steps in the substitution system going up the page.

making the generated pattern nested. But when α is not a square root the pattern can be more complicated. And if more than two sine functions are involved there no longer seems to be any particular connection to generalized substitution systems or continued fractions.

Among all the various mathematical functions defined, say, in *Mathematica* it turns out that there are also a few—not traditionally common in natural science—which yield complex curves but which do not appear to have any explicit dependence on representations of individual numbers. Many of these are related to the so-called Riemann zeta function, a version of which is shown in the picture below.

The basic definition of this function is fairly simple. But in the end the function turns out to be related to the distribution of primes—and the curve it generates is quite complicated. Indeed, despite immense mathematical effort for over a century, it has so far been impossible even to establish for example the so-called Riemann Hypothesis, which in effect just states that all the peaks in the curve lie above the axis, and all the valleys below.



A curve associated with the so-called Riemann zeta function. The zeta function *Zeta[s]* is defined as *Sum[1/k^s, {k, ∞}]*. The curve shown here is the so-called Riemann-Siegel Z function, which is essentially *Zeta[1/2 + i t]*. The celebrated Riemann Hypothesis in effect states that all peaks after the first one in this curve must lie above the axis.

Iterated Maps and the Chaos Phenomenon

The basic idea of an iterated map is to take a number between 0 and 1, and then in a sequence of steps to update this number according to a fixed rule or "map". Many of the maps I will consider can be expressed in terms of standard mathematical functions, but in general all that is needed is that the map take any possible number between 0 and 1 and yield some definite number that is also between 0 and 1.

The pictures on the next two pages show examples of behavior obtained with four different possible choices of maps.

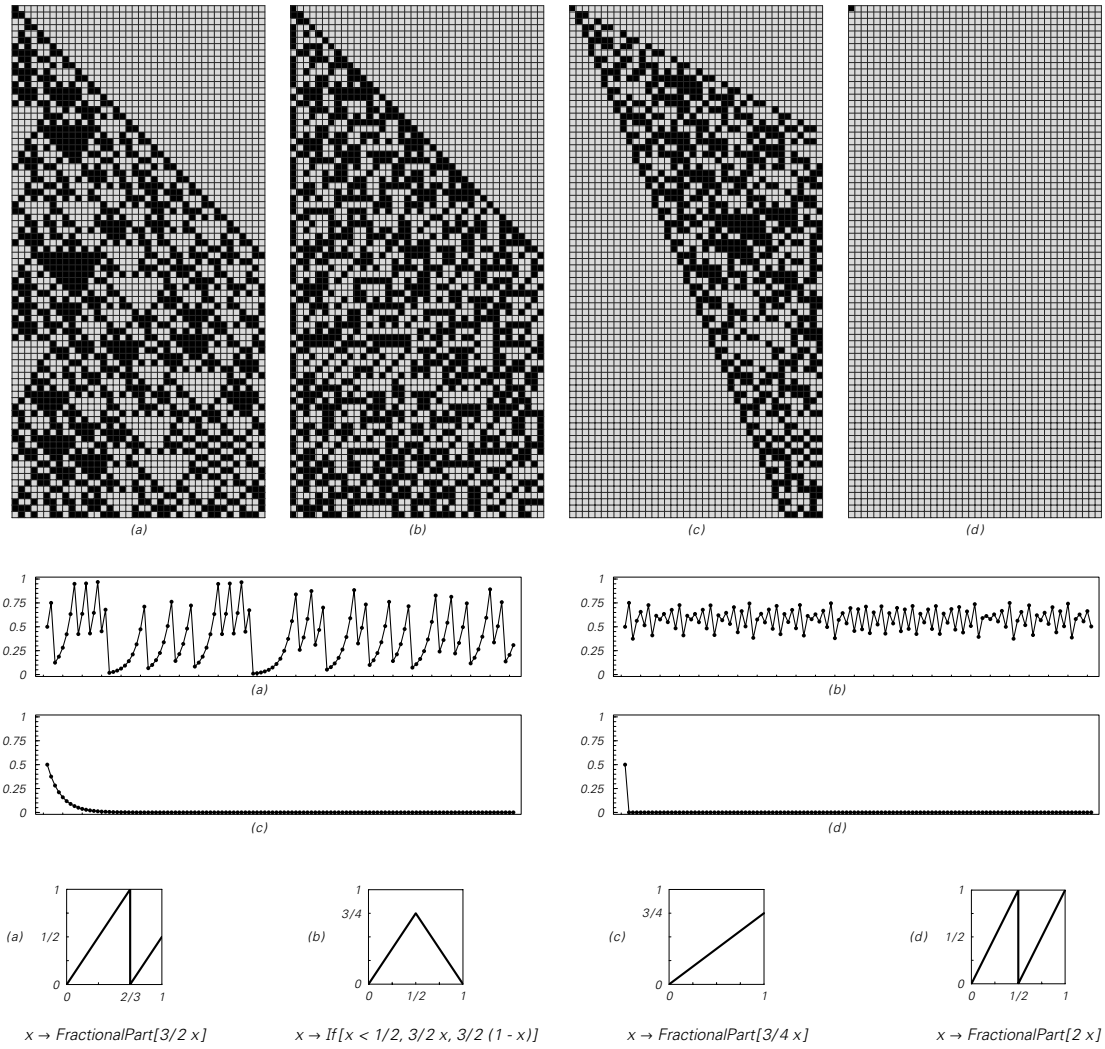
Cases (a) and (b) on the first page show much the same kind of complexity that we have seen in many other systems in this chapter—in both digit sequences and sizes of numbers. Case (c) shows complexity in digit sequences, but the sizes of the numbers it generates rapidly tend to 0. Case (d), however, seems essentially trivial—and shows no complexity in either digit sequences or sizes of numbers.

On the first of the next two pages all the examples start with the number $1/2$—which has a simple digit sequence. But the examples on the second of the next two pages instead start with the number $\pi/4$—which has a seemingly random digit sequence.

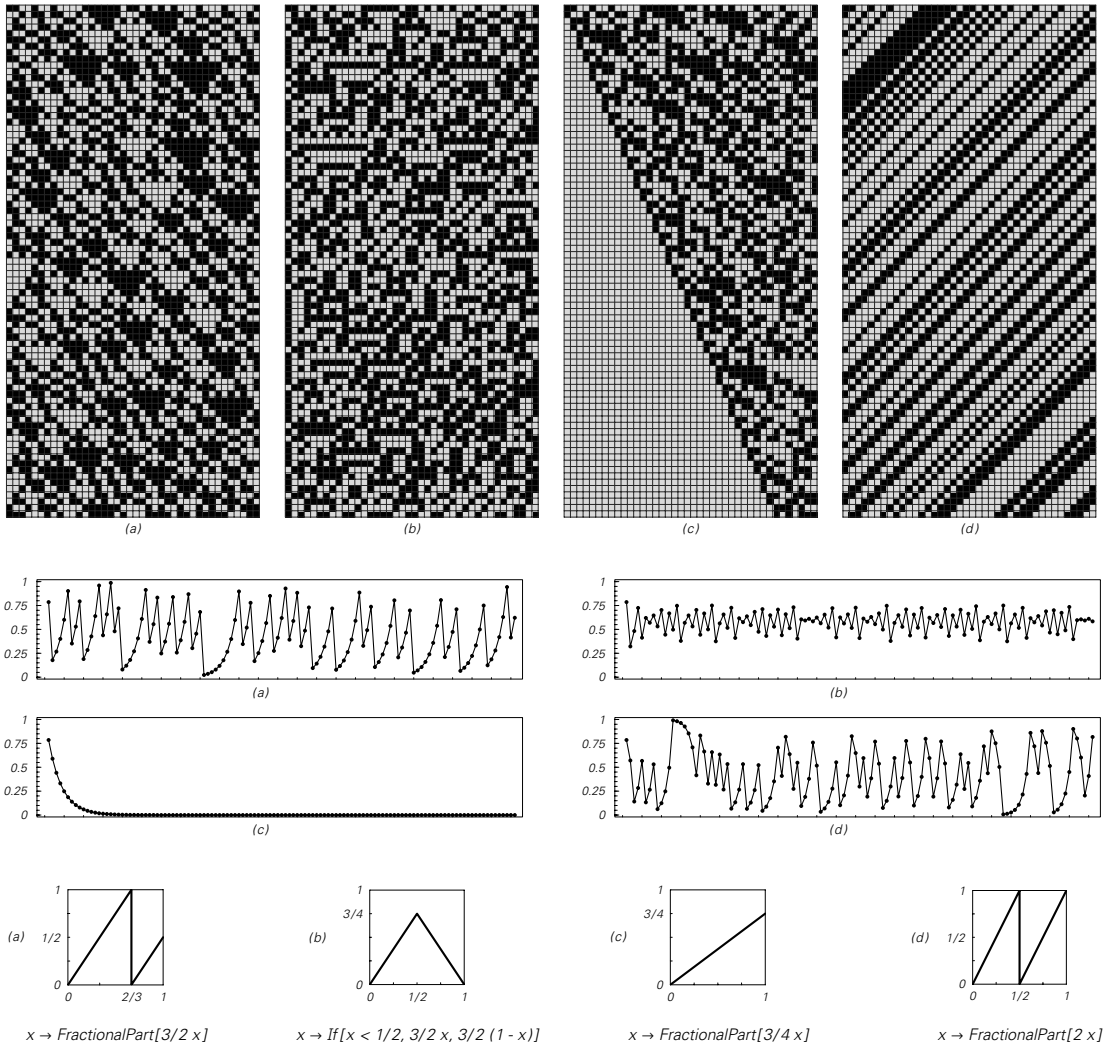
Cases (a), (b) and (c) look very similar on both pages, particularly in terms of sizes of numbers. But case (d) looks quite different. For on the first page it just yields 0's. But on the second page, it yields numbers whose sizes continually vary in a seemingly random way.

If one looks at digit sequences, it is rather clear why this happens. For as the picture illustrates, the so-called shift map used in case (d) simply serves to shift all digits one position to the left at each step. And this means that over the course of the evolution of the system, digits further to the right in the original number will progressively end up all the way to the left—so that insofar as these digits show randomness, this will lead to randomness in the sizes of the numbers generated.

It is important to realize, however, that in no real sense is any randomness actually being generated by the evolution of this system. Instead, it is just that randomness that was inserted in the digit sequence of the original number shows up in the results one gets.



Examples of iterated maps starting from simple initial conditions. At each step there is a number x between 0 and 1 that is updated by applying a fixed mapping. The four mappings considered here are given above both as formulas and in terms of plots. The pictures at the top of the page show the base 2 digit sequences of successive numbers obtained by iterating this mapping, while the pictures in the middle of the page plot the sizes of these numbers. In all cases, the initial conditions consist of the number 1/2—which has a very simple digit sequence. Yet despite this simplicity, cases (a) and (b) show considerable complexity in both the digit sequences and the sizes of the numbers produced (compare page 122). In case (c), the digit sequences are complicated but the sizes of the numbers tend rapidly to zero. And finally, in case (d), neither the digit sequences nor the sizes of numbers are anything but trivial. Note that in the pictures above each horizontal row of digits corresponds to a number, and that digits further to the left contribute progressively more to the size of this number.



The same iterated maps as on the facing page, but now started from the initial condition $\pi/4$—a number with a seemingly random digit sequence. After fairly few steps, cases (a) and (b) yield behavior that is almost indistinguishable from what was seen with simple initial conditions on the facing page. And in case (c), the same exponential decay in the sizes of numbers occurs as before. But in case (d), the behavior is much more complicated. Indeed, if one just looked at the sizes of numbers produced, then one sees the same kind of complexity as in cases (a) and (b). But looking at digit sequences one realizes that this complexity is actually just a direct transcription of complexity introduced by giving an initial condition with a seemingly random digit sequence. Case (d) is the so-called shift map—a classic example of a system that exhibits the sensitive dependence on initial conditions often known as chaos.

This is very different from what happens in cases (a) and (b). For in these cases complex and seemingly random results are obtained even on the first of the previous two pages—when the original number has a very simple digit sequence. And the point is that these maps actually do intrinsically generate complexity and randomness; they do not just transcribe it when it is inserted in their initial conditions.

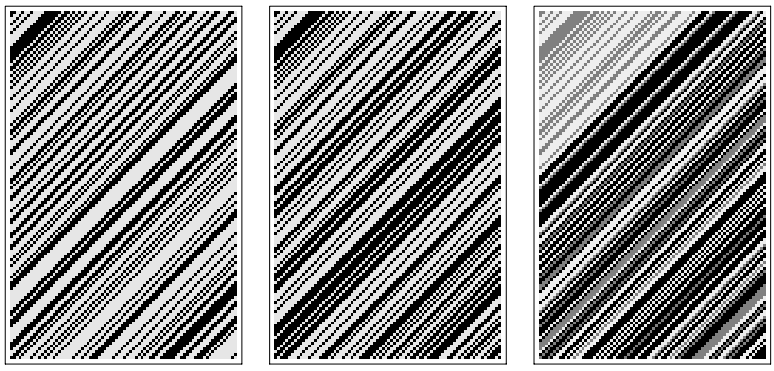
In the context of the approach I have developed in this book this distinction is easy to understand. But with the traditional mathematical approach, things can get quite confused. The main issue—already mentioned at the beginning of this chapter—is that in this approach the only attribute of numbers that is usually considered significant is their size. And this means that any issue based on discussing explicit digit sequences for numbers—and whether for example they are simple or complicated—tends to seem at best bizarre.

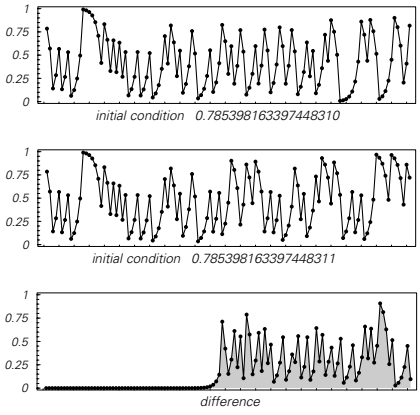
Indeed, thinking about numbers purely in terms of size, one might imagine that as soon as any two numbers are sufficiently close in size they would inevitably lead to results that are somehow also close. And in fact this is for example the basis for much of the formalism of calculus in traditional mathematics.

But the essence of the so-called chaos phenomenon is that there are some systems where arbitrarily small changes in the size of a number can end up having large effects on the results that are produced. And the shift map shown as case (d) on the previous two pages turns out to be a classic example of this.

The pictures at the top of the facing page show what happens if one uses as the initial conditions for this system two numbers whose sizes differ by just one part in a billion billion. And looking at the plots of sizes of numbers produced, one sees that for quite a while these two different initial conditions lead to results that are indistinguishably close. But at some point they diverge and soon become quite different.

And at least if one looks only at the sizes of numbers, this seems rather mysterious. But as soon as one looks at digit sequences, it immediately becomes much clearer. For as the pictures at the top of the facing page show, the fact that the numbers which are used as initial conditions differ only by a very small amount in size just means that their first several digits are the same. And for a while these digits are





The effect of making a small change in the initial conditions for the shift map—shown as case (d) on pages 150 and 151. The first picture shows results for the same initial condition as on page 151. The second picture shows what happens if one changes the size of the number in this initial condition by just one part in a billion billion. The plots to the left indicate that for a while the sizes of numbers obtained by the evolution of the system in these two cases are indistinguishable. But suddenly the results diverge and become completely different. Looking at the digit sequences above shows why this happens. The point is that a small change in the size of the number in the initial conditions corresponds to a change in digits far to the right. But the evolution of the system progressively shifts digits to the left, so that the digits which differ eventually become important. The much-investigated chaos phenomenon consists essentially of this effect.

what is important. But since the evolution of the system continually shifts digits to the left, it is inevitable that the differences that exist in later digits will eventually become important.

The fact that small changes in initial conditions can lead to large changes in results is a somewhat interesting phenomenon. But as I will discuss at length in Chapter 7 one must realize that on its own this cannot explain why randomness—or complexity—should occur in any particular case. And indeed, for the shift map what we have seen is that randomness will occur only when the initial conditions that are given happen to be a number whose digit sequence is random.

But in the past what has often been confusing is that traditional mathematics implicitly tends to assume that initial conditions of this kind are in some sense inevitable. For if one thinks about numbers

purely in terms of size, one should make no distinction between numbers that are sufficiently close in size. And this implies that in choosing initial conditions for a system like the shift map, one should therefore make no distinction between the exact number $1/2$ and numbers that are sufficiently close in size to $1/2$.

But it turns out that if one picks a number at random subject only to the constraint that its size be in a certain range, then it is overwhelmingly likely that the number one gets will have a digit sequence that is essentially random. And if one then uses this number as the initial condition for a shift map, the results will also be correspondingly random—just like those on the previous page.

In the past this fact has sometimes been taken to indicate that the shift map somehow fundamentally produces randomness. But as I have discussed above, the only randomness that can actually come out of such a system is randomness that was explicitly put in through the details of its initial conditions. And this means that any claim that the system produces randomness must really be a claim about the details of what initial conditions are typically given for it.

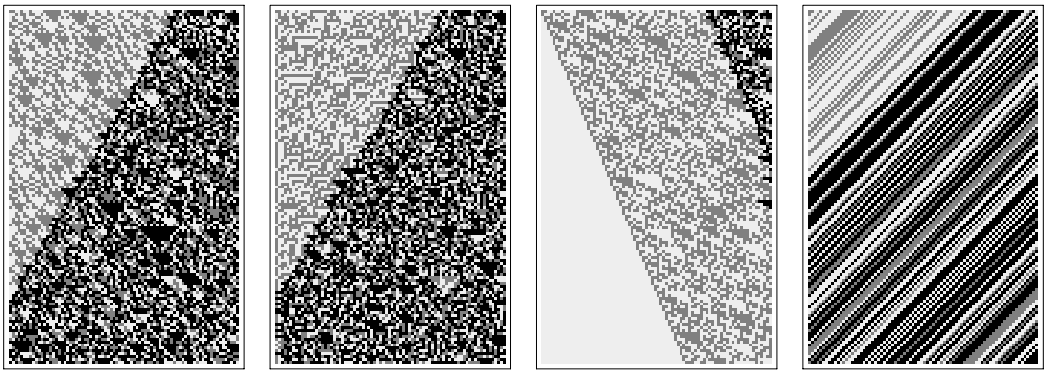
I suppose in principle it could be that nature would effectively follow the same idealization as in traditional mathematics, and would end up picking numbers purely according to their size. And if this were so, then it would mean that the initial conditions for systems like the shift map would naturally have digit sequences that are almost always random.

But this line of reasoning can ultimately never be too useful. For what it says is that the randomness we see somehow comes from randomness that is already present—but it does not explain where that randomness comes from. And indeed—as I will discuss in Chapter 7—if one looks only at systems like the shift map then it is not clear any new randomness can ever actually be generated.

But a crucial discovery in this book is that systems like (a) and (b) on pages 150 and 151 can show behavior that seems in many respects random even when their initial conditions show no sign of randomness and are in fact extremely simple.

Yet the fact that systems like (a) and (b) can intrinsically generate randomness even from simple initial conditions does not mean that they

do not also show sensitive dependence on initial conditions. And indeed the pictures below illustrate that even in such cases changes in digit sequences are progressively amplified—just like in the shift map case (d).



Differences in digit sequences produced by a small change in initial conditions for the four iterated maps discussed in this section. Cases (a), (b) and (d) exhibit sensitive dependence on initial conditions, in the sense that a change in insignificant digits far to the right eventually grows to affect all digits. Case (c) does not show such sensitivity to initial conditions, but instead always evolves to 0, independent of its initial conditions.

But the crucial point that I will discuss more in Chapter 7 is that the presence of sensitive dependence on initial conditions in systems like (a) and (b) in no way implies that it is what is responsible for the randomness and complexity we see in these systems. And indeed, what looking at the shift map in terms of digit sequences shows us is that this phenomenon on its own can make no contribution at all to what we can reasonably consider the ultimate production of randomness.

Continuous Cellular Automata

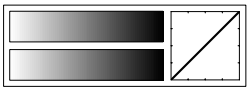
Despite all their differences, the various kinds of programs discussed in the previous chapter have one thing in common: they are all based on elements that can take on only a discrete set of possible forms, typically just colors black and white. And in this chapter, we have introduced a similar kind of discreteness into our study of systems based on numbers

by considering digit sequences in which each digit can again have only a discrete set of possible values, typically just 0 and 1.

So now a question that arises is whether all the complexity we have seen in the past three chapters somehow depends on the discreteness of the elements in the systems we have looked at.

And to address this question, what I will do in this section is to consider a generalization of cellular automata in which each cell is not just black or white, but instead can have any of a continuous range of possible levels of gray. One can update the gray level of each cell by using rules that are in a sense a cross between the totalistic cellular automaton rules that we discussed at the beginning of the last chapter and the iterated maps that we just discussed in the previous section.

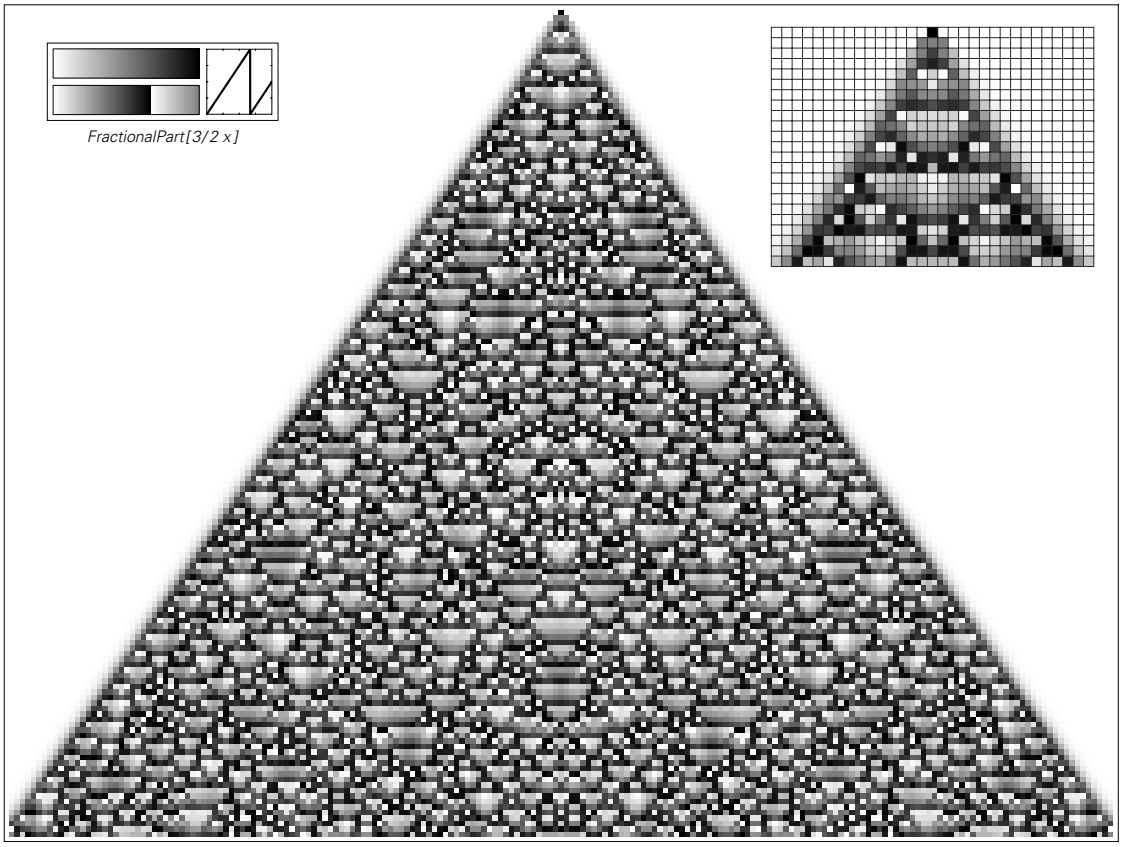
The idea is to look at the average gray level of a cell and its immediate neighbors, and then to get the gray level for that cell at the next step by applying a fixed mapping to the result. The picture below shows a very simple case in which the new gray level of each cell is exactly the average of the one for that cell and its immediate neighbors. Starting from a single black cell, what happens in this case is that the gray essentially just diffuses away, leaving in the end a uniform pattern.



A continuous cellular automaton in which each cell can have any level of gray between white (0) and black (1). The rule shown here takes the new gray level of each cell to be the average of its own gray level and those of its immediate neighbors.

0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0.333	0.333	0.333	0	0	0	0
0	0	0	0.111	0.222	0.333	0.222	0.111	0	0	0
0	0	0.037	0.111	0.222	0.259	0.222	0.111	0.037	0	0
0	0.012	0.049	0.123	0.198	0.235	0.198	0.123	0.049	0.012	0
0.004	0.021	0.062	0.123	0.185	0.21	0.185	0.123	0.062	0.021	0.004

The picture on the facing page shows what happens with a slightly more complicated rule in which the average gray level is multiplied by $3/2$, and then only the fractional part is kept if the result of this is greater than 1.



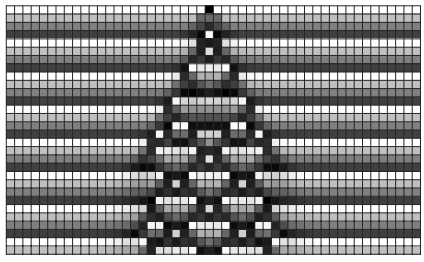
0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0.5	0.5	0.5	0	0	0	0
0	0	0	0.25	0.5	0.75	0.5	0.25	0	0	0
0	0	0.125	0.375	0.75	0.875	0.75	0.375	0.125	0	0
0	0.063	0.25	0.625	0	0.188	0	0.625	0.25	0.063	0
0.031	0.156	0.469	0.438	0.406	0.094	0.406	0.438	0.469	0.156	0.031

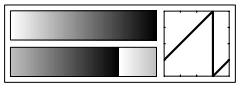
A continuous cellular automaton with a slightly more complicated rule. The rule takes the new gray level of each cell to be the fractional part of the average gray level of the cell and its neighbors multiplied by 3/2. The picture shows that starting from a single black cell, this rule yields behavior of considerable complexity. Note that the operation performed on individual average gray levels is exactly iterated map (a) from page 150.

And what we see is that despite the presence of continuous gray levels, the behavior that is produced exhibits the same kind of complexity that we have seen in many ordinary cellular automata and other systems with discrete underlying elements.

In fact, it turns out that in continuous cellular automata it takes only extremely simple rules to generate behavior of considerable complexity. So as an example the picture below shows a rule that determines the new gray level for a cell by just adding the constant 1/4 to the average gray level for the cell and its immediate neighbors, and then taking the fractional part of the result.

0	0	0	0	0	1	0	0	0	0	0
0.25	0.25	0.25	0.25	0.583	0.583	0.583	0.25	0.25	0.25	0.25
0.5	0.5	0.5	0.611	0.722	0.833	0.722	0.611	0.5	0.5	0.5
0.75	0.75	0.787	0.861	0.972	0.009	0.972	0.861	0.787	0.75	0.75
0	0.012	0.049	0.123	0.864	0.901	0.864	0.123	0.049	0.012	0
0.254	0.271	0.312	0.596	0.88	0.127	0.88	0.596	0.312	0.271	0.254





FractionalPart[x+1/4]

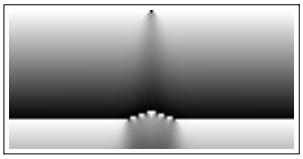
A continuous cellular automaton whose rule adds the constant 1/4 to the average gray level for a cell and its immediate neighbors, and takes the fractional part of the result. The background simply repeats every 4 steps, but the main pattern has a complex and in many respects random form.

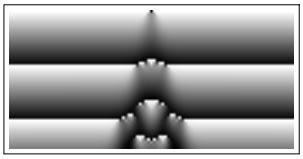
The facing page and the one after show what happens when one chooses different values for the constant that is added. A remarkable diversity of behavior is seen. Sometimes the behavior is purely repetitive, but often it has features that seem effectively random.

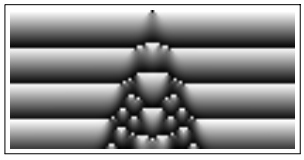
And in fact, as the picture in the middle of page 160 shows, it is even possible to find cases that exhibit localized structures very much like those occasionally seen in ordinary cellular automata.

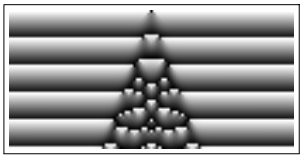
Continuous cellular automata with the same kind of rules as in the picture above, but with a variety of different constants being added. Note that it is not so much the size of the constant as properties like its digit sequence that seem to determine the overall form of behavior produced in each case. ▸

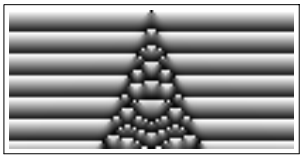

0

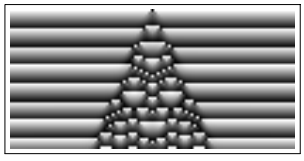

0.025

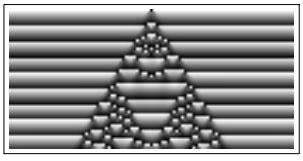

0.05

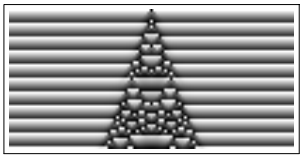

0.075

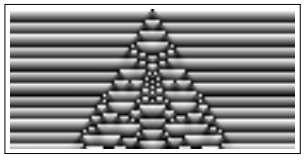

0.1

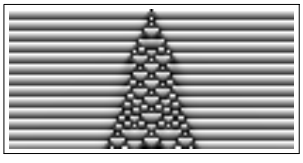

0.125

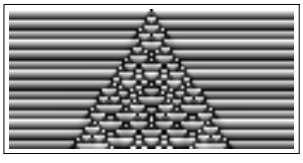

0.15

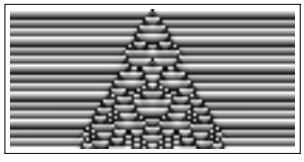

0.175

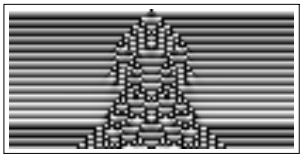

0.2

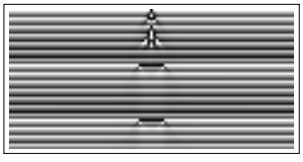

0.225

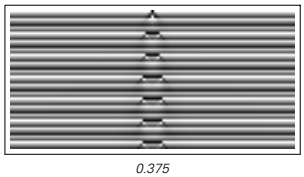

0.25

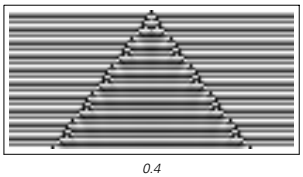

0.275

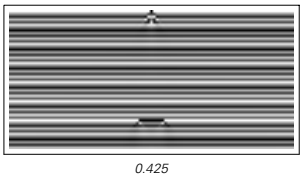

0.3

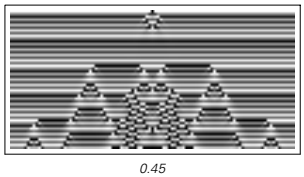

0.325

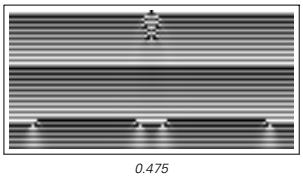

0.35

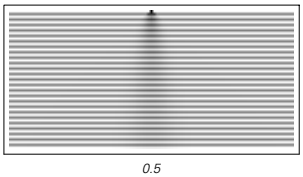

0.375

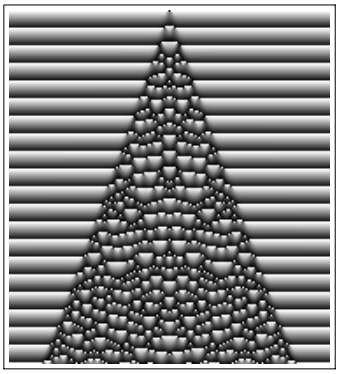

0.4


0.425

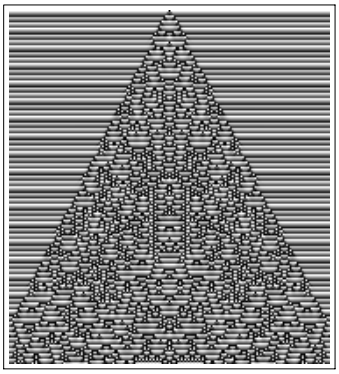

0.45


0.475

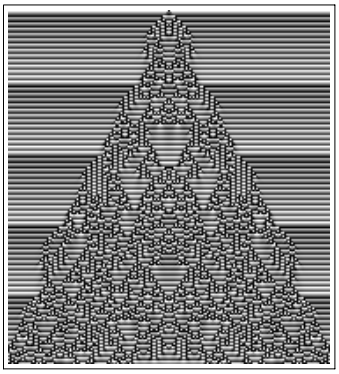

0.5



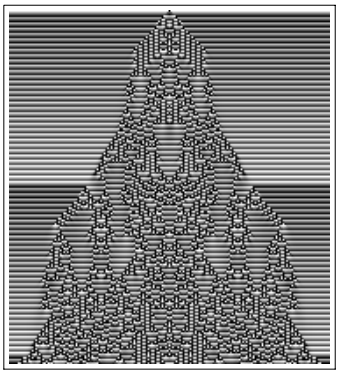
0.1



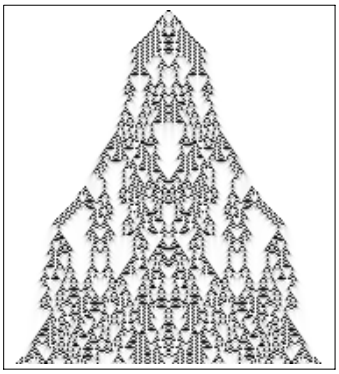
0.3



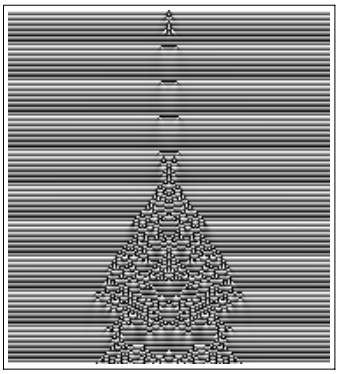
0.325



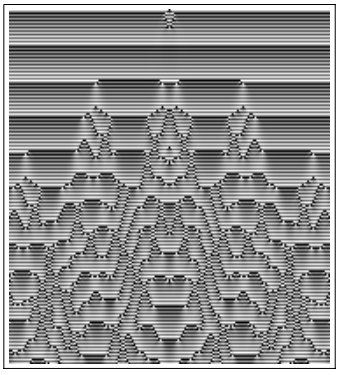
0.3299



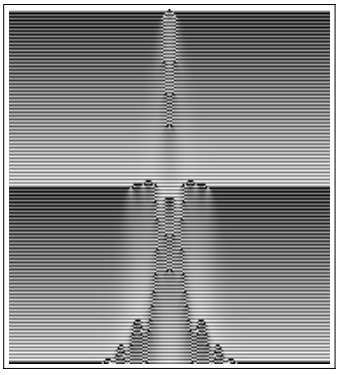
0.3299 (differences)



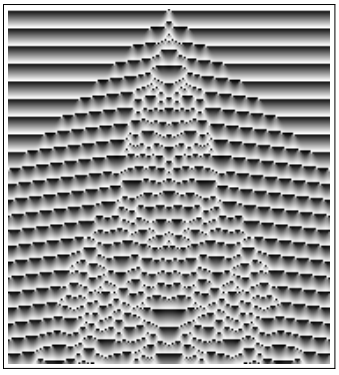
0.35



0.475



0.495



0.9

More steps in the evolution of continuous cellular automata with the same kind of rules as on the previous page. In order to remove the uniform stripes, the picture in the middle shows the difference between the gray level of each cell and its immediate neighbor. Note the presence of discrete localized structures even though the underlying rules for the system involve continuous gray levels.

Partial Differential Equations

By introducing continuous cellular automata with a continuous range of gray levels, we have successfully removed some of the discreteness that exists in ordinary cellular automata. But there is nevertheless much discreteness that remains: for a continuous cellular automaton is still made up of discrete cells that are updated in discrete time steps.

So can one in fact construct systems in which there is absolutely no such discreteness? The answer, it turns out, is that at least in principle one can, although to do so requires a somewhat higher level of mathematical abstraction than has so far been necessary in this book.

The basic idea is to imagine that a quantity such as gray level can be set up to vary continuously in space and time. And what this means is that instead of just having gray levels in discrete cells at discrete time steps, one supposes that there exists a definite gray level at absolutely every point in space and every moment in time—as if one took the limit of an infinite collection of cells and time steps, with each cell being an infinitesimal size, and each time step lasting an infinitesimal time.

But how does one give rules for the evolution of such a system? Having no explicit time steps to work with, one must instead just specify the rate at which the gray level changes with time at every point in space. And typically one gives this rate as a simple formula that depends on the gray level at each point in space, and on the rate at which that gray level changes with position.

Such rules are known in mathematics as partial differential equations, and in fact they have been widely studied for about two hundred years. Indeed, it turns out that almost all the traditional mathematical models that have been used in physics and other areas of science are ultimately based on partial differential equations. Thus, for example, Maxwell's equations for electromagnetism, Einstein's equations for gravity, Schrödinger's equation for quantum mechanics and the Hodgkin-Huxley equation for the electrochemistry of nerve cells are all examples of partial differential equations.

It is in a sense surprising that systems which involve such a high level of mathematical abstraction should have become so widely used

in practice. For as we shall see later in this book, it is certainly not that nature fundamentally follows these abstractions.

And I suspect that in fact the current predominance of partial differential equations is in many respects a historical accident—and that had computer technology been developed earlier in the history of mathematics, the situation would probably now be very different.

But particularly before computers, the great attraction of partial differential equations was that at least in simple cases explicit mathematical formulas could be found for their behavior. And this meant that it was possible to work out, for example, the gray level at a particular point in space and time just by evaluating a single mathematical formula, without having in a sense to follow the complete evolution of the partial differential equation.

The pictures on the facing page show three common partial differential equations that have been studied over the years.

The first picture shows the diffusion equation, which can be viewed as a limiting case of the continuous cellular automaton on page 156. Its behavior is always very simple: any initial gray progressively diffuses away, so that in the end only uniform white is left.

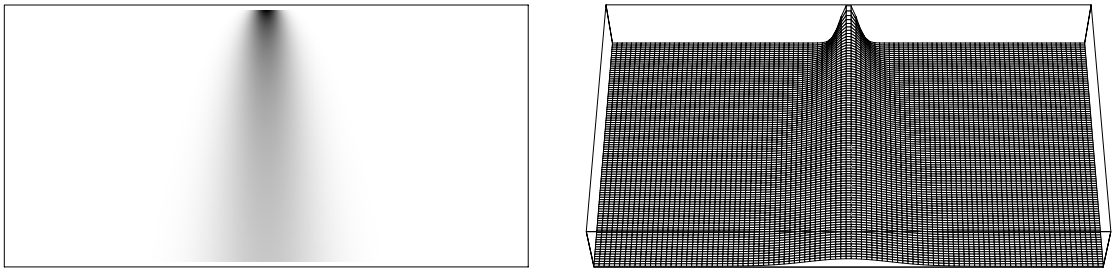
The second picture shows the wave equation. And with this equation, the initial lump of gray shown just breaks into two identical pieces which propagate to the left and right without change.

The third picture shows the sine-Gordon equation. This leads to slightly more complicated behavior than the other equations—though the pattern it generates still has a simple repetitive form.

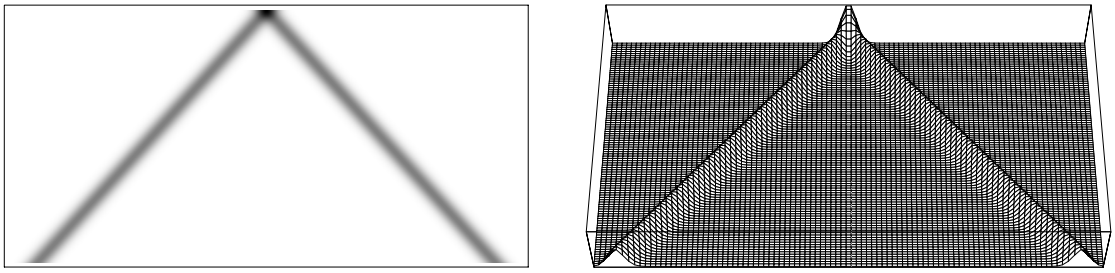
Considering the amount of mathematical work that has been done on partial differential equations, one might have thought that a vast range of different equations would by now have been studied. But in fact almost all the work—at least in one dimension—has concentrated on just the three specific equations on the facing page, together with a few others that are essentially equivalent to them.

And as we have seen, these equations yield only simple behavior.

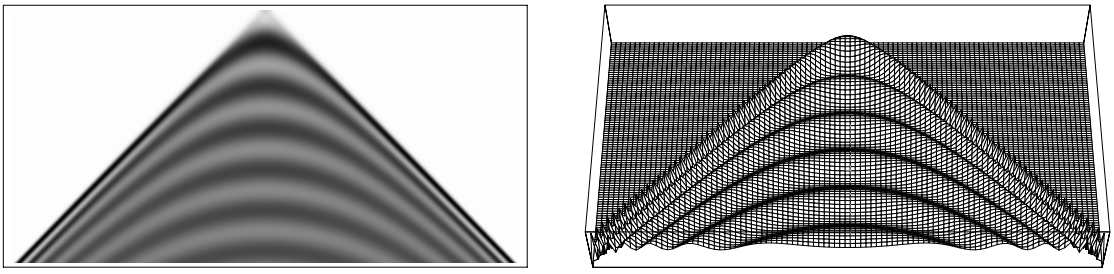
So is it in fact possible to get more complicated behavior in partial differential equations? The results in this book on other kinds of systems strongly suggest that it should be. But traditional



diffusion equation: $\partial_t u[t, x] = 1/4\, \partial_{xx} u[t, x]$



wave equation: $\partial_{tt} u[t, x] = \partial_{xx} u[t, x]$



sine-Gordon soliton equation: $\partial_{tt} u[t, x] = \partial_{xx} u[t, x] + \text{Sin}[u[t, x]]$

Three partial differential equations that have historically been studied extensively. Just like in other pictures in this book, position goes across the page, and time down the page. In each equation u is the gray level at a particular point, $\partial_t u$ is the rate of change (derivative) of the gray level with time, and $\partial_{tt} u$ is the rate of change of that rate of change (second derivative). Similarly, $\partial_x u$ is the rate of change with position in space, and $\partial_{xx} u$ is the rate of change of that rate of change. On this page and the ones that follow the initial conditions used are $u = e^{-x^2}$, $\partial_t u = 0$.

mathematical methods give very little guidance about how to find such behavior. Indeed, it seems that the best approach is essentially just to search through many different partial differential equations, looking for ones that turn out to show complex behavior.

But an immediate difficulty is that there is no obvious way to sample possible partial differential equations. In discrete systems such as cellular automata there are always a discrete set of possible rules. But in partial differential equations any mathematical formula can appear.

Nevertheless, by representing formulas as symbolic expressions with discrete sets of possible components, one can devise at least some schemes for sampling partial differential equations.

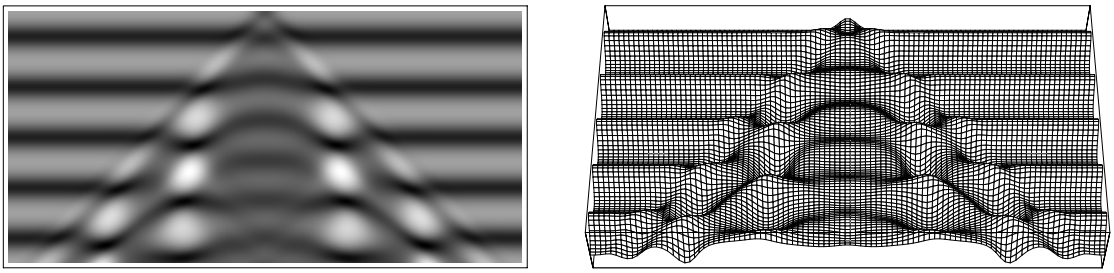
But even given a particular partial differential equation, there is no guarantee that the equation will yield self-consistent results. Indeed, for a very large fraction of randomly chosen partial differential equations what one finds is that after just a small amount of time, the gray level one gets either becomes infinitely large or starts to vary infinitely quickly in space or time. And whenever such phenomena occur, the original equation can no longer be used to determine future behavior.

But despite these difficulties I was eventually able to find the partial differential equations shown on the next two pages.

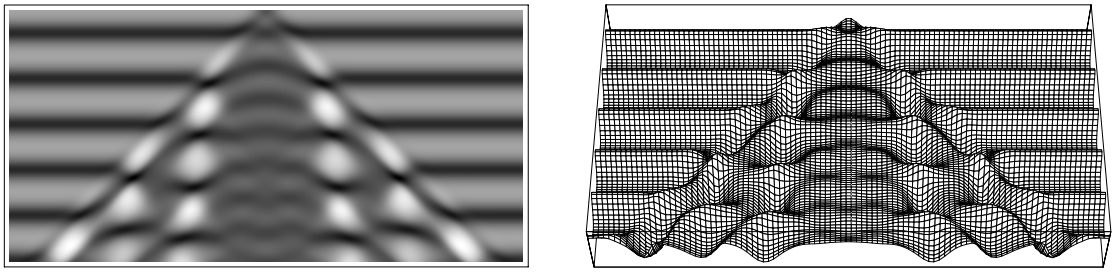
The mathematical statement of these equations is fairly simple. But as the pictures show, their behavior is highly complex.

Indeed, strangely enough, even though the underlying equations are continuous, the patterns they produce seem to involve patches that have a somewhat discrete structure.

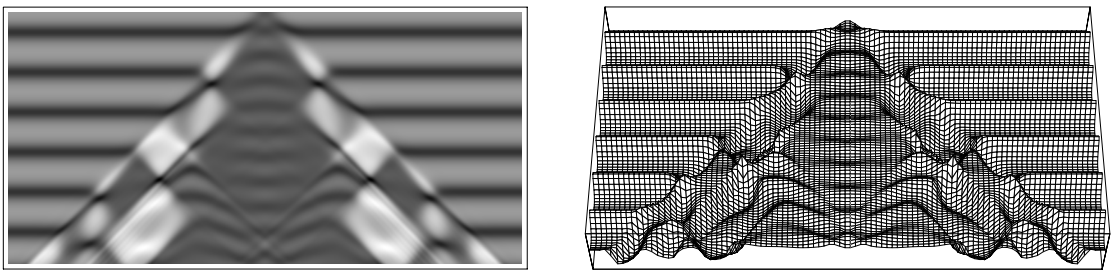
But the main point that the pictures on the next two pages make is that the kind of complex behavior that we have seen in this book is in no way restricted to systems that are based on discrete elements. It is certainly much easier to find and to study such behavior in these discrete systems, but from what we have learned in this section, we now know that the same kind of behavior can also occur in completely continuous systems such as partial differential equations.



$$\partial_{tt} u[t, x] = \partial_{xx} u[t, x] + (1 - u[t, x]^2)(1 + u[t, x])$$

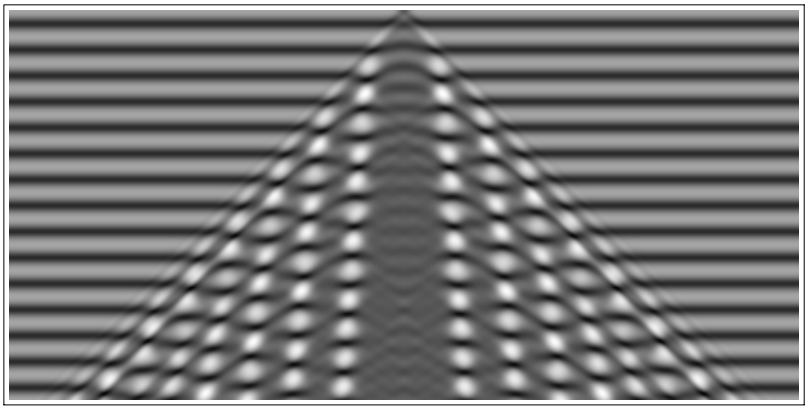


$$\partial_{tt} u[t, x] = \partial_{xx} u[t, x] + (1 - u[t, x]^2)(1 + 2\, u[t, x])$$

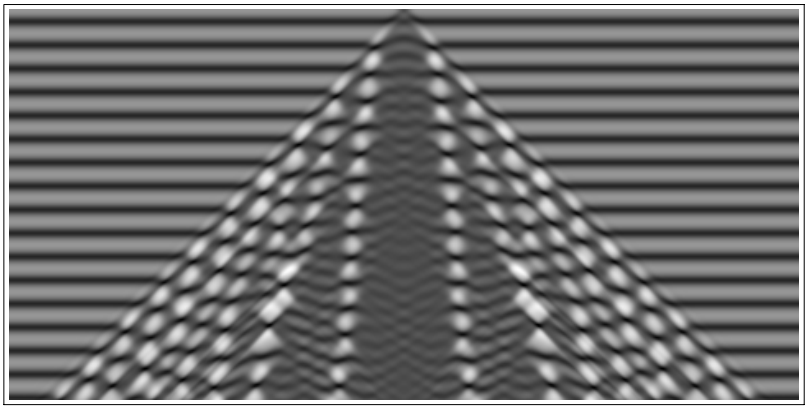


$$\partial_{tt} u[t, x] = \partial_{xx} u[t, x] + (1 - u[t, x]^2)(1 + 4\, u[t, x])$$

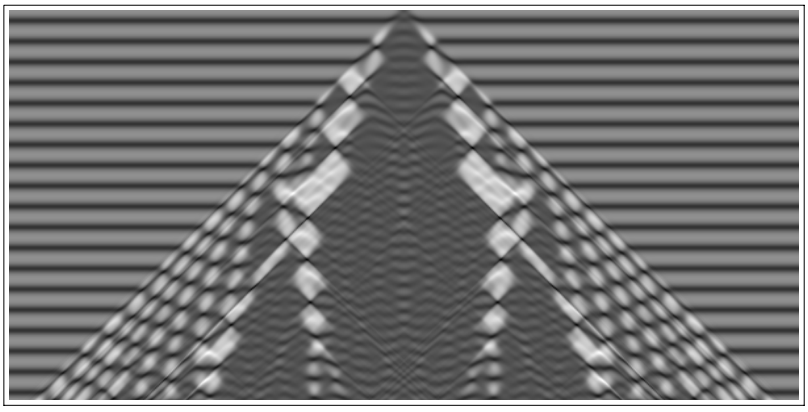
Examples of partial differential equations I have found that have more complicated behavior. The background in each case purely is repetitive, but the main part of the pattern is complex, and reminiscent of what is produced by continuous cellular automata and many other kinds of systems discussed in this book.



$$\partial_{tt} u[t, x] = \partial_{xx} u[t, x] + (1 - u[t, x]^2)(1 + u[t, x])$$



$$\partial_{tt} u[t, x] = \partial_{xx} u[t, x] + (1 - u[t, x]^2)(1 + 2\,u[t, x])$$



$$\partial_{tt} u[t, x] = \partial_{xx} u[t, x] + (1 - u[t, x]^2)(1 + 4\,u[t, x])$$

Continuous Versus Discrete Systems

One of the most obvious differences between my approach to science based on simple programs and the traditional approach based on mathematical equations is that programs tend to involve discrete elements while equations tend to involve continuous quantities.

But how significant is this difference in the end?

One might have thought that perhaps the basic phenomenon of complexity that I have identified could only occur in discrete systems. But from the results of the last few sections, we know that this is not the case.

What is true, however, is that the phenomenon was immensely easier to discover in discrete systems than it would have been in continuous ones. Probably complexity is not in any fundamental sense rarer in continuous systems than in discrete ones. But the point is that discrete systems can typically be investigated in a much more direct way than continuous ones.

Indeed, given the rules for a discrete system, it is usually a rather straightforward matter to do a computer experiment to find out how the system will behave. But given an equation for a continuous system, it often requires considerable analysis to work out even approximately how the system will behave. And in fact, in the end one typically has rather little idea which aspects of what one sees are actually genuine features of the system, and which are just artifacts of the particular methods and approximations that one is using to study it.

With all the work that was done on continuous systems in the history of traditional science and mathematics, there were undoubtedly many cases in which effects related to the phenomenon of complexity were seen. But because the basic phenomenon of complexity was not known and was not expected, such effects were probably always dismissed as somehow not being genuine features of the systems being studied. Yet when I came to investigate discrete systems there was no

◀ Solutions to the same equations as on the previous page over a longer period of time. Note the appearance of discrete structures. Particularly in the last picture some details are sensitive to the numerical approximation scheme used in computing the solution to the equation.

possibility of dismissing what I saw in such a way. And as a result I was in a sense forced into recognizing the basic phenomenon of complexity.

But now, armed with the knowledge that this phenomenon exists, it is possible to go back and look again at continuous systems.

And although there are significant technical difficulties, one finds as the last few sections have shown that the phenomenon of complexity can occur in continuous systems just as it does in discrete ones.

It remains much easier to be sure of what is going on in a discrete system than in a continuous one. But I suspect that essentially all of the various phenomena that we have observed in discrete systems in the past several chapters can in fact also be found even in continuous systems with fairly simple rules.

Systems Based on Numbers

The Notion of Numbers

■ **Implementation of digit sequences.** A whole number *n* can be converted to a sequence of digits in base *k* using *IntegerDigits[n, k]* or (see also page 1094)

```
Reverse[Mod[NestWhileList[Floor[#/k] &, n, # ≥ k &], k]]
```

and from a sequence of digits using *FromDigits[list, k]* or

```
Fold[k #1 + #2 &, 0, list]
```

For a number *x* between 0 and 1, the first *m* digits in its digit sequence in base *k* are given by *RealDigits[x, k, m]* or

```
Floor[k NestList[Mod[k #, 1] &, x, m - 1]]
```

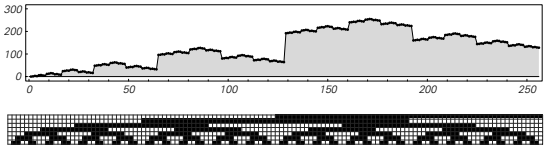
and from these digits one can reconstruct an approximation to the number using *FromDigits[{list, 0}, k]* or

```
Fold[#1/k + #2 &, 0, Reverse[list]]/k
```

■ **Gray code.** In looking at digit sequences, it is sometimes useful to consider ordering numbers by a criterion other than their size. An example is Gray code ordering, in which successive numbers are arranged to differ in only one digit. One possible such ordering for numbers with a total of *m* digits is

```
GrayCode[m_] :=
  Nest[Join[#, Length[#] + Reverse[#]] &, {0}, m]
```

The succession of sizes and digit sequences of numbers ordered in this way are shown below. (Note that the digit sequence picture is turned on its side relative to those in the main text). The number which appears at position *i* is given by *BitXor[i, Floor[i/2]]*. (Iterating the related function *BitXor[i, 2 i]* yields numbers whose digit sequences correspond to the rule 60 cellular automaton).



■ **A note for mathematicians.** Some mathematicians will at first find what I say in this chapter quite bizarre. It may help however to point out that the traditional view of numbers already shows signs of breaking down in many studies of dynamical systems done over the past few decades. Thus for example, instead of getting results in terms of continuous functions, Cantor sets very often appear. Indeed, the symbolic dynamics approach that is often used in dynamical systems theory is quite close to the digit sequence approach I use here—Markov partitions in dynamical systems theory are essentially just generalizations of digit expansions.

However, in the cases that are analyzed in dynamical systems theory, only shifts and other very simple operations are typically performed on digit sequences. And as a result, most of the phenomena that I discuss in this chapter have not been seen in work done in dynamical systems theory.

■ **History of numbers.** Numbers were probably first used many thousands of years ago in commerce, and initially only whole numbers and perhaps rational numbers were needed. But already in Babylonian times, practical problems of geometry began to require square roots. Nevertheless, for a very long time, and despite some development of algebra, only numbers that could somehow in principle be constructed mechanically were ever considered. The invention of fluxions by Isaac Newton in the late 1600s, however, introduced the idea of continuous variables—numbers with a continuous range of possible sizes. But while this was a convenient and powerful notion, it also involved a new level of abstraction, and it brought with it considerable confusion about fundamental issues. In fact, it was really only through the development of rigorous mathematical analysis in the late 1800s that this confusion finally began to clear up. And already by the 1880s Georg Cantor and others had constructed completely discontinuous functions, in which the idea of treating numbers as continuous variables where only the size matters was called into question. But until almost the 1970s, and the emergence of fractal geometry and chaos theory, these functions were largely considered as

mathematical curiosities, of no practical relevance. (See also page 1168.)

Independent of pure mathematics, however, practical applications of numbers have always had to go beyond the abstract idealization of continuous variables. For whether one does calculations by hand, by mechanical calculator or by electronic computer, one always needs an explicit representation for numbers, typically in terms of a sequence of digits of a certain length. (From the 1930s to 1960s, some work was done on so-called analog computers which used electrical voltages to represent continuous variables, but such machines turned out not to be reliable enough for most practical purposes.) From the earliest days of electronic computing, however, great efforts were made to try to approximate a continuum of numbers as closely as possible. And indeed for studying systems with fairly simple behavior, such approximations can typically be made to work. But as we shall see later in this chapter, with more complex behavior, it is almost inevitable that the approximation breaks down, and there is no choice but to look at the explicit representations of numbers. (See also page 1128.)

■ **History of digit sequences.** On an abacus or similar device numbers are in effect represented by digit sequences. In antiquity however most systems for writing numbers were like the Roman one and not based on digit sequences. An exception was the Babylonian base 60 system (from which hours:minutes:seconds notation derives). The Hindu-Arabic base 10 system in its modern form probably originated around 600 AD, and particularly following the work of Leonardo Fibonacci in the early 1200s, became common by the 1400s. Base 2 appears to have first been considered explicitly in the early 1600s (notably by John Napier in 1617), and was studied in detail by Gottfried Leibniz starting in 1679. The possibility of arbitrary bases was stated by Blaise Pascal in 1658. Various bases were used in puzzles, but rarely in pure mathematics (work by Georg Cantor in the 1860s being an exception). The first widespread use of base 2 was in electronic computers, starting in the late 1940s. Even in the 1980s digit sequences were viewed by most mathematicians as largely irrelevant for pure mathematical purposes. The study of fractals and nesting, the appearance of many algorithms involving digit sequences and the routine use of long numbers in *Mathematica* have however gradually made digit sequences be seen as more central to mathematics.

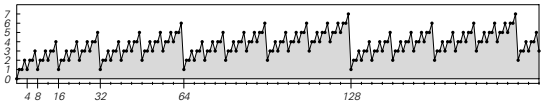
Elementary Arithmetic

■ **Page 117 · Substitution systems.** There are many connections between digit sequences and substitution systems, as discussed on page 891. The pattern shown here is essentially a rotated version of the pattern generated by the first substitution system on page 83.

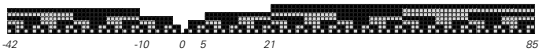
■ **Page 117 · Digit counts.** The number of black squares on row n in the pattern shown here is given by *DigitCount[n, 2, 1]* and is plotted below. This function appeared on page 870 in the discussion of binomial coefficients modulo 2, and will appear again in several other places in this book. Note the inequality *1 ≤ DigitCount[n, 2, 1] ≤ Log[2, n]*. Formulas for *DigitCount[n, 2, 1]* include *n - IntegerExponent[n!, 2]* and

2 n - Log[2, Denominator[Derivative[n][(1 - #)^-1/2 &][0]/n!]]

Straightforward generalizations of *DigitCount* can be defined for integer and non-integer bases and by looking not only at the total number of digits but also at correlations between digits. In all cases the analogs of the picture below have a nested structure.

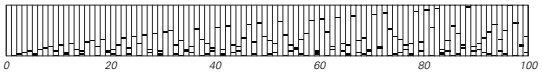


■ **Negative bases.** Given a suitable list of digits from 0 to $k - 1$ one can obtain any positive or negative number using *FromDigits[list, -k]*. The picture below shows the digit sequences of successive numbers in base -2; the row j from the bottom turns out to consist of alternating black and white blocks of length 2^j. (In ordinary base 2 a number $-n$ can be represented as on a typical electronic computer by complementing each digit, including leading 0's.) (See also page 1093.)

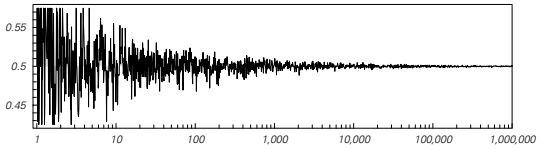


■ **Non-power bases.** One can consider representing numbers by *Sum[a[n] f[n], {n, 0, ∞}]* where the *f[n]* need not be k^n. So long as *f[n]* grows less rapidly than 2^n (as when *f = Fibonacci* or *f = Prime*), digits 0 and 1 will suffice, though the representation is not generally unique. (See page 1070.)

■ **Multiplicative digit sequences.** One can consider generalizations of digit sequences in which numbers are broken into parts combined not by addition but by multiplication. Since numbers can be factored uniquely into products of powers of primes, a number can be specified by a list in which 1's appear at the positions of the appropriate *Prime[m]*^{n} (which can be sorted by size) and 0's appear elsewhere, as shown below. Note that unlike the case of ordinary additive digits, far more than *Log[m]* digits are required to specify a number m.



▪ **Page 120 · Powers of three in base 2.** The n^{th} row in the pattern shown can be obtained simply as *IntegerDigits[3^n, 2]*. Even such individual rows seem in many respects random. The picture below shows the fraction of 1's that appear on successive rows. The fraction seems to tend to 1/2.



If one looks only at the rightmost s columns of the pattern, one sees repetition—but the period of the repetition grows like 2^s. Typical vertical columns have one obvious deviation from randomness: it is twice as probable for the same colors to occur on successive steps than for opposite colors. (For multiplier m in base k, the relative frequencies of pairs *{i, j}* are given by *Quotient[a i - j - 1 + m, k] - Quotient[m i - j - 1, k]*.)

The sequence *Mod[3^n, 2^s]* obtained from the rightmost s digits corresponds to a simple linear congruential pseudorandom number generator. Such generators are widely used in practical computer systems, as discussed further on page 974. (Note that in the particular case used here, pairs of numbers *Mod[{3^n, 3^{n+1}}, 2^s]* always lie on lines; with multipliers other than 3, such regularities may occur for longer blocks of numbers.)

Note that if one uses base 6 rather than base 2, then as shown on page 614 powers of 3 still yield a complicated pattern, but all operations are strictly local, and the system corresponds to a cellular automaton with 6 possible colors for each cell and rule *{a_, b_, c_} → 3 Mod[b, 2] + Floor[c/2]* (see page 1093).

▪ **Leading digits.** In base b the leading digits of powers are not equally probable, but follow the logarithmic law from page 914.

▪ **Page 122 · Powers of 3/2.** The n^{th} value shown in the plot here is *Mod[(3/2)^n, 1]*. Measurements suggest that these values are uniformly distributed in the range 0 to 1, but despite a fair amount of mathematical work since the 1940s, there has been no substantial progress towards proving this.

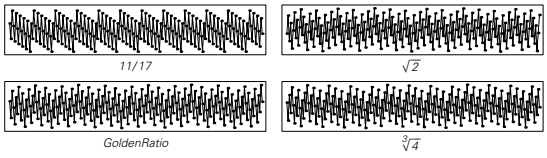
In base 6, $(3/2)^n$ is a cellular automaton with rule

```
{a_, b_, c_} → 3 Mod[a + Quotient[b, 2], 2] +
    Quotient[3 Mod[b, 2] + Quotient[c, 2], 2]
```

(Note that this rule is invertible.) Looking at $u\,(3/2)^n$ then corresponds to studying the cellular automaton with an initial condition given by the base 6 digits of u. It is then possible to find special values of u (an example is 0.166669170371...) which make the first digit in the fractional part of $u\,(3/2)^n$ always nonzero, so that *Mod[u (3/2)^n, 1] > 1/6*. In general, it seems that *Mod[u (3/2)^n, 1]* can be kept as large as about *0.3* (e.g. with *u = 0.38906669065 ...*) but no larger.

▪ **General powers.** It has been known in principle since the 1930s that *Mod[h^n, 1]* is uniformly distributed in the range 0 to 1 for almost all values of h. However, no specific value of h for which this is true has ever been explicitly found. (Some attempts to construct such values were made in the 1970s.) Exceptions are known to include so-called Pisot numbers such as *GoldenRatio*, $\sqrt{2}+1$ and *Root[#^3 - # - 1 &, 1]* (the numerically smallest of all Pisot numbers) for which *Mod[h^n, 1]* becomes 0 or 1 for large n. Note that *Mod[x h^n, 1]* effectively extracts successive digits of x in base h (see pages 149 and 919).

▪ **Multiples of irrational numbers.** Instead of powers one can consider successive multiples *Mod[h n, 1]* of a number h. The pictures below show results obtained as a function of n for various choices of h. (These correspond to positions of a particle bouncing around in an idealized box, as discussed on pages 971 and 1022.)



When h is a rational number, the sequence always repeats. But in all other cases, the sequence does not repeat, and in fact it is known that a uniform distribution of values is obtained. (The average difference of successive values is maximized for *h = GoldenRatio*, as mentioned on page 891.)

▪ **Relation to substitution systems.** Despite the uniform distribution result in the note above, the sequence *Floor[(n + 1) h] - Floor[n h]* is definitely not completely random, and can in fact be generated by a sequence of substitution rules. The first m rules (which yield far more than m elements of the original sequence) are obtained for any h that is not a rational number from the continued fraction form (see page 914) of h by

```
Map[({0 → Join[#, {1}], 1 → Join[#, {1, 0}]} &)[Table[0,
    {# - 1}]] &, Reverse[Rest[ContinuedFraction[h, m]]]]
```

Given these rules, the original sequence is given by

```
Floor[h] + Fold[Flatten[#1 /. #2] &, {0}, rules]
```

If h is the solution to a quadratic equation, then the continued fraction form is repetitive, and so there are a limited number

of different substitution rules. In this case, therefore, the original sequence can be found by a neighbor-independent substitution system of the kind discussed on page 82. For *h = GoldenRatio* the substitution system is *{0 → {1}, 1 → {1, 0}}* (see page 890), for $h = \sqrt{2}$ it is *{0 → {0, 1}, 1 → {0, 1, 0}}* (see page 892) and for $h = \sqrt{3}$ it is *{0 → {1, 1, 0}, 1 → {1, 1, 0, 1}}*. (The presence of nested structure is particularly evident in *FoldList[Plus, 0, Table[Mod[h n, 1] - 1/2, {n, max}]]*.) (See also pages 892, 916, 932 and 1084.)

■ **Other uniformly distributed sequences.** Cases in which *Mod[a[n], 1]* is uniformly distributed include $\sqrt{n}$, *n Log[n]*, *Log[Fibonacci[n]]*, *Log[n!]*, $h n^2$ and *h Prime[n]* (*h* irrational) and probably *n Sin[n]*. (See also page 914.)

■ **Page 122 · Implementation.** The evolution for *t* steps of the system at the top of the page can be computed simply by

```
NestList[If[EvenQ[#], 3#/2, 3(# + 1)/2] &, 1, t]
```

■ **Page 122 · The *3n+1* problem.** The system described here is similar to the so-called *3 n + 1* problem, in which one looks at the rule *n → If[EvenQ[n], n/2, (3 n + 1)/2]* and asks whether for any initial value of *n* the system eventually evolves to 1 (and thereafter simply repeats the sequence 1, 2, 1, 2, ...). It has been observed that this happens for all initial values of *n* up to at least 10^{16}, but despite a fair amount of mathematical effort since the problem was first posed in the 1930s, no general proof for all values of *n* has ever been found. (For negative initial *n*, the evolution appears always to reach -1, -5 or -17, and then repeat with periods 1, 3 or 11 respectively.) An alternative formulation is to ask whether for all *n*

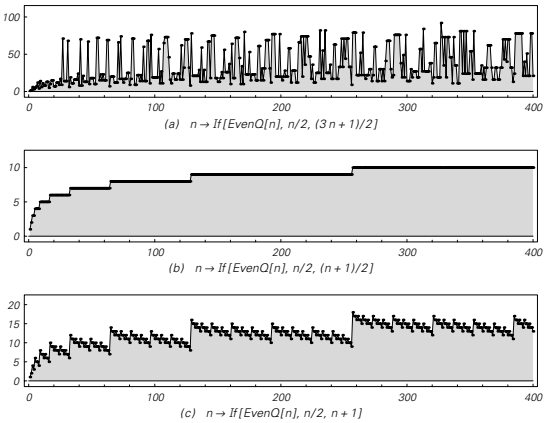
```
FixedPoint[(3#/2^IntegerExponent[#, 2] + 1)/2 &, n] == 2
```

With the rule *n → If[EvenQ[n], 5 n/2, (n + 1)/2]* used in the main text, the sequence produced repeats if *n* ever reaches 2, 4 or 40 (and possibly higher numbers). But with initial values of *n* up to 10,000, this happens in only 642 cases, and with values up to 100,000 it happens in only 2683 cases. In all other cases, the values of *n* in the sequence appear to grow forever.

To get some idea about the origin of this behavior, one can assume that successive values of *n* are randomly even and odd with equal probability. And with this assumption, *n* should increase by a factor of 5/2 half the time, and decrease by a factor close to 1/2 the rest of the time—so that after *t* steps it should be multiplied by an overall factor of about $(\sqrt{5}/2)^t$. Starting with *n = 6*, the effective exponents for *t = 10^Range[6]* are *{39.6, 245.1, 1202.8, 9250.7, 98269.8, 1002020.4}*. One reason that all sequences do not grow forever is that even with perfect randomness, there will be fluctuations, and occasionally *n* will reach a low value that makes it get stuck in a repetitive sequence.

If one applies the same kind of argument to the standard *3 n + 1* problem, then one concludes that *n* should on average decrease by a factor of $\sqrt{3}/2$ at each step, making it unsurprising that at least in most cases *n* eventually reaches the value 1. Indeed, averaging over many initial values of *n*, there is good quantitative agreement between the predictions of the randomness approximation and the actual *3 n + 1* problem. But since there is no fundamental basis for the randomness approximation, it is still conceivable that a particular value of *n* exists that does not follow its predictions.

The pictures below show how many steps are needed to reach value 1 starting from different values of *n*. Case (a) is the standard *3 n + 1* problem. Cases (b) and (c) use somewhat different rules that yield considerably simpler behavior. In case (b), the number of steps is equal to the number of base 2 digits in *n*, while in case (c) it is determined by the number of 1's in the base 2 digit sequence of *n*.



■ ***3n+1* problem as cellular automaton.** If one writes the digits of *n* in base 6, then the rule for updating the digit sequence is a cellular automaton with 7 possible colors (color 6 works as an end marker that appears to the left and right of the actual digit sequence):

```
{a_, b_, c_} → If[b == 6, If[EvenQ[a], 6, 4],
    3 Mod[a, 2] + Quotient[b, 2] /. 0 :> 6 /; a == 6]
```

The *3 n + 1* problem can then be viewed as a question about the existence of persistent structure in this cellular automaton.

■ **Reconstructing initial conditions.** Given a particular starting value of *n*, it is difficult to predict what precise sequence of even and odd values will be obtained in the system on page 122. But given *t* steps in this sequence as a list of 0's and 1's, the

following function will reconstruct the rightmost t digits in the starting value of n:

```
IntegerDigits[First[Fold[{Mod[If[OddQ[#2], 2 First[#1] - 1,
      2 First[#1] PowerMod[5, -1, Last[#1]]], Last[#1]],
    2 Last[#1]} &, {0, 2}, Reverse[list]]], 2, Length[list]]
```

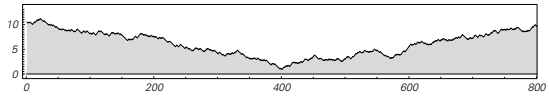
■ **A reversible system.** In both the ordinary $3n+1$ problem and in the systems discussed in the main text different numbers often evolve to the same value so that there is no unique way to reverse the evolution. However, with the rule

```
n → If[EvenQ[n], 3 n/2, Round[3 n/4]]
```

it is always possible to go backwards by the rule

```
n → If[Mod[n, 3] == 0, 2 n/3, Round[4 n/3]]
```

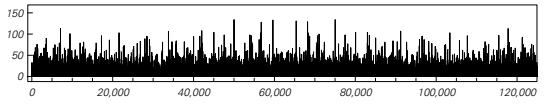
The picture shows the number of base 10 digits in numbers obtained by backward and forward evolution from $n = 8$. For $n < 8$, the system always enters a short cycle. Starting at $n = 44$, there is also a length 12 cycle. But apart from these cycles, the numbers produced always seem to grow without bound at an average rate of $3/(2\sqrt{2})$ in the forward direction, and $2\,4^{1/3}/3$ in the backward direction (at least all numbers up to 10,000 grow to above 10^{100}). Approximately one number in 20 has the property that evolution either backward or forward from it never leads to a smaller number.



■ **Page 125 · Reversal-addition systems.** The operation that is performed here is

```
n → n + FromDigits[Reverse[IntegerDigits[n, 2]], 2]
```

After a few steps, the digit sequence obtained is typically reversal symmetric (a generalized palindrome) except for the interchange of 0 and 1, and for the presence of localized structures. The sequence expands by at least one digit every two steps; more rapid expansion is typically correlated with increased randomness. For most initial n, the overall pattern obtained quickly becomes repetitive, with an effective period of 4 steps. But with the initial condition $n = 512$, no repetition occurs for at least a million steps, at which point n has 568418 base 2 digits. The plot below shows the lengths of the successive regions of regularity visible on the right-hand edge of the picture on page 126 over the course of the first million steps.



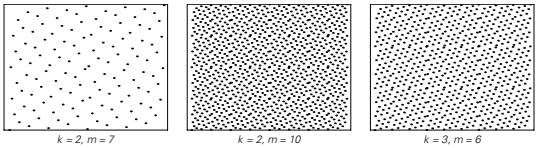
If one works directly with a digit sequence of fixed length, dropping any carries on the left, then a repetitive pattern is typically obtained fairly quickly. If one always includes one new digit on the left at every step, even when it is 0, then a rather random pattern is produced.

■ **History.** Systems similar to the one described here (though often in base 10) were mentioned in the recreational mathematics literature at least as long ago as 1939. A few small computer experiments were done around 1970, but no large-scale investigations seem to have previously been made.

■ **Digit reversal.** Sequences of the form

```
Table[FromDigits[
    Reverse[IntegerDigits[n, k, m]], k], {n, 0, k^m - 1}]
```

shown below appear in algorithms such as the fast Fourier transform and, with different values of k for different coordinates, in certain quasi-Monte Carlo schemes. (See pages 1073 and 1085.) Such sequences were considered by Johannes van der Corput in 1935.



■ **Iterated run-length encoding.** Starting say with $\{1\}$ consider repeatedly replacing *list* by (see page 1070)

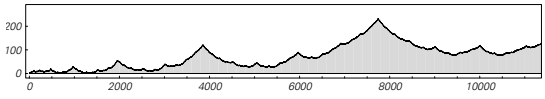
```
Flatten[Map[{Length[#], First[#]} &, Split[list]]]
```

The resulting sequences contain only the numbers 1, 2 and 3, but otherwise at first appear fairly random. However, as noticed by John Conway around 1986, the sequences can actually be obtained by a neighbor-independent substitution system, acting on 92 subsequences, with rules such as $\{3, 1, 1, 3, 3, 2, 2, 1, 1, 3\} \to \{\{1, 3, 2\}, \{1, 2, 3, 2, 2, 2, 1, 1, 3\}\}$. The system thus in the end produces patterns that are purely nested, though formed from rather complicated elements. The length of the sequence at the n^{th} step grows like λ^n, where $\lambda \simeq 1.3$ is the root of a degree 71 polynomial, corresponding to the largest eigenvalue of the transition matrix for the substitution system.

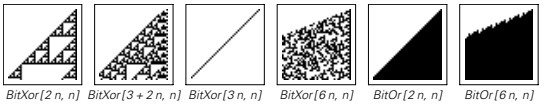
■ **Digit count sequences.** Starting say with $\{1\}$ repeatedly replace *list* by

```
Join[list, IntegerDigits[Apply[Plus, list], 2]]
```

The resulting sequences grow in length roughly like $n\,\text{Log}[n]$. The picture below shows the fluctuations around $m/2$ of the cumulative number of 1's up to position m in the sequence obtained at step 1000. A definite nested structure similar to picture (c) on page 130 is evident.



■ **Iterated bitwise operations.** The pictures below show digit sequences generated by repeatedly applying combinations of bitwise and arithmetic operations. The first example corresponds to elementary cellular automaton rule 60. Note that any cellular automaton rule can be reproduced by some appropriate combination of bitwise and arithmetic operations.



BitXor[2 n, n] BitXor[3 + 2 n, n] BitXor[3 n, n] BitXor[6 n, n] BitOr[2 n, n] BitOr[6 n, n]

Recursive Sequences

■ **Page 128 · Recurrence relations.** The rules for the sequences given here all have the form of linear recurrence relations. An explicit formula for the n^{th} term in each sequence can be found by solving the algebraic equation obtained by applying the replacement $f[m\_] \to t^m$ to the recurrence relation. (In case (e), for example, the equation is $t^n == -t^{n-1} + t^{n-2}$.) Note that (d) is the Fibonacci sequence, discussed on page 890.

Standard examples of recursive sequences that do not come from linear recurrence relations include factorial

```
f[1] = 1; f[n_] := n f[n - 1]
```

and Ackermann functions (see below). These two sequences both grow rapidly, but smoothly.

A recurrence relation like

```
f[0] = x; f[n_] := a f[n - 1] (1 - f[n - 1])
```

corresponds to an iterated map of the kind discussed on page 920, and has complicated behavior for many rational x.

■ **Ackermann functions.** A convenient example is

```
f[1, n_] := n; f[m_, 1] := f[m - 1, 2]

f[m_, n_] := f[m - 1, f[m, n - 1] + 1]
```

The original function constructed by Wilhelm Ackermann around 1926 is essentially

```
f[1, x_, y_] := x + y;
f[m_, x_, y_] := Nest[f[m - 1, x, #] &, x, y - 1]
```

or

```
f[m_, x_, y_] :=
  Nest[Function[z, Nest[#1, x, z - 1]] &, x + # &, m - 1][y]
```

For successive m (following the so-called Grzegorczyk hierarchy) this is $x + y$, $x\,y$, x^y, $\text{Nest}[x^{\#}\,\&, 1, y]$, ... `f[4, x, y]` can also be written `Array[x &, y, 1, Power]` and is sometimes called tetration and denoted $x\uparrow\uparrow y$.

■ **Page 129 · Computation of sequences.** It is straightforward to compute the various sequences given here, but to avoid a rapid increase in computer time, it is essential to store all the values of `f[n]` that one has already computed, rather than recomputing them every time they are needed. This is achieved for example by the definitions

```
f[n_] := f[n] = f[n - f[n - 1]] + f[n - f[n - 2]]

f[1] = f[2] = 1
```

The question of which recursive definitions yield meaningful sequences can depend on the details of how the rules are applied. For example, `f[-1]` may occur, but if the complete expression is `f[-1] - f[-1]`, then the actual value of `f[-1]` is irrelevant. The default form of evaluation for recursive functions implemented by all standard computer languages (including *Mathematica*) is the so-called leftmost innermost scheme, which attempts to find explicit values for each `f[k]` that occurs first, and will therefore never notice if `f[k]` in fact occurs only in the combination `f[k] - f[k]`. (The SMP system that I built around 1980 allowed different schemes—but they rarely seemed useful and were difficult to understand.)

■ **Page 131 · Properties of sequences.** Sequence (d) is given by

```
f[n_] := (n + g[IntegerDigits[n, 2]])/2

g[{(1)..}] = 1; g[{1, (0)..}] = 0

g[{1, s__}] := 1 + g[IntegerDigits[FromDigits[{s}, 2] + 1, 2]]
```

The list of elements in the sequence up to value m is given by

```
Flatten[Table[Table[n, {IntegerExponent[n, 2] + 1}], {n, m}]]
```

The differences between the first $2(2^k - 1)$ of these elements is

```
Nest[Replace[#, {x___} → {x, 1, x, 0}] &, {}, k]
```

The largest n for which `f[n] == m` is given by $2m + 1 - \text{DigitCount}[m, 2, 1]$ or $\text{IntegerExponent}[(2m)!, 2] + 1$ (this satisfies `h[1] = 2; h[m_] := h[Floor[m/2]] + m`).

The form of sequence (c) is similar to that obtained from concatenation numbers on page 913. Hump m in the picture of sequence (c) shown is given by

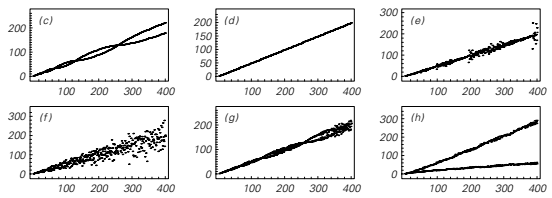
```
FoldList[Plus, 0, Flatten[Nest[Delete[NestList[Rest, #,
  Length[#] - 1], 2] &, Append[Table[1, {m}], 0], m]] - 1/2]
```

The first 2^m elements in the sequence can also be generated in terms of reordered base 2 digit sequences by

```
FoldList[Plus, 1, Map[Last[Last[#]] &,
    Sort[Table[({Length[#], Apply[Plus, #], 1 - #} &)[
        IntegerDigits[i, 2]], {i, 2^m}]]]]
```

Note that the positive and negative fluctuations in sequence (f) are not completely random: although the probability for individual fluctuations in each direction seems to be the same, the probability for two positive fluctuations in a row is smaller than for two negative fluctuations in a row.

In the sequences discussed here, `f[n_]` always has the form `f[p[n]] + f[q[n]]`. The plots at the top of the next page show `p[n]` and `q[n]` as a function of n.



The process of evaluating *f[n]* for a particular *n* can be thought of as yielding a tree where each node is a particular *f[k]* which has two successors, *f[p[k]]* and *f[q[k]]*. The distinct nodes reached starting from *f[12]* for sequence (f) are then for example *{{12}, {3, 7}, {1, 2, 4}, {1, 2}, {1}}*. The total lengths of these chains (corresponding to the depth of the evaluation tree) seem to increase roughly like *Log[n]* for all the rules on this page. For the Fibonacci sequence, it is instead *n – 1*. The maximum number of distinct nodes at any level in the tree has large fluctuations but its peaks seem to increase roughly linearly for all the rules on this page (in the Fibonacci case it is *Ceiling[n/2]*).

■ **History.** The idea of sequences in which later terms are deduced from earlier ones existed in antiquity, notably in the method of induction and in various approximation schemes (compare page 918). The Fibonacci sequence also appears to have arisen in antiquity (see page 890). A fairly clear idea of integer recurrence relations has existed since about the 1600s, but until very recently mainstream mathematics has almost never investigated them. In the late 1800s and early 1900s issues about the foundations of mathematics (see note below) led to the formal definition of so-called recursive functions. But almost without exception the emphasis was on studying what such functions could in principle do, not on looking at the actual behavior of particular ones. And indeed, despite their simple forms, recursive sequences of the kind I discuss here do not for the most part ever appear to have been studied before—although sequence (c) was mentioned in lectures by John Conway around 1988, and the first 17 terms of sequence (e) were given by Douglas Hofstadter in 1979.

■ **Primitive recursive functions.** As part of trying to formalize foundations of arithmetic Richard Dedekind began around 1888 to discuss possible functions that could be defined using recursion (induction). By the 1920s there had then emerged a definite notion of primitive recursive functions. The proof of Gödel's Theorem in 1931 made use of both primitive and general recursive functions—and by the mid-1930s emphasis had shifted to discussion of general recursive functions.

Primitive recursive functions are defined to deal with non-negative integers and to be set up by combining the basic functions *z = 0 &* (zero), *s = # + 1 &* (successor) and *p[i_] := Slot[i] &* (projection) using the operations of composition and primitive recursion

```
f[0, y___Integer] := g[y]
f[x_Integer, y___Integer] := h[f[x - 1, y], x - 1, y]
```

Plus and *Times* can then for example be defined as

```
plus[0, y_] = y; plus[x_, y_] := s[plus[x - 1, y]]
times[0, y_] = 0; times[x_, y_] := plus[times[x - 1, y], y]
```

Most familiar integer mathematical functions also turn out to be primitive recursive—examples being *Power*, *Mod*, *Binomial*, *GCD* and *Prime*. And indeed in the early 1900s it was thought that perhaps any function that could reasonably be computed would be primitive recursive (see page 1125). But the construction in the late 1920s of the Ackermann function *f[m, x, y]* discussed above showed that this was not correct. For any primitive recursive function can grow for large *x* at most like *f[m, x, x]* with fixed *m*. Yet *f[x, x, x]* will always eventually grow faster than this—demonstrating that the whole Ackermann function cannot be primitive recursive. (See page 1162.)

A crucial feature of primitive recursive functions is that the number of steps they take to evaluate is always limited, and can always in effect be determined in advance, since the basic operation of primitive recursion can be unwound simply as

```
f[x_, y___] := Fold[h[#1, #2, y] &, g[y], Range[0, x - 1]]
```

And what this means is that any computation that for example fundamentally involves a search that might not terminate cannot be implemented by a primitive recursive function. General recursive functions, however, also allow

```
μ[f_] = NestWhile[# + 1 &, 0, Function[n, f[n, ##1] ≠ 0]] &
```

which can perform unbounded searches. (Ordinary primitive recursive functions are always total functions, that give definite values for every possible input. But general recursive functions can be partial functions, that do not terminate for some inputs.) As discussed on page 1121 it turns out that general recursive functions are universal, so that they can be used to represent any possible computable function. (Note that any general recursive function can be expressed in the form *c[f, μ[g]]* where *f* and *g* are primitive recursive.)

In enumerating recursive functions it is convenient to use symbolic definitions for composition and primitive recursion

```
c[g_, h___] = Apply[g, Through[{h}[##]]] &
r[g_, h_] =
  If[#1 == 0, g[##2], h[#0[#1 - 1, ##2], #1 - 1, ##2]] &
```

where the more efficient unwound form is

```
r[g_, h_] = Fold[Function[{u, v}, h[u, v, ##2]],
                 g[##2], Range[0, # - 1]] &
```

And in terms of these, for example, *plus = r[p[1], s]*.

The total number of recursive functions grows roughly exponentially in the size (*LeafCount*) of such expressions, and roughly linearly in the number of arguments.

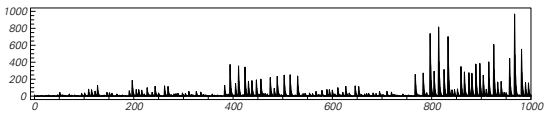
Most randomly selected primitive recursive functions show very simple behavior—either constant or linearly increasing when fed successive integers as arguments. The smallest examples that show other behavior are:

- *r[z, r[s, s]]*, which is *1/2 # (# + 1) &*, with quadratic growth
- *r[z, r[s, c[s, s]]]*, which is $2^{\#+1} - \# - 2$ &, with exponential growth
- *r[z, r[s, p[2]]]*, which is *2^Ceiling[Log[2, # + 2]] - # - 2 &*, which shows very simple nesting
- *r[z, r[c[s, z], z]]*, which is *Mod[#, 2] &*, with repetitive behavior
- *r[z, r[s, r[s, s]]]* which is *Fold[1/2 #1 (#1 + 1) + #2 &, 0, Range[#]] &*, growing like 2^{2^x}.

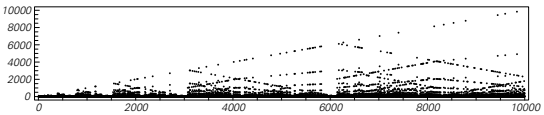
r[z, r[s, r[s, r[s, p[2]]]]] is the first function to show significantly more complex behavior, and indeed as the picture below indicates, it already shows remarkable randomness. From its definition, the function can be written as

```
Fold[Fold[2^Ceiling[Log[2, Ceiling[(#1 + 2)/(#2 + 2)]]]
    (#2 + 2) - 2 - #1 &, #2, Range[#1]] &, 0, Range[#]] &
```

Its first zeros are at *{4, 126, 813, 966, 1166, 1177, 1666, 1897}*.



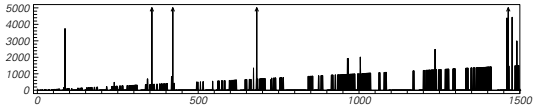
Each zero is immediately followed by a maximum equal to x, and as picture below shows, values tend to accumulate for example on lines of the form $\pm x/2^u \pm (2m+1)2^v$.



Note that functions of the form *Nest[r[c[s, z], #] &, c[s, s], n]* are given in terms of the original Ackermann function in the note above by *f[n + 1, 2, # + 1] - 1 &*.

Before the example above one might have thought that primitive recursive functions would always have to show rather simple behavior. But already an immediate counterexample is *Prime*. And it turns out that if they never sample values below *f[0]* the functions in the main text are also all primitive recursive. (Their definitions have a primitive recursive structure, but to operate correctly they must be given integers that are non-negative.)

Among functions with simple explicit definitions, essentially the only examples known fundamentally to be not primitive recursive are ones closely related to the Ackermann function. But given an enumeration of primitive recursive functions (say ordered first by *LeafCount*, then with *Sort*) in which the m^{th} function is *w[m]* diagonalization (see page 1128) yields the function *w[x][x]* shown below which cannot be primitive recursive. It is inevitable that the function shown must eventually grow faster than any primitive recursive function (at $x = 356$ its value is 63190, while at $x = 1464$ it is 1073844). But by reducing the results modulo 2 one gets a function that does not grow—and has seemingly quite random behavior—yet is presumably again not primitive recursive.

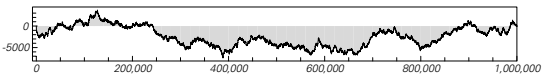


(Note that multiple arguments to a recursive function can be encoded as a single argument using functions like the β of page 1120—though the irregularity of such functions tends to make it difficult then to tell what is going on in the underlying recursive function.)

■ **Ulam sequences.** Slightly more complicated definitions in terms of numbers yield all sorts of sequences with very complicated forms. An example suggested by Stanislaw Ulam around 1960 (in a peculiar attempt to get a 1D analog of a 2D cellular automaton; see pages 877 and 928) starts with *{1, 2}*, then successively appends the smallest number that is the sum of two previous numbers in just one way, yielding

{1, 2, 3, 4, 6, 8, 11, 13, 16, 18, 26, 28, 36, 38, 47, 48, 53, 57, ...}

With this initial condition, the sequence is known to go on forever. At least up to $n = 10^6$ terms, it increases roughly like $13.5n$, but as shown below the fluctuations seem random.



The Sequence of Primes

■ **History of primes.** Whether the Babylonians had the notion of primes is not clear, but before 400 BC the Pythagoreans had introduced primes as numbers of objects that can be

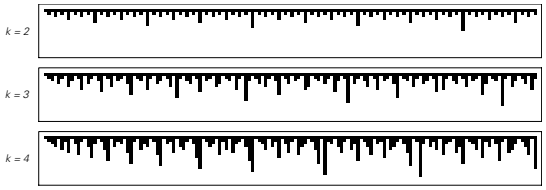
arranged only in a single line, and not in any other rectangular array. Around 300 BC Euclid discussed various properties of primes in his *Elements*, giving for example a proof that there are an infinity of primes. The sieve of Eratosthenes was described in 200 BC, apparently following ideas of Plato. Then starting in the early 1600s various methods for factoring were developed, and conjectures about formulas for primes were made. Pierre Fermat suggested $2^{2^n}+1$ as a source for primes and Marin Mersenne `2^Prime[n]-1` (see page 911). In 1752 Christian Goldbach showed that no ordinary polynomial could generate only primes, though as pointed out by Leonhard Euler n^2-n+41 does so for $n<40$. (With *If* or *Floor* included there are at least complicated cases known where polynomial-like formulas can be set up whose evaluation corresponds to explicit prime-generating procedures—see page 1162.) Starting around 1800 extensive work was done on analytical approximations to the distribution of primes (see below). There continued to be slow progress in finding specific large primes; $2^{31}-1$ was found prime around 1750 and $2^{127}-1$ in 1876. ($2^{2^5}+1$ was found composite in 1732, as have now all $2^{2^n}+1$ for $n \le 32$.) Then starting in the 1950s with the use of electronic computers many new large primes were found. The number of digits in the largest known prime has historically increased roughly exponentially with time over the past two decades, with a prime of over 4 million digits ($2^{13466917}-1$) now being known (see page 911).

■ **Page 132 · Finding primes.** The sieve of Eratosthenes shown in the picture is an appropriate procedure if one wants to find every prime, but testing whether an individual number is prime can be done much more efficiently, as in *PrimeQ[n]* in *Mathematica*, for example by using Fermat's so-called little theorem that $\text{Mod}[a^{p-1}, p] == 1$ whenever p is prime. The n^{th} prime *Prime[n]* can also be computed fairly efficiently using ideas from analytic number theory (see below).

■ **Decimation systems.** A somewhat similar system starts with a line of cells, then at each step removes every k^{th} cell that remains, as in the pictures below. The number of steps for which a cell at position n will survive can be computed as

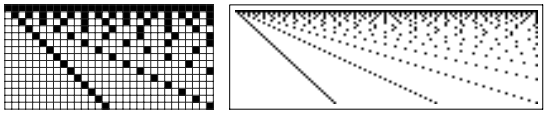
```
Module[{q = n + k - 1, s = 1},
  While[Mod[q, k] ≠ 0, q = Ceiling[(k - 1) q/k]; s++]; s]
```

If a cell is going to survive for s steps, then it turns out that this can be determined by looking at the last s digits in the base k representation of its position. For $k=2$, a cell survives for s steps if these digits are all 0 (so that s==*IntegerExponent[n, k]*). But for $k>2$, no such simple characterization appears to exist.



If the cells are arranged on a circle of size n, the question of which cell is removed last is the so-called Josephus problem. The solution is `Fold[Mod[#1 + k, #2, 1] &, 0, Range[n]]`, or `FromDigits[RotateLeft[IntegerDigits[n, 2]], 2]` for $k=2$.

■ **Page 132 · Divisors.** The picture below shows as black squares the divisors of each successive number (which correspond to the gray dots in the picture in the main text). Primes have divisors 1 and n only. (See also pages 902 and 747.)



■ **Page 133 · Results about primes.** *Prime[n]* is given approximately by $n\,\text{Log}[n]+n\,\text{Log}[\text{Log}[n]]$. (*Prime[$10^9$]* is 22,801,763,489 while the approximation gives 2.38×10^{10}.) A first approximation to *PrimePi[n]* is *n/Log[n]*. A somewhat better approximation is *LogIntegral[n]*, equal to *Integrate[1/Log[t], {t, 2, n}]*. This was found empirically by Carl Friedrich Gauss in 1792, based on looking at a table of primes. (*PrimePi[10^9]* is 50,847,534 while *LogIntegral[10^9]* is about 50,849,235.) A still better approximation is obtained by subtracting *Sum[LogIntegral[n^{r_i}], {i, -∞, ∞}]* where the r_i are the complex zeros of the Riemann zeta function *Zeta[s]*, discussed on page 918. According to the Riemann Hypothesis, the difference between *PrimePi[n]* and *LogIntegral[n]* is of order $\sqrt{n}\,\text{Log}[n]$. More refined analytical estimates of *PrimePi[n]* are good enough that they are used by *Mathematica* to compute *Prime[n]* for large n.

It is known that the ratio of the number of primes of the form $4k+1$ and $4k+3$ asymptotically approaches 1, but almost nothing has been proved about the fluctuations.

The gap between successive primes *Prime[n] - Prime[n - 1]* is thought to grow on average at most like $\text{Log}[\text{Prime}[n]]^2$. It is known that for sufficiently large n a gap of any size must exist. It is believed but not proved that there are an infinite number of "twin primes" with a gap of exactly 2.

■ **History of number theory.** Most areas of mathematics go from inception to maturity within at most a century. But in

number theory there are questions that were formulated more than 2000 years ago (such as whether any odd perfect numbers exist) that have still not been answered. Of the principles that have been established in number theory, a great many were first revealed by explicit experiments. From its inception in classical times, through its development in the 1600s to 1800s, number theory was largely separate from other fields of mathematics. But starting at the end of the 1800s, increasing connections were found to other areas of both continuous and discrete mathematics. And through these connections, sophisticated proofs of such results as Fermat's Last Theorem—open for 350 years—have been constructed. Long considered a rather esoteric branch of mathematics, number theory has in recent years grown in practical importance through its use in areas such as coding theory, cryptography and statistical mechanics. Properties of numbers and certain elementary aspects of number theory have also always played a central role in amateur and recreational mathematics. And as this chapter indicates, number theory can also be used to provide many examples of the basic phenomena discussed in this book.

■ **Page 134 · Tables of primes.** No explicit tables of primes appear to have survived from antiquity, but it seems likely that all primes up to somewhere between 5000 and 10000 were known. (In 348 BC, Plato mentioned divisors of 5040, and by 100 AD there is evidence that the fifth perfect number was known, requiring the knowledge that 8191 is prime.) In 1202 Leonardo Fibonacci explicitly gave as an example a list of primes up to 100. And by the mid-1600s there were printed tables of primes up to 100,000, containing as much data as in plots (c) and (d). In the 1700s and 1800s many tables of number factorizations were constructed; by the 1770s there was a table up to 2 million, and by the 1860s up to 100 million. A table of primes up to a trillion could now be generated fairly easily with current computer technology—though for most purposes computation of specific primes is more useful.

■ **Page 134 · Numbers of primes.** The fact that curve (c) must cross the axis was proved by John Littlewood in 1914, and it is known to have at least one crossing below 10^{317}. Somewhat related to the curves shown here is the function `MoebiusMu[n]`, equal to 0 if n has a repeated prime factor and otherwise `(-1)^Length[FactorInteger[n]]`. The quantity `FoldList[Plus, 0, Table[MoebiusMu[i], {i, n}]]` behaves very much like a random walk. The so-called Mertens Conjecture from 1897 stated that the magnitude of this quantity is less than $\sqrt{n}$. But this was disproved in 1983, although the necessary n is not known explicitly.

■ **Relative primes.** A single number is prime if it has no non-trivial factors. Two numbers are said to be relatively prime if they share no non-trivial factors. The pattern formed by numbers with this property is shown on page 613.

■ **Page 135 · Properties.** (a) The number of divisors of n is given by `DivisorSigma[0, n]`, equal to `Length[Divisors[n]]`. For large n this number is on average of order `Log[n] + 2 EulerGamma - 1`.

(b) *(Aliquot sums)* The quantity that is plotted is `DivisorSigma[1, n] - 2 n`, equal to `Apply[Plus, Divisors[n]] - 2 n`. This quantity was considered of great significance in antiquity, particularly by the Pythagoreans. Numbers were known as abundant, deficient or perfect depending on whether the quantity was positive, negative or zero. (See notes on perfect numbers below.) For large n, `DivisorSigma[1, n]` is known to grow at most like `Log[Log[n]] n Exp[EulerGamma]`, and on average like $\pi^2 n/6$ (see page 1093). As discovered by Srinivasa Ramanujan in 1918 its fluctuations (see below) can be obtained from the formula

```
1/6 π^2 n Sum[Apply[Plus, Cos[2 π n Select[
                Range[s], GCD[s, #] == 1 &]/s]]/s^2, {s, ∞}]
```

(c) Squares are taken to be of positive or negative integers, or zero. The number of ways of expressing an integer n as the sum of two such squares is `4 Apply[Plus, Im[i^Divisors[n]]]`. This is nonzero when all prime factors of n of the form $4k+3$ appear with even exponents. There is no known simple formula for the number of ways of expressing an integer as a sum of three squares, although part of the condition in the main text for integers to be expressible in this way was established by René Descartes in 1638 and the rest by Adrien Legendre in 1798. Note that the total number of integers less than n which can be expressed as a sum of three squares increases roughly like $5n/6$, with fluctuations related to `IntegerDigits[n, 4]`. It is known that the directions of all vectors `{x, y, z}` for which $x^2 + y^2 + z^2 == n$ are uniformly distributed in the limit of large n.

The total number of ways that integers less than n can be expressed as a sum of d squares is equal to the number of integer lattice points that lie inside a sphere of radius $\sqrt{n}$ in d-dimensional space. For $d = 2$, this approaches πn for large n, with an error of order n^c, where $1/4 < c \lesssim 0.315$.

(d) All numbers n can be expressed as the sum of four squares, in exactly `8 Apply[Plus, Select[Divisors[n], Mod[#, 4] ≠ 0 &]]` ways, as established by Carl Jacobi in 1829. Edward Waring stated in 1770 that any number can be expressed as a sum of at most 9 cubes and 19 fourth powers. Seven cubes appear to suffice for all but 17 numbers, the last of which is 455; four

cubes may suffice for all but 113936676 numbers, the last of which is 7373170279850. (See also page 1166.)

(e) Goldbach's Conjecture has been verified for all even numbers up to 4×10^{14}. In 1973 it was proved that any even number can be written as the sum of a prime and a number that has at most two prime factors, not necessarily distinct. The number of ways of writing an integer n as a sum of two primes can be calculated explicitly as *Length[Select[n - Table[Prime[i], {i, PrimePi[n]}], PrimeQ]]*. This quantity was conjectured by G. H. Hardy and John Littlewood in 1922 to be proportional to

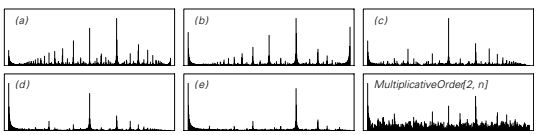
```
2 n Apply[Times, Map[(# - 1)/(# - 2) &,
        Map[First, Rest[FactorInteger[n]]]]]/Log[n]^2
```

It was proved in 1937 by Ivan Vinogradov that any large odd integer can be expressed as a sum of three primes.

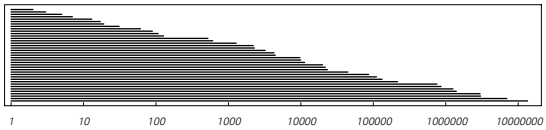
■ **Trapezoidal primes.** If one lays out n objects in an $a \times b$ rectangular array, then n is prime if either a or b must be 1. Following the Pythagorean idea of figurate numbers one can instead consider laying out objects in an array of b rows, containing successively a, $a-1$, … objects. It turns out all numbers except powers of 2 can be represented this way.

■ **Other integer functions.** *IntegerExponent[n, k]* gives nested behavior as for decimation systems on page 909, while *MultiplicativeOrder[k, n]* and *EulerPhi[n]* yield more complicated behavior, as shown on pages 257 and 1093.

■ **Spectra.** The pictures below show frequency spectra obtained from the sequences in the main text. Some regularity is evident, and in cases (a) and (b) it can be understood from trigonometric sum formulas of Ramanujan discussed above (see also pages 586 and 1081).

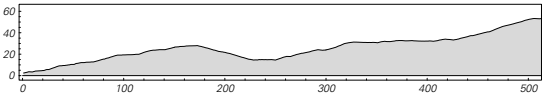


■ **Perfect numbers.** Perfect numbers with the property that *Apply[Plus, Divisors[n]] == 2 n* have been studied since at least the time of Pythagoras around 500 BC. The first few perfect numbers are *{6, 28, 496, 8128, 33550336}* (a total of 39 are currently known). It was shown by Euclid in 300 BC that $2^{n-1}(2^n - 1)$ is a perfect number whenever $2^n - 1$ is prime. Leonhard Euler then proved around 1780 that every even perfect number must have this form. The values of n for the known Mersenne primes $2^n - 1$ are shown below. These values can be found using the so-called Lucas-Lehmer test *Nest[Mod[#^2 - 2, 2^n - 1] &, 4, n - 2] == 0*, and in all cases n itself must be prime.



Whether any odd perfect numbers exist is probably the single oldest unsolved problem in mathematics. It is known that any odd perfect number must be greater than 10^{300}, must have a factor of at least 10^6, and must be less than 4^{4^s} if it has only s prime factors. Looking at curve (b) on page 135, however, it does not seem inconceivable that an odd perfect number could exist. For odd n up to 500 million the only values near 0 that appear in the curve are *{-6, -5, -4, -2, -1, 6, 18, 26, 30, 36}*, with, for example, the first 6 occurring at $n = 8925$ and last 18 occurring at $n = 159030135$. Various generalizations of perfect numbers have been considered, requiring for example *IntegerQ[DivisorSigma[1, n]/n]* (pluperfect) or *Abs[DivisorSigma[1, n] - 2 n] < r* (quasiperfect).

■ **Iterated aliquot sums.** Related to case (b) above is a system which repeats the replacement $n \to$ *Apply[Plus, Divisors[n]] - n* or equivalently $n \to$ *DivisorSigma[1, n] - n*. The fixed points of this procedure are the perfect numbers (see above). Other numbers usually evolve to perfect numbers, or to short repetitive sequences of numbers. But if one starts, for example, with the number 276, then the picture below shows the number of base 10 digits in the value obtained at each step.



After 500 steps, the value is the 53-digit number

39448887705043893375102470161238803295318090278129552

The question of whether such values can increase forever was considered by Eugène Catalan in 1887, and has remained unresolved since.

Mathematical Constants

■ **Page 137 · Digits of pi.** The digits of π shown here can be obtained in less than a second from *Mathematica* on a typical current computer using *N[π, 7000]*. Historically, the number of decimal digits of π that have been computed is roughly as follows: 2000 BC (Babylonians, Egyptians): 2 digits; 200 BC (Archimedes): 5 digits; 1430 AD: 14 digits; 1610: 35 digits; 1706: 100 digits; 1844: 200 digits; 1855: 500 digits; 1949 (ENIAC computer): 2037 digits; 1961: 100,000 digits (IBM 7090); 1973: 1 million; 1983: 16 million; 1989: 1 billion; 1997: 50 billion; 1999: 206 billion. In the first 200

billion digits, the frequencies of 0 through 9 differ from 20 billion by

```
{30841, -85289, 136978, 69393, -78309,
  -82947, -118485, -32406, 291044, -130820}
```

An early approximation to π was

```
4 Sum[(-1)^k/(2 k + 1), {k, 0, m}]
```

30 digits were obtained with

```
2 Apply[Times, 2/Rest[NestList[Sqrt[2 + #] &, 0, m]]]
```

An efficient way to compute π to n digits of precision is

```
(#[[2]]^2/#[[3]] &)[NestWhile[Apply[Function[{a, b, c, d},
      {(a + b)/2, Sqrt[a b], c - d (a - b)^2, 2 d}], #] &,
  {1, 1/Sqrt[N[2, n]], 1/4, 1/4}, #[[1]] ≠ #[[2]] &]]
```

This requires about *Log[2, n]* steps, or a total of roughly $n \, \text{Log}[n]^2$ operations (see page 1134).

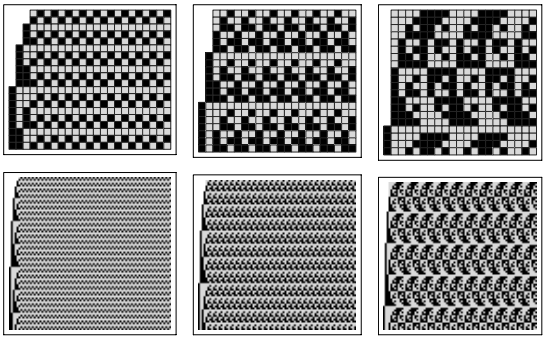
■ **Computing n^{th} digits directly.** Most methods for computing mathematical constants progressively generate each additional digit. But following work by Simon Plouffe and others in 1995 it became clear that it is sometimes possible to generate, at least with overwhelming probability, the n^{th} digit without explicitly finding previous ones. As an example, the n^{th} digit of *Log[2]* in base 2 is formally given by *Round[FractionalPart[2^n Sum[2^-k/k, {k, ∞}]]]*. And in practice the n^{th} digit can be found just by computing slightly over n terms of the sum, according to

```
Round[FractionalPart[
    Sum[FractionalPart[PowerMod[2, n - k, k]/k], {k, n}] +
      Sum[2^(n-k)/k, {k, n + 1, n + d}]]]
```

where several values of d can be tried to check that the result does not change. (Note that with finite-precision arithmetic, some exponentially small probability exists that truncation of numbers will lead to incorrect results.) The same basic approach as for *Log[2]* can be used to obtain base 16 digits in π from the following formula for π:

```
Sum[16^-k (4/(8 k + 1) - 2/(8 k + 4) -
      1/(8 k + 5) - 1/(8 k + 6)), {k, 0, ∞}]
```

A similar approach can also be used for many other constants that can be viewed as related to values of *PolyLog*.



■ **Page 139 · Rational numbers.** The pictures above show the base 2 digit sequences of numbers m/n for successive m.

The digits of $1/n$ in base b repeat with period

```
MultiplicativeOrder[b, FixedPoint[#/GCD[#, b] &, n]]
```

which is equal to *MultiplicativeOrder[b, n]* for prime n, and is at most $n - 1$. Each repeating block of digits typically seems quite random, and has properties such as all possible subblocks of digits up to a certain length appearing (see page 1084).

■ **Page 139 · Digit sequence properties.** Empirical evidence for the randomness of the digit sequences of $\sqrt{n}$, π, etc. has been accumulating since early computer experiments in the 1940s. The evidence is based on applying various standard statistical tests of randomness, and remains somewhat haphazard. (Already in 1888 John Venn had noted for example that the first 707 digits of π lead to an apparently typical 2D random walk.) (See page 1089.)

The fact that $\sqrt{2}$ is not a rational number was discovered by the Pythagoreans. Numbers that arise as solutions of polynomial equations are called algebraic; those that do not are called transcendental. e and π were proved to be transcendental in 1873 and 1882 respectively. It is known that *Exp[n]* and *Log[n]* for whole numbers n (except 0 and 1 respectively) are transcendental. It is also known for example that *Gamma[1/3]* and *BesselJ[0, n]* are transcendental. It is not known for example whether *EulerGamma* is even irrational.

A number is said to be "normal" in a particular base if every digit and every block of digits of any length occur with equal frequency. Note that the fact that a number is normal in one base does not imply anything about its normality in another base (unless the bases are related for example by both being powers of 2). Despite empirical evidence, no number expressed just in terms of standard mathematical functions has ever been rigorously proved to be normal. It has nevertheless been known since the work of Emile Borel in 1909 that numbers picked randomly on the basis of their value are almost always normal. And indeed with explicit constructions in terms of digits, it is quite straightforward to get numbers that are normal. An example of this is the number 0.1234567891011121314... obtained by concatenating the digits of successive integers in base 10 (see below). This number was discussed by David Champernowne in 1933, and is known to be transcendental. A few other results are also known. One based on gradual extension of work by Richard Stoneham from 1971 is that numbers of the form *Sum[1/(p^n b^(p^n)), {n, ∞}]* for prime $p > 2$ are normal in base b (for *GCD[b, p] == 1*), and are transcendental.

■ **Page 141 · Square roots.** A standard way to compute $\sqrt{n}$ is Newton's method (actually used already in 2000 BC by the Babylonians), in which one takes an estimate of the value x and then successively applies the rule $x \to 1/2\,(x + n/x)$. After t steps, this method yields a result accurate to about t^2 digits.

Another approach to computing square roots is based on the fact that the ratio of successive terms in for example the sequence $f[i] = 2\,f[i-1] + f[i-2]$ with $f[1] = f[2] = 1$ tends to $1 + \sqrt{2}$. This method yields about $2.5\,t$ base 2 digits after t steps.

The method of computing square roots shown in the main text is less efficient (it computes t digits in t steps), but illustrates more of the mechanisms involved. The basic idea is at every step t to maintain the relation $s^2 + 4\,r == 4^t\,n$, keeping r as small as possible so as to make $s \le 2^t\,\sqrt{n} < s + 4$. Note that the method works not only for integers, but for any rational number n for which $1 \le n < 4$.

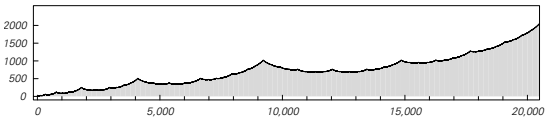
■ **Nested digit sequences.** The number obtained from the substitution system $\{1 \to \{1, 0\}, 0 \to \{0, 1\}\}$ is approximately 0.587545966 in base 10. It is certainly conceivable that a quantity such as Feigenbaum's constant (approximately 4.6692016091) could have a digit sequence with this kind of nested structure.

From the result on page 890, the number whose digits are obtained from $\{1 \to \{1, 0\}, 0 \to \{1\}\}$ is given by `Sum[2^(-Floor[n GoldenRatio]), {n, ∞}]`. This number is known to be transcendental. The n^{th} term in its continued fraction representation turns out to be `2^Fibonacci[n-2]`.

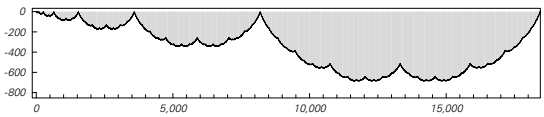
The fact that nested digit sequences do not correspond to algebraic numbers follows from work by Alfred van der Poorten and others in the early 1980s. The argument is based on showing that an algebraic function always exists for which the coefficients in its power series correspond to any given nested sequence when reduced modulo some p. (See page 1092.) But then there is a general result that if a particular sequence of power series coefficients can be obtained from an algebraic (but not rational) function modulo a particular p, then it can only be obtained from transcendental functions modulo any other p—or over the integers.

■ **Concatenation sequences.** One can consider forming sequences by concatenating digits of successive integers in base k, as in `Flatten[Table[IntegerDigits[i, k], {i, n}]]`. In the limit, such sequences contain with equal frequency all possible blocks of any given length, but as shown on page 597, they exhibit other obvious deviations from randomness. The picture below shows the $k = 2$ sequence chopped into length 256 blocks.

Applying `FoldList[Plus, 0, 2 list - 1]` to the whole sequence yields the pattern shown below.



The systematic increase is a consequence of the leading 1 in each concatenated sequence. Dropping this 1 yields the pattern below.



This is similar to picture (c) on page 131, and is a digit-by-digit version of

```
FoldList[Plus, 0,
  Table[Apply[Plus, 2 Rest[IntegerDigits[i, 2]] - 1], {i, n}]]
```

Note that although the picture above has a nested structure, the original concatenation sequences are not nested, and so cannot be generated by substitution systems. The element at position n in the first sequence discussed above can however be obtained in about `Log[n]` steps using

```
((IntegerDigits[#3 + Quotient[#1, #2], 2][[
        Mod[#1, #2] + 1]] &)[n - (# - 2) 2^(#-1) - 2, #,
    2^(#-1)] &)[NestWhile[# + 1 &, 0, (# - 1) 2^# + 1 < n &]]
```

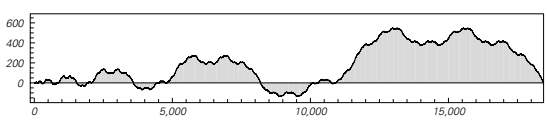
where the result of the `NestWhile` can be expressed as

```
Ceiling[1 + ProductLog[1/2 (n - 1) Log[2]]/Log[2]]
```

Following work by Maxim Rytin in the late 1990s about k^{n+1} digits of a concatenation sequence can be found fairly efficiently from

```
k/(k - 1)^2 -
  (k - 1) Sum[k^((k^s-1) (1+s-s k)/(k-1)) (1/((k - 1) (k^s - 1)^2) -
        k/((k - 1) (k^(s+1) - 1)^2) + 1/(k^(s+1) - 1)), {s, n}]
```

Concatenation sequences can also be generated by joining together digits from other representations of numbers; the picture below shows results for the Gray code representation from page 901.



■ **Specially constructed transcendental numbers.** Numbers known to be transcendental include ones whose digit sequences contain 1's only at positions $n!$, 2^n or *Fibonacci[n]*. Concatenation sequences, as well as generalizations formed by concatenating values of polynomials at successive integer points, are also known to yield numbers that are transcendental.

■ **Runs of digits.** One can consider any base 2 digit sequence as consisting of successive runs of 0's and 1's, constructed from the list of run lengths by

```
Fold[Join[#1, Table[1 - Last[#1], {#2}]] &, {0}, list]
```

This representation is related to so-called surreal numbers (though with the first few digits different). The number with run lengths corresponding to successive integers (so that the n^{th} digit is *Mod[Floor[1/2 + Sqrt[2 n]], 2]*) turns out to be $(1 - 2^{1/4}\ \text{EllipticTheta}[2, 0, 1/2] + \text{EllipticTheta}[3, 0, 1/2])/2$, and appears at least not to be algebraic.

■ **Leading digits.** Even though in individual numbers generated by simple mathematical procedures all possible digits often appear to occur with equal frequency, leading digits in sequences of numbers typically do not. Instead it is common for a leading digit s in base b to occur with frequency *Log[b, (s + 1)/s]* (so that in base 10 1's occur 30% of the time and 9's 4.5%). This will happen whenever *FractionalPart[Log[b, a[n]]]* is uniformly distributed, which, as discussed on page 903, is known to be true for sequences such as r^n (with *Log[b, r]* irrational), n^n, $n!$, *Fibonacci[n]*, but not $r\,n$, *Prime[n]* or *Log[n]*. A logarithmic law for leading digits is also found in many practical numerical tables, as noted by Simon Newcomb in 1881 and Frank Benford in 1938.

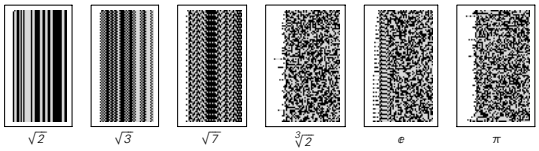
■ **Page 143 · Continued fractions.** The first n terms in the continued fraction representation for a number x can be found from the built-in *Mathematica* function *ContinuedFraction*, or from

```
Floor[NestList[1/Mod[#, 1] &, x, n - 1]]
```

A rational approximation to the number x can be reconstructed from the continued fraction using *FromContinuedFraction* or by

```
Fold[1/#1 + #2 &, Last[list], Rest[Reverse[list]]]
```

The pictures below show the digit sequences of successive iterates obtained from *NestList[1/Mod[#, 1] &, x, n]* for several numbers x.



Unlike ordinary digits, the individual terms in a continued fraction can be of any size. In the continued fraction for a randomly chosen number, the probability to find a term of size s is *Log[2, (1 + 1/s)/(1 + 1/(s + 1))]*, so that the probability of getting a 1 is about 41.50%, and the probability of getting a large term falls off like $1/s^2$. If one looks at many terms, then their geometric mean is finite, and approaches Khinchin's constant $\text{Khinchin} \approx 2.68545$.

In the first 1000 terms of the continued fraction for π, there are 412 1's, and the geometric mean is about 2.6656. The largest individual term is the 432th one, which is equal to 20,776. In the first million terms, there are 414,526 1's, the geometric mean is 2.68447, and the largest term is the 453,294th one, which is 12,996,958.

Note that although the usual continued fraction for π looks quite random, modified forms such as

```
4/(Fold[#2/#1 + 2 &, 2, Reverse[Range[1, n, 2]^2]] - 1)
```

can be very regular.

The continued fractions for *Exp[2/k]* and *Tan[k/2]* have simple forms (as discussed by Leonhard Euler in the mid-1700s); other rational powers of e and tangents do not appear to. The sequence of odd numbers gives the continued fraction for *Coth[1]*; the sequence of even numbers for *BesselI[0, 1]/BesselI[1, 1]*. In general, continued fractions whose n^{th} term is $a\,n + b$ correspond to numbers given by *BesselI[b/a, 2/a]/BesselI[b/a + 1, 2/a]*. Numbers whose continued fraction terms are polynomials in n can presumably also be represented in terms of suitably generalized hypergeometric functions. (All so-called Hurwitz numbers have continued fractions that consist of interleaved polynomial sequences—a property left unchanged by $x \to (a\,x + b)/(c\,x + d)$.)

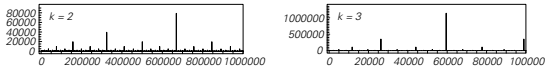
As discovered by Jeffrey Shallit in 1979, numbers of the form *Sum[1/k^i, {i, 0, ∞}]* that have nonzero digits in base k only at positions 2^i turn out to have continued fractions with terms of limited size, and with a nested structure that can be found using a substitution system according to

```
{0, k - 1, k + 2, k, k, k - 2, k, k + 2, k - 2, k}[[
  Nest[Flatten[{{1, 2}, {3, 4}, {5, 6}, {7, 8}, {5, 6}, {3, 4},
      {9, 10}, {7, 8}, {9, 10}, {3, 4}}[[#]]] &, 1, n]]
```

The continued fractions for square roots are always periodic; for higher roots they never appear to show any significant regularities. The first million terms in the continued fraction for $2^{1/3}$ contain 414,983 1's, have geometric mean 2.68505, and have largest term 4,156,269 at position 484,709. Terms of any size presumably in the end always occur in continued fractions for higher roots, though this is not known for certain. Fairly large terms are sometimes seen quite early: in $5^{1/3}$ term 19 is 3052, while in *Root[10 + 8 # - #^3 &, 1]* term 34

is 1,501,790. The presence of a large term indicates a close approximation to a rational number. In a few known cases simple formulas yield numbers that are close but not equal to integers. An example discovered by Srinivasa Ramanujan around 1913 is *Exp[π √163]*, which is an integer to one part in 10^{30}, and has second continued fraction term 1,333,462,407,511. (This particular example can be understood from the fact that as *d* increases *Exp[π √d]* becomes extremely close to *-1728 KleinInvariantJ[(1 + √-d)/2]*, which turns out to be an integer whenever there is unique factorization of numbers of the form $a + b\sqrt{-d}$ —and *d = 163* is the largest of the 9 cases for which this is so.) Other less spectacular examples include *Exp[π] - π* and *163/Log[163]*.

Numbers with digits given by concatenation sequences in any base *k* (see note above) seem to have unusual continued fractions, in which most terms are fairly small, but some are extremely large. Thus with *k = 2*, term 30 is 4,534,532, term 64 is 4,682,854,730,443,938, term 152 is about 2×10^{34} and term 669,468 is about 2×10^{78902}. (For the *k = 10* case of the original Champernowne number, even term 18 is already about 5×10^{165}.) The plots below of the numbers of digits in successive terms turn out to have patterns of peaks that show some signs of nesting.



In analogy to digits in a concatenation sequence the terms in the sequence

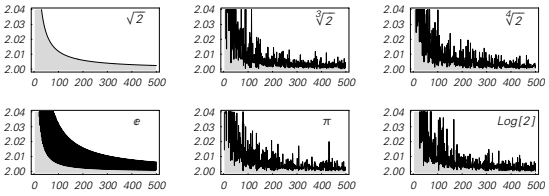
```
Flatten[Table[Rest[ContinuedFraction[a/b]],
                                {b, 2, n}, {a, b - 1}]]
```

are known to occur with the same frequencies as they would in the continued fraction representation for a randomly chosen number.

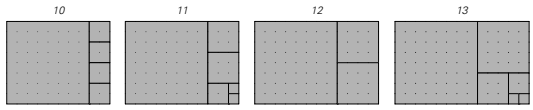
The pictures below show as a function of *n* the quantity

```
With[{r = FromContinuedFraction[ContinuedFraction[x, n]]},
                        -Log[Denominator[r], Abs[x - r]]]
```

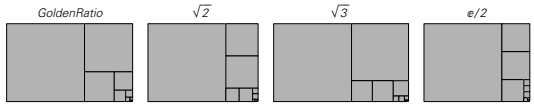
which gives a measure of the closeness of successive rational approximations to *x*. For any irrational number this quantity cannot be less than 2, while for algebraic irrationals Klaus Roth showed in 1955 that it can only have finitely many peaks that reach above any specified level.



▪ **History.** Euclid's algorithm states that starting from integers *{a, b}* iterating *{a_, b_} :→ If[a > b, {a - b, b}, {a, b - a}]* eventually leads to *{GCD[a, b], 0}*. (See page 1093.) The pictures below show how this works. The numbers of successively smaller squares (corresponding to the numbers of steps in the algorithm) turn out to be exactly *ContinuedFraction[a/b]*.



It was discovered in antiquity that Euclid's algorithm starting with *{x, 1}* terminates only when *x* is rational. In all cases, however, the relationship with continued fractions remains, as below.



Infinite continued fractions appear to have first been explicitly written down in the mid-1500s, and to have become popular in many problems in number theory by the 1700s. Leonhard Euler studied many continued fractions, while Joseph Lagrange seems to have thought that it might be possible to recognize any algebraic number from its continued fraction. The periodicity of continued fractions for quadratic irrationals was proved by Evariste Galois in 1828. From the late 1800s interest in continued fractions as such waned; it finally increased again in the 1980s in connection with problems in dynamical systems theory.

▪ **Egyptian fractions.** Following the ancient Egyptian number system, rational numbers can be represented by sums of reciprocals, as in *3/7 == 1/3 + 1/11 + 1/231*. With suitable distinct integers *a[n]* one can represent any number by *Sum[1/a[n], {n, ∞}]*. The representation is not unique; *a[n] = 2^n*, *n(n + 1)* and *(n + 1)!/n* all yield *1*. Simple choices for *a[n]* yield many standard transcendental numbers: *n!*: *e - 1*; *n!^2*: *BesselI[0, 2] - 1*; *n 2^n*: *Log[2]*; *n^2*: *π^2/6*; *(2 n - 1)(2 n - 3)*: *π √3 /9*; *3 - 16 n + 16 n^2*: *π/8*; *n n!*: *ExpIntegralEi[1] - EulerGamma*. (See also page 902.)

▪ **Nested radicals.** Given a list of integers acting like digits one can consider representing numbers in the form *Fold[Sqrt[#1 + #2] &, 0, Reverse[list]]*. A sequence of identical digits *d* then corresponds to the number *(1 + Sqrt[4 d + 1])/2*. (Note that *Nest[Sqrt[# + 2] &, 0, n] == 2 Cos[π/2^(n+1)]*.) Repeats of a digit block *b* give numbers that solve *Fold[#1^2 - #2 &, x, b] == x*. It appears that digits 0, 1, 2 are

sufficient to represent uniquely all numbers between 1 and 2. For any number x the first n digits are given by

```
Ceiling[NestList[(2 - Mod[-#, 1])^2 &, x^2, n - 1] - 2]
```

Even rational numbers such as $3/2$ do not yield simple digit sequences. For random x, digits 0, 1, 2 appear to occur with limiting frequencies $\sqrt{2+d} - \sqrt{1+d}$.

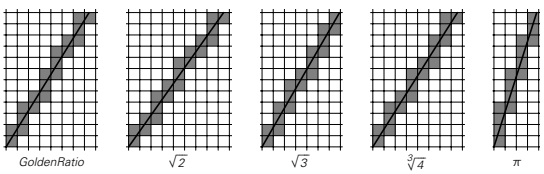
■ **Digital slope representation.** One can approximate a line of any slope h as in the picture below by a sequence of segments on a square grid (such as a digital display device). The vertical distance moved at the n^{th} horizontal position is $\text{Floor}[n\,h] - \text{Floor}[(n-1)\,h]$, and the sequence obtained from this (which contains only terms $\text{Floor}[h]$ and $\text{Floor}[h]+1$) provides a unique representation for h. As discussed on page 903 this sequence can be generated by applying substitution rules derived from the continued fraction form of h. If h is rational, the sequence is repetitive, while if h is a quadratic irrational, it is nested. Given a sequence of length n, an approximation to h can be reconstructed using

```
Max[MapIndexed[#1/First[#2] &,
    FoldList[Plus, First[list], Rest[list]]]]
```

The fractional part of the result obtained is always an element of the Farey sequence

```
Union[Flatten[Table[a/b, {b, n}, {a, 0, b}]]]
```

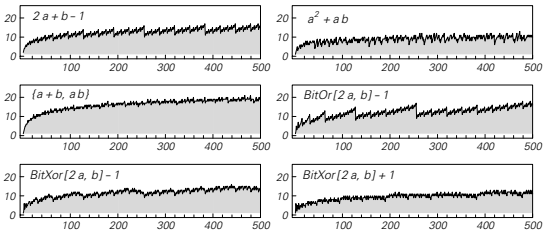
(See also pages 892, 932 and 1084.)



■ **Representations for integers.** See page 560.

■ **Operator representations.** Instead of repeatedly applying an operation to a sequence of digits one can consider forming integers (or other numbers) by performing trees of operations on a single constant. Thus, for example, any integer m can be obtained by a tree of $m-1$ additions of 1's such as $(1+(1+1))+1$. Another operator that can be used to generate any integer is $a \circ b = 2a+b-1$. In this case 6 is $(1\circ(1\circ 1))\circ 1$, and an integer m can be obtained by $\text{Tr}[1 + \text{IntegerDigits}[m, 2]] - 2$ or at most $\text{Log}[2, m]$ applications of $\circ$. The operator $k\,a+b-k+1$ can be used for any k. It also turns out that $\text{BitXor}[2\,a, b]+1$ works, though in this case even for 2 the smallest representation is $(1\circ 1)\circ(1\circ((1\circ 1)\circ 1))$. (For $\text{BitOr}[2\,a, b]-1$ the number of applications needed is $\text{With}[\{i = \text{IntegerDigits}[m, 2]\}, \text{Tr}[i + 1] + i[[2]]\,(1 + i[[3]]) - 1]$.)

The pictures below show the smallest number of operator applications required for successive integers. With the pair of operators $a+b$ and $a \times b$ (a case considered in recreational mathematics for n-ary operators) numbers of the form 3^s have particularly small representations. Note that in all cases the size of the smallest representation must at some level increase like $\text{Log}[m]$ (compare pages 1067 and 1070), but there may be some "algorithmically simple" integers that have shorter representations.



■ **Number classification.** One can imagine classifying real numbers in terms of what kinds of operations are needed to obtain them from integers. Rational numbers require only division (or solving linear equations), while algebraic numbers require solving polynomial equations. Rather little is known about numbers that require solving transcendental equations—and indeed it can even be undecidable (see page 1138) whether two equations can yield the same number. Starting with integers and then applying arithmetic operations and fractional powers one can readily reproduce all algebraic numbers up to degree 4, but not beyond. The sets of numbers that can be obtained by applying elementary functions like Exp, Log and Sin seem in various ways to be disjoint from algebraic numbers. But if one applies multivariate elliptic or hypergeometric functions it was established in the late 1800s and early 1900s that one can in principle reach any algebraic number. One can also ask what numbers can be generated by integrals (or by solving differential equations). For rational functions $f[x]$, $\text{Integrate}[f[x], \{x, 0, 1\}]$ must always be a linear function of Log and ArcTan applied to algebraic numbers ($f[x] = 1/(1+x^2)$ for example yields $\pi/4$). Multiple integrals of rational functions can be more complicated, as in

```
Integrate[1/(1 + x^2 + y^2), {x, 0, 1}, {y, 0, 1}] ==
  HypergeometricPFQ[{1/2, 1, 1}, {3/2, 3/2}, 1/9]/6 +
    1/2 π ArcSinh[1] - Catalan
```

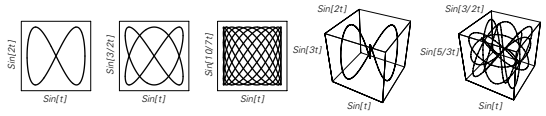
and presumably often cannot be expressed at all in terms of standard mathematical functions. Integrals of rational functions over regions defined by polynomial inequalities have recently been discussed under the name "periods". Many numbers associated with Zeta and Gamma can readily be generated, though apparently for example e and EulerGamma cannot. One can also consider numbers obtained from infinite sums (or by solving recurrence

equations). If *f[n]* is a rational function, *Sum[f[n], {n, ∞}]* must just be a linear combination of *PolyGamma* functions, but again the multivariate case can be much more complicated.

Mathematical Functions

▪ **Page 145 · Mathematical functions.** (See page 1091.) *BesselJ[0, x]* goes like $Sin[x]/\sqrt{x}$ for large *x* while *AiryAi[-x]* goes like $Sin[x^{3/2}]/x^{1/4}$. Other standard mathematical functions that oscillate at large *x* include *JacobiSN* and *MathieuC*. Most hypergeometric-type functions either increase or decrease exponentially for large arguments, though in the directions of Stokes lines in the complex plane they can oscillate sinusoidally. (For *AiryAi[x]* the Stokes lines are in directions *(-1)^({1, 2, 3}/3)*.)

▪ **Lissajous figures.** Plotting multiple sine functions each on different coordinate axes yields so-called Lissajous or Bowditch figures, as illustrated below. If the coefficients inside all the sine functions are rational, then going from *t = 0* to *t = 2 π Apply[LCM, Map[Denominator, list]]* yields a closed curve. Irrational ratios of coefficients lead to curves that never close and eventually fill space uniformly.

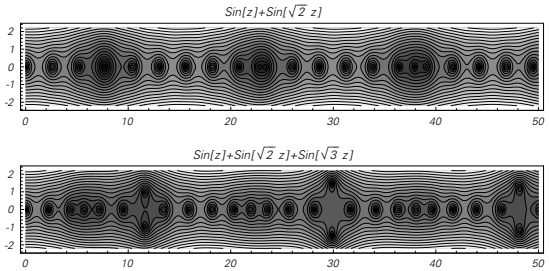


▪ **Page 146 · Two sine functions.** *Sin[a x] + Sin[b x]* can be rewritten as *2 Sin[1/2 (a + b) x] Cos[1/2 (a - b) x]* (using *TrigFactor*), implying that the function has two families of equally spaced zeros: $2\pi n/(a+b)$ and $2\pi (n+1/2)/(b-a)$.

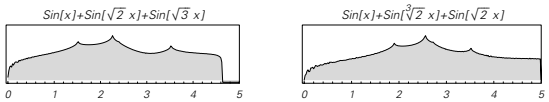
▪ **Differential equations.** The function $Sin[x] + Sin[\sqrt{2}\, x]$ can be obtained as the solution of the differential equation *y''[x] + 2 y[x] - Sin[x] == 0* with the initial conditions *y[0] == 0, y'[0] == 2*.

▪ **Musical chords.** In a so-called equal temperament scale the 12 standard musical notes that make up an octave have a progression of frequencies $2^{n/12}$. Most schemes for musical tuning use rational approximations to these numbers. Until the past century, and since at least the 1300s, diminished fifth or tritone chords that consist of two notes (such as C and G♭) with frequency ratio $\sqrt{2}$ have generally been avoided as sounding discordant. (See also page 1079.)

▪ **Page 146 · Three sine functions.** All zeros of the function *Sin[a x] + Sin[b x]* lie on the real axis. But for *Sin[a x] + Sin[b x] + Sin[c x]*, there are usually zeros off the real axis (even say for *a = 1, b = 3/2, c = 5/3*), as shown in the pictures below.



If *a*, *b* and *c* are rational, *Sin[a x] + Sin[b x] + Sin[c x]* is periodic with period *2 π/GCD[a, b, c]*, and there are a limited number of different spacings between zeros. But in a case like $Sin[x] + Sin[\sqrt{2}\, x] + Sin[\sqrt{3}\, x]$ there is a continuous distribution of spacings between zeros, as shown on a logarithmic scale below. (For $0 < x < 10^6$ there are a total of 448,494 zeros, with maximum spacing $\simeq 4.6$ and minimum spacing $\simeq 0.013$.)

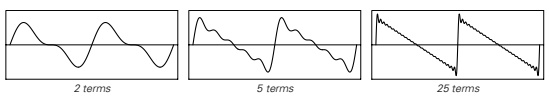


▪ **Page 147 · Substitution systems.** *Cos[a x] - Cos[b x]* has two families of zeros: $2\pi n/(a+b)$ and $2\pi n/(b-a)$. Assuming $b > a > 0$, the number of zeros from the second family which appear between the n^{th} and $n+1^{\text{th}}$ zero from the first family is

(Floor[(n + 1) #] - Floor[n #] &)[(b - a)/(a + b)]

and as discussed on page 903 this sequence can be obtained by applying a sequence of substitution rules. For *Sin[a x] + Sin[b x]* a more complicated sequence of substitution rules yields the analogous sequence in which *-1/2* is inserted in each *Floor*.

▪ **Many sine functions.** Adding many sine functions yields a so-called Fourier series (see page 1074). The pictures below show *Sum[Sin[n x]/n, {n, k}]* for various numbers of terms *k*. Apart from a glitch that gets narrower with increasing *k* (the so-called Gibbs phenomenon), the result has a simple triangular form. Other so-called Fourier series in which the coefficient of *Sin[m x]* is a smooth function of *m* for all integer *m* yield similarly simple results.

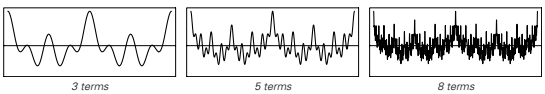


The pictures below show *Sum[Sin[n^2 x]/n^2, {n, k}]*, where in effect all coefficients of *Sin[m x]* other than those where *m* is

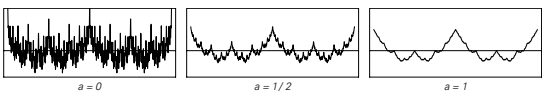
a perfect square are set to zero. The result is a much more complicated curve. Note that for x of the form $p\,\pi/q$, the $k = \infty$ sum is just

```
(π/(2 q))^2 Sum[Sin[n^2 p π/q]/Sin[n π/(2 q)]^2, {n, q - 1}]
```

The pictures below show `Sum[Cos[2^n x], {n, k}]` (as studied by Karl Weierstrass in 1872). The curves obtained in this case show a definite nested structure, in which the value at a point x is essentially determined directly from the base 2 digit sequence of x. (See also page 1080.)



The curves below are approximations to `Sum[Cos[2^n x]/2^(a n), {n, ∞}]`. They can be thought of as having dimensions $2 - a$ and smoothed power spectra $\omega^{-(1+2a)}$.



▪ **FM synthesis.** More complicated curves can be obtained for example using FM synthesis, as discussed on page 1079.

▪ **Page 148 · Zeta function.** For real s the Riemann zeta function `Zeta[s]` is given by `Sum[1/n^s, {n, ∞}]` or `Product[1/(1 - Prime[n]^s), {n, ∞}]`. The zeta function as analytically continued for complex s was studied by Bernhard Riemann in 1859, who showed that `PrimePi[n]` could be approximated (see page 909) up to order $\sqrt{n}$ by `LogIntegral[n] - Sum[LogIntegral[n^r[i]], {i, -∞, ∞}]`, where the `r[i]` are the complex zeros of `Zeta[s]`. The Riemann Hypothesis then states that all `r[i]` satisfy `Re[r[i]] == 1/2`, which implies a certain randomness in the distribution of prime numbers, and a bound of order $\sqrt{n}\,\mathrm{Log}[n]$ on `PrimePi[n] - LogIntegral[n]`. The Riemann Hypothesis is also equivalent to the statement that a bound of order $\sqrt{n}\,\mathrm{Log}[n]^2$ exists on `Abs[Log[Apply[LCM, Range[n]]] - n]`.

The picture in the main text shows `RiemannSiegelZ[t]`, defined as `Zeta[1/2 + i t] Exp[i RiemannSiegelTheta[t]]`, where

```
RiemannSiegelTheta[t_] =
  Arg[Gamma[1/4 + i t/2]] - 1/2 t Log[π]
```

The first term in an approximation to `RiemannSiegelZ[t]` is `2 Cos[RiemannSiegelTheta[t]]`; to get results to a given precision requires summing a number of terms that increases like $\sqrt{t}$, making routine computation possible up to $t \sim 10^{10}$.

It is known that:

- The average spacing between zeros decreases like `1/Log[t]`.
- The amplitude of wiggles grows with t, but more slowly than $t^{0.16}$.
- At least the first 10 billion zeros have `Re[s] == 1/2`.

The statistical distribution of zeros was studied by Andrew Odlyzko and others starting in the late 1970s (following ideas of David Hilbert and George Pólya in the early 1900s), and it was found that to a good approximation, the spacings between zeros are distributed like the spacings between eigenvalues of random unitary matrices (see page 977).

In 1972 Sergei Voronin showed that `Zeta[z + (3/4 + i t)]` has a certain universality in that there always in principle exists some t (presumably in practice usually astronomically large) for which it can reproduce to any specified precision over say the region `Abs[z] < 1/4` any analytic function without zeros.

Iterated Maps and the Chaos Phenomenon

▪ **History of iterated maps.** Newton's method from the late 1600s for finding roots of polynomials (already used in specific cases in antiquity) can be thought of as a smooth iterated map (see page 920) in which a rational function is repeatedly applied (see page 1101). Questions of convergence led in the late 1800s and early 1900s to interest in iteration theory, particularly for rational functions in the complex plane (see page 933). There were occasional comments about complicated behavior (notably by Arthur Cayley in 1879) but no real investigation seems to have been made. In the 1890s Henri Poincaré studied so-called return maps giving for example positions of objects on successive orbits. Starting in the 1930s iterated maps were sometimes considered as possible models in fields like population biology and business cycle theory—usually arising as discrete annualized versions of continuous equations like the Verhulst logistic differential equation from the mid-1800s. In most cases the most that was noted was simple oscillatory behavior, although for example in 1954 William Ricker iterated empirical reproduction curves for fish, and saw more complex behavior—though made little comment on it. In the 1950s Paul Stein and Stanislaw Ulam did an extensive computer study of various iterated maps of nonlinear functions. They concentrated on questions of convergence, but nevertheless noted complicated behavior. (Already in the

late 1940s John von Neumann had suggested using $x \to 4x(1-x)$ as a random number generator, commenting on its extraction of initial condition digits, as mentioned on page 921.) Some detailed analytical studies of logistic maps of the form $x \to ax(1-x)$ were done in the late 1950s and early 1960s—and in the mid-1970s iterated maps became popular, with much analysis and computer experimentation on them being done. But typically studies have concentrated on repetition, nesting and sensitive dependence on initial conditions—not on more general issues of complexity.

In connection with his study of continued fractions Carl Friedrich Gauss noted in 1799 complexity in the behavior of the iterated map $x \to$ *FractionalPart[1/x]*. Beginning in the late 1800s there was number theoretical investigation of the sequence *FractionalPart[a^n x]* associated with the map $x \to$ *FractionalPart[a x]* (see page 903), notably by G. H. Hardy and John Littlewood in 1914. Various features of randomness such as uniform distribution were established, and connections to smooth iterated maps emerged after the development of symbolic dynamics in the late 1930s.

■ **History of chaos theory.** See page 971.

■ **Page 150 · Exact iterates.** For any integer a the n^{th} iterate of $x \to$ *FractionalPart[a x]* can be written as *FractionalPart[a^n x]*, or equivalently *1/2 - ArcTan[Cot[a^n π x]]/π*. In the specific case $a = 2$ the iterates of *If[x < 1/2, a x, a(1-x)]* have the form *ArcCos[Cos[2^n π x]]/π*. (See pages 903 and 1098.)

■ **Page 151 · Problems with computer experiments.** The defining characteristic of a system that exhibits chaos is that on successive steps the system samples digits which lie further and further to the right in its initial condition. But in a practical computer, only a limited number of digits can ever be stored. In *Mathematica*, one can choose how many digits to store (and in the pictures shown in the main text, enough digits were used to avoid the problems discussed in this note). But a low-level language such as FORTRAN, C or Java always stores a fixed number of digits, typically around 53, in its standard double-precision floating-point representation of numbers.

So what happens when a system one is simulating tries to sample digits in its initial conditions beyond the ones that are stored? The answer depends on the way that arithmetic is handled in the computer system one uses.

When doing high-precision arithmetic, *Mathematica* follows the principle that it should only ever give digits that are known to be correct on the basis of the input that was provided. This means that in simulating chaotic systems, the numbers produced will typically have progressively fewer digits: later digits cannot be known to be correct without more precise knowledge of this initial condition. (An example is *NestList[Mod[2#, 1] &, N[π/4, 40], 200]*; *Map[Precision, list]* gives the number of significant digits of each element in the list.)

But most current languages and hardware systems follow a rather different approach. (For low-precision machine arithmetic, *Mathematica* is also forced to follow this approach.) What they do is to give a fixed number of digits as the result of every computation, whether or not all those digits are known to be correct. It is then the task of numerical analysis to establish that in a particular computation, the final results obtained are not unduly affected by digits that are not known to be correct. And in practice, for many kinds of computations, this is to a large extent the case. But whenever chaos is involved, it is inevitably not.

As an example, consider the iterated map $x \to$ *Mod[2x, 1]* discussed in the main text. At each step, this map shifts all the base 2 digits in x one position to the left. But if the computer gives a fixed number of digits at each step, then additional digits must be filled in on the right. On most computers, these additional digits are always 0. And so after some number of steps, all the digits in x are 0, and thus the value of x is simply 0.

But it turns out that a typical pocket calculator gives a different result. For pocket calculators effectively represent numbers in base 10 (actually so-called binary-coded decimal) not base 2, and fill in unknown digits with 0 in base 10. (Base 10 is used so that multiplying for example 1/3 by 3 gives exactly 1 rather than the more confusing result 0.9999... obtained with base 2.)

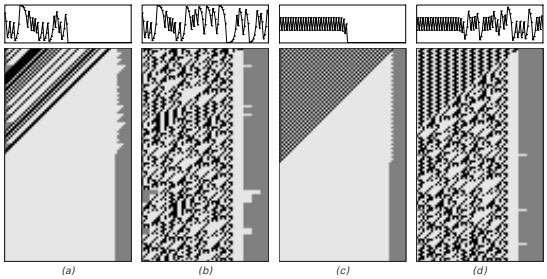
Pictures (a) and (c) below show simulations of the shift map on a typical computer, while pictures (b) and (d) show corresponding simulations on a pocket calculator. (Starting with initial condition x the digit sequence at step n is essentially

IntegerDigits[Mod[2^n Floor[2^{53} x], 2^{53}], 2, 53]

on the computer, and

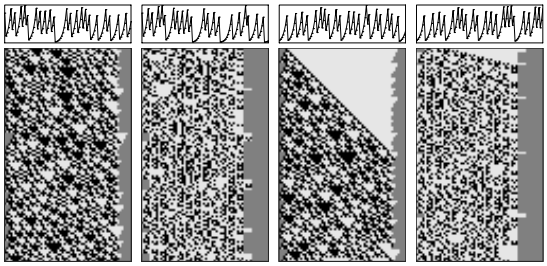
Flatten[IntegerDigits[IntegerDigits[
Mod[2^n Floor[10^{12} x], 10^{12}], 10, 12], 2, 4]]

on the calculator. In both cases the limited number of digits implies behavior that ultimately repeats—but only long after the other effects we discuss have occurred.)



For the first several steps, the results as shown at the top of each corresponding picture agree. But as soon as the effect of sampling beyond the digits explicitly stored in the initial condition becomes important, the results are completely different. The computer gives simply 0, but the pocket calculator yields apparently random sequences—which turn out to be analogous to those discussed on page 319.

Other chaotic systems have a similar sensitivity to the details of computer arithmetic. But the simple behavior of the shift map turns out to be rather rare: in most cases—such as the multiplication by 3/2 shown in the pictures below—apparent randomness is produced, even on a typical computer.



It is important to realize however that this randomness has little to do with the details of the initial conditions. Instead, just as in other examples in this book, the randomness arises from an intrinsic process that occurs even with the simple repetitive initial condition shown in pictures (c) and (d) above.

Computer simulations of chaotic systems have been done since the 1950s. And it has often been observed that the sequences generated in these simulations look quite random. But as we now see, such randomness cannot in fact be a consequence of the chaos phenomenon and of sensitive dependence on initial conditions.

Nevertheless, confusingly enough, even though it does not come from sensitive dependence on initial conditions, such randomness is what makes the overall properties of simulations typically follow the idealized mathematical predictions of chaos theory. The point is that the presence of randomness makes the system behave on different steps as if it were evolving from slightly different initial conditions. But statistical averages over different initial conditions typically yield essentially the results one would get by evolution from a single initial condition containing an infinite number of randomly chosen digits.

▪ **Page 152 · Mathematical perspectives.** Mathematicians may be confused by my discussion of complexity in iterated maps. The first point to make is that the issues I am studying are rather different from the ones that are traditionally studied in the mathematics of these systems. The next point is that I have specifically chosen not to make the idealizations about numbers and operations on numbers that are usually made in mathematics.

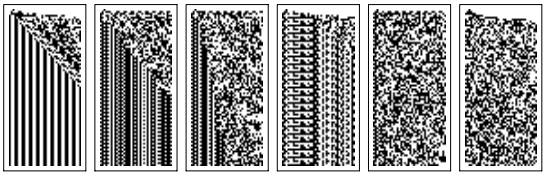
In particular, it is usually assumed that performing some standard mathematical operation, such as taking a square root, cannot have a significant effect on the system one is studying. But in trying to track down the origins of complex behavior, the effects of such operations can be significant. Indeed, as we saw on page 141, taking square roots can for example generate seemingly random digit sequences.

Many mathematicians may object that digit sequences are just too fragile an entity to be worth studying. They may argue that it is only robust and invariant concepts that are useful. But robustness with respect to mathematical operations is a different issue from robustness with respect to computational operations. Indeed, we will see later in this book that large classes of digit sequences can be considered equivalent with respect to computational operations, but these classes are quite different ones from those that are considered equivalent with respect to mathematical operations.

▪ **Information content of initial conditions.** Common sense suggests that it is a quite different thing to specify a simple initial condition containing, say, a single black cell on a white background, than to specify an initial condition containing an infinite sequence of randomly chosen cells. But in traditional mathematics no distinction is usually made between these kinds of specifications. And as a result, mathematicians may find it difficult to understand my distinction between randomness generated intrinsically by the evolution of a system and randomness from initial conditions (see page 299). The distinction may seem more obvious if one considers, for example, sequential substitution systems or cyclic tag systems. For such systems cannot meaningfully be given infinite random initial conditions, yet they can still perfectly well generate highly random behavior. (Their initial conditions correspond in a sense to integers rather than real numbers.)

▪ **Smooth iterated maps.** In the main text, all the functions used as mappings consist of linear pieces, usually joined together discontinuously. But the same basic phenomena seen with such mappings also occur when smooth functions are used. A particularly well-studied example (see page 918) is the so-called logistic map $x \to a\,x\,(1-x)$. The base 2 digit

sequences obtained with this map starting from $x = 1/8$ are shown below for various values of a. The quadratic nature of the map typically causes the total number of digits to double at each step. But at least for small a, progressively more digits on the left show purely repetitive behavior. As a increases, the repetition period goes through a series of doublings. The detailed behavior is different for every value of a, but whenever the repetition period is 2^j, it turns out that with any initial condition the leftmost digit always eventually follows a sequence that consists of repetitions of step j in the evolution of the substitution system $\{1 \to \{1, 0\}, 0 \to \{1, 1\}\}$ starting either from $\{0\}$ or $\{1\}$. As a approaches 3.569946, the period doublings get closer and closer together, and eventually a point is reached at which the sequence of leftmost digits is no longer repetitive but instead corresponds to the nested pattern formed after an infinite number of steps in the evolution of the substitution system. (An important result discovered by Mitchell Feigenbaum in 1975 is that this basic setup is universal to all smooth maps whose functions have a single hump.) When a is increased further, there is usually no longer repetitive or nested behavior. And although there are typically some constraints, the behavior obtained tends to depend on the details of the digit sequence of the initial conditions. In the special case $a = 4$, it turns out that replacing x by $\text{Sin}[\pi u]^2$ makes the mapping become just $u \to \text{FractionalPart}[2u]$, revealing simple shift map dependence on the initial digit sequence. (See pages 1090 and 1098.)



■ **Higher-dimensional generalizations.** One can consider so-called Anosov maps such as $\{x, y\} \to \text{Mod}[m . \{x, y\}, 1]$ where m is a matrix such as $\{\{2, 1\}, \{1, 1\}\}$. Any initial condition containing only rational numbers will then yield repetitive behavior, much as in the shift map. But as soon as m itself contains rational numbers, complicated behavior can be obtained even with an initial condition such as $\{1, 1\}$.

■ **Distribution of chaotic behavior.** For iterated maps, unlike for discrete systems such as cellular automata, one can get continuous ranges of rules by varying parameters. With maps based on piecewise linear functions the regions of parameters in which chaotic behavior occurs typically have simple shapes; with maps based, say, on quadratic functions, however, elaborate nested shapes can occur. (See page 934.)

■ **Page 155 · Lyapunov exponents.** The number of new digits that are affected at each step by a small change in initial conditions gives the so-called Lyapunov exponent λ for the evolution. After t steps, the difference in size resulting from the change in initial conditions will be multiplied by approximately $2^{\lambda t}$—at least until this difference is of order 1. (See page 950.)

■ **Chaos in nature.** See page 304.

■ **Bitwise operations.** Cellular automata can be thought of as analogs of iterated maps in which bitwise operations such as *BitXor* are used instead of ordinary arithmetic ones. (See page 906.)

Continuous Cellular Automata

■ **Implementation.** The state of a continuous cellular automaton at a particular step can be represented by a list of numbers, each lying between 0 and 1. This list can then be updated using

```
CCAEvolveStep[f_, list_List] :=
  Map[f, (RotateLeft[list] + list + RotateRight[list])/3]
CCAEvolveList[f_, init_List, t_Integer] :=
  NestList[CCAEvolveStep[f, #] &, init, t]
```

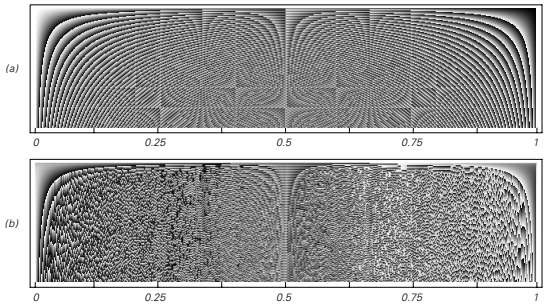
where for the rule on page 157 f is *FractionalPart[3#/2] &* while for the rule on page 158 it is *FractionalPart[# + 1/4] &*.

Note that in the definitions above, the elements of *list* can be either exact rational numbers, or approximate numbers obtained using N. For rough calculations, standard machine-precision numbers may sometimes suffice, but for detailed calculations exact rational numbers are essential. Indeed, the presence of exponentially increasing errors would make the bottom of the picture on page 157 qualitatively wrong if just 64-bit double-precision numbers had been used. On page 160 the effect is much larger, and almost all the pictures would be completely wrong—with the notable exception of the one that shows localized structures.

■ **History.** Continuous cellular automata have been introduced independently several times, under several different names. In all cases the rules have been at least slightly more complicated than the ones I consider here, and behavior starting from simple initial conditions does not appear to have been studied before. Versions of continuous cellular automata arose in the mid-1970s as idealizations of coupled ordinary differential equations for arrays of nonlinear oscillators, and implicitly in finite difference approximations to partial differential equations. They began

to be studied with extensive computer simulations in the early 1980s, probably following my work on ordinary cellular automata. Most often considered, notably by Kunihiko Kaneko and co-workers, were so-called "coupled map lattices" or "lattice dynamical systems" in which an iterated map (typically a logistic map) was applied at each step to a combination of neighboring cell value. A transition from regular class 2 to irregular class 3 behavior, with class 4 behavior involving localized structures in between, was observed, and was studied in detail by Hugues Chaté and Paul Manneville, starting in the late 1980s.

■ **Page 158 · Properties.** At step t the background is *FractionalPart[a t]*. For rational a this always repeats, cycling through *Denominator[a]* possible values (compare page 255). In most patterns generated from initial conditions containing say a single black cell most cells whose values are not forced to be the same end up being at least slightly different—even in cases like $a = 0.375$. Note that in cases like $a = 0.475$ there is some trace of a pattern at every step—but it only becomes obvious when it makes values wrap around from 1 to 0. The pictures below show successive colors of (a) the background (compare page 950) and (b) the center cell for each $a = n/500$ from 0 to 1 for the systems on page 159. (Compare page 243.)

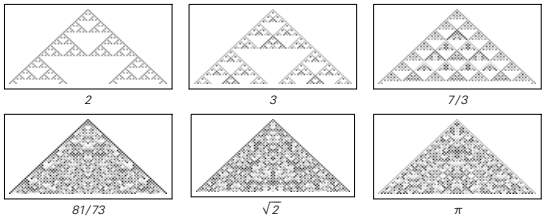


If a is not a rational number the background never repeats, but the main features of patterns obtained seem similar.

■ **Additive rules.** In the case $a = 0$ the systems on page 159 are purely additive. A simpler example is the rule

Mod[RotateLeft[list] + RotateRight[list], 1]

With a single nonzero initial cell with value $1/k$ the pattern produced is just Pascal's triangle modulo k. If k is a rational number only a limited set of values appear, and the pattern has a nested form analogous to those shown on page 870. If k is irrational then equidistribution of *Mod[Binomial[t, x], k]* implies that all possible values eventually appear; the corresponding patterns seem fairly irregular, as shown below. (Compare pages 953 and 1092.)



■ **Probabilistic cellular automata.** As an alternative to having continuous values at each cell, one can consider ordinary cellular automata with discrete values, but introduce probabilities for, say, two different rules to be applied at each cell. Examples of probabilistic cellular automata are shown on page 591; their behavior is typically quite similar to continuous cellular automata.

Partial Differential Equations

■ **Ordinary differential equations.** It is also possible to set up systems which have a finite number of continuous variables (say *a[t]*, *b[t]*, etc.) that change continuously with time. The rules for such systems correspond to ordinary differential equations. Over the past century, the field of dynamical systems theory has produced many results about such systems. If all equations are of the form *a'[t] == f[a[t], b[t], …]*, etc. then it is known for example that it is necessary to have at least three equations in order to get behavior that is not ultimately fixed or repetitive. (The Lorenz equations are an example.) If the function f depends explicitly on time, then two equations suffice. (The van der Pol equations are an example.)

Just as in iterated maps, a small change in the initial values *a[0]* etc. can often lead to an exponentially increasing difference in later values of *a[t]*, etc. But as in iterated maps, the main part of this process that has been analyzed is simply the excavation of progressively less significant digits in the number *a[0]*.

(Note that numerical simulations of ODEs on computers must approximate continuous time by discrete steps, making the system essentially an iterated map, and often yielding spurious complicated behavior.)

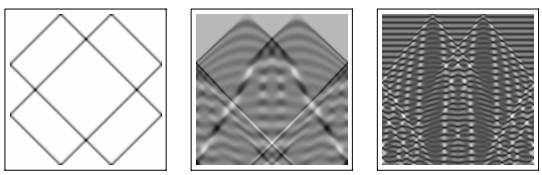
■ **Klein-Gordon equation.** The behavior of the Klein-Gordon equation $\partial_{tt} u[t, x] == \partial_{xx} u[t, x] - u[t, x]$ is visually very similar to that shown for the sine-Gordon equation. For the Klein-Gordon equation, however, there is an exact solution:

$u[t, x] = If[x^2 > t^2, 0, BesselJ[0, Sqrt[t^2 - x^2]]]$

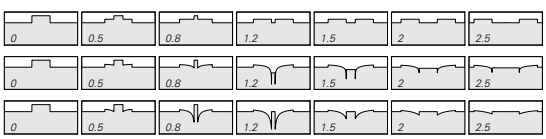
■ **Origins of the equations.** The diffusion equation arises in physics from the evolution of temperature or of gas density.

The wave equation represents the propagation of linear waves, for example along a compressible spring. The sine-Gordon equation represents nonlinear waves obtained for example as the limit of a very large number of pendulums all connected to a spring. The traditional name of the equation is a pun on the Klein-Gordon equation that appears in relativistic quantum mechanics and in describing strings in elastic media. It is notable that unlike with ODEs, essentially all PDEs that have been widely studied come quite directly from physics. My PDE on page 165 is however an exception.

■ **Nonlinearity.** The pictures below show behavior with initial conditions containing two Gaussians (and periodic boundary conditions). The diffusion and wave equations are linear, so that results are linear sums of those with single Gaussians. The sine-Gordon equation is nonlinear, but its solutions satisfy a generalized linear superposition principle. The equation from page 165 shows no such simple superposition principle. Note that even with a linear equation, fairly complicated patterns of behavior can sometimes emerge as a result of boundary conditions.


wave equation sine-Gordon equation my equation

■ **Higher dimensions.** The pictures below show as examples the solution to the wave equation in 1D, 2D and 3D starting from a stationary square pulse.



In each case a 1D slice through the solution is shown, and the solution is multiplied by r^{d-1}. For the wave equation, and for a fair number of other equations, even and odd dimensions behave differently. In 1D and 3D, the value at the origin quickly becomes exactly 0; in 2D it is given by $1 - t/\text{Sqrt}[t^2 - 1]$, which tends to zero only like $-1/(2\,t^2)$ (which means that a sound pulse cannot propagate in a normal way in 2D).

■ **Page 164 · Singular behavior.** An example of an equation that yields inconsistent behavior is the diffusion equation with a negative diffusion constant:

$\partial_t u[t, x] == -\partial_{xx} u[t, x]$

This equation makes any variation in u as a function of x eventually become infinitely rapid.

Many equations used in physics can lead to singularities: the Navier-Stokes equations for fluid flow yield shock waves, while the Einstein equations yield black holes. At a physical level, such singularities usually indicate that processes not captured by the equations have become important. But at a mathematical level one can simply ask whether a particular equation always has solutions which are at least as regular as its initial conditions. Despite much work, however, only a few results along these lines are known.

■ **Existence and uniqueness.** Unlike systems such as cellular automata, PDEs do not have a built-in notion of "evolution" or "time". Instead, as discussed on page 940, a PDE is essentially just a constraint on the values of a function at different times or different positions. In solving a PDE, one is usually interested in determining values that satisfy this constraint inside a particular region, based on information about values on the edges. It is then a fundamental question how much can be specified on the edges in order to obtain a unique solution. If too little is specified, there may be many possible solutions, while if too much is specified there may be no consistent solution at all. For some very simple PDEs, the conditions for unique solutions are known. So-called hyperbolic equations (such as the wave equation, the sine-Gordon equation and my equation) work a little like cellular automata in that in at least one dimension information can propagate only at a limited speed, say c. The result is that in such equations, giving values for $u[t, x]$ at $t = 0$ for $-s < x < s$ will uniquely determine $u[t, x]$ at larger t for $-s + c\,t < x < s - c\,t$. In other PDEs, such as so-called elliptic ones, there is no such limit on the rate of information propagation, and as a result, it is immediately necessary to know values of $u[t, x]$ at all x, and on the boundaries of the region, in order to determine $u[t, x]$ for any $t > 0$.

■ **Page 165 · Field equations.** Any equation of the form

$\partial_{tt} u[t, x] == \partial_{xx} u[t, x] + f[u[t, x]]$

can be thought of as a classical field equation for a scalar field. Defining

$v[u] = -\text{Integrate}[f[u], u]$

the field then has Lagrangian density

$((\partial_t u)^2 - (\partial_x u)^2)/2 - v[u]$

and conserves the Hamiltonian (energy function)

$\text{Integrate}[((\partial_t u)^2 + (\partial_x u)^2)/2 + v[u], \{x, -\infty, \infty\}]$

With the choice for $f[u]$ made here (with $a \geq 0$), $v[u]$ is bounded from below, and as a result it follows that no singularities ever occur in $u[t, x]$.

■ **Equation for the background.** If $u[t, x]$ is independent of x, as it is sufficiently far away from the main pattern, then the partial differential equation on page 165 reduces to the ordinary differential equation

$$u''[t] == (1 - u[t]^2)(1 + a\,u[t])$$

$$u[0] == u'[0] == 0$$

For $a = 0$, the solution to this equation can be written in terms of Jacobi elliptic functions as

$$\sqrt{3}\ \text{JacobiSN}[t/3^{1/4}, 1/2]^2/(1 + \text{JacobiCN}[t/3^{1/4}, 1/2]^2)$$

In general the solution is

$$b\,d\,\text{JacobiSN}[r\,t, s]^2/(b - d\,\text{JacobiCN}[r\,t, s]^2)$$

where

$$r = -\text{Sqrt}[1/8\,a\,c\,(b - d)]$$

$$s = d\,(c - b)/(c\,(d - b))$$

and b, c, d are determined by the equation

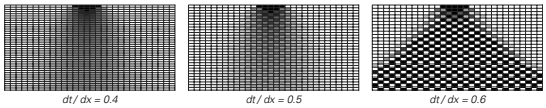
$$(x - b)(x - c)(x - d) == -(12 + 6\,a\,x - 4\,x^2 - 3\,a\,x^3)/(3\,a)$$

In all cases (except when $-8/3 < a < -1/\sqrt{6}$), the solution is periodic and non-singular. For $a = 0$, the period is $2\,3^{1/4}\,\text{EllipticK}[1/2] \simeq 4.88$. For $a = 1$, the period is about 4.01; for $a = 2$, it is about 3.62; while for $a = 4$, it is about 3.18. For $a = 8/3$, the solution can be written without Jacobi elliptic functions, and is given by

$$3\,\text{Sin}[\text{Sqrt}[5/6]\,t]^2/(2 + 3\,\text{Cos}[\text{Sqrt}[5/6]\,t]^2)$$

■ **Numerical analysis.** To find numerical solutions to PDEs on a digital computer one has no choice but to make approximations. In the typical case of the finite difference method one sets up a system with discrete cells in space and time that is much like a continuous cellular automaton, and then hopes that when the cells in this system are made small enough its behavior will be close to that of the continuous PDE.

Several things can go wrong, however. The pictures below show as one example what happens with the diffusion equation when the cells have size dt in time and dx in space. So long as the so-called Courant condition $dt/dx < 1/2$ is satisfied, the results are correct. But when dt/dx is made larger, an instability develops, and the discrete approximation yields completely different results from the continuous PDE.



$dt/dx = 0.4$ $dt/dx = 0.5$ $dt/dx = 0.6$

Many methods beyond finite differences have been invented over the past 30 years for finding numerical solutions to PDEs. All however ultimately involve discretization, and can suffer from difficulties that are similar—though often more insidious—to those for finite differences.

For equations where one can come at least close to having explicit algebraic formulas for solutions, it has often been possible to prove that a certain discretization procedure will yield correct results. But when the form of the true solution is more complicated, such proofs are typically impossible.

And indeed in practice it is often difficult to tell whether complexity that is seen is actually a consequence of the underlying PDE, or is instead merely a reflection of the discretization procedure. I strongly suspect that many equations, particularly in fluid dynamics, that have been studied over the past few decades exhibit highly complex behavior. But in most publications such behavior is never shown, presumably because the authors are not sure whether the behavior is a genuine consequence of the equations they are studying.

■ **Implementation.** All the numerical solutions shown were found using the *NDSolve* function built into *Mathematica*. In general, finite difference methods, the method of lines and pseudospectral methods can be used. For equations of the form

$$\partial_{tt}\,u[t, x] == \partial_{xx}\,u[t, x] + f[u[t, x]]$$

one can set up a simple finite difference method by taking f in the form of pure function and creating from it a kernel with space step dx and time step dt:

```
PDEKernel[f_, {dx_, dt_}] := Compile[{a, b, c, d},
    Evaluate[(2 b - d) + ((a + c - 2 b)/dx^2 + f[b]) dt^2]]
```

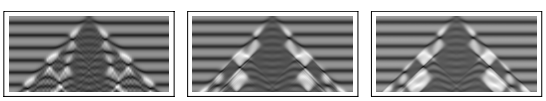
Iteration for n steps is then performed by

```
PDEEvolveList[ker_, {u0_, u1_}, n_] :=
  Map[First, NestList[PDEStep[ker, #] &, {u0, u1}, n]]
PDEStep[ker_, {u1_, u2_}] := {u2, Apply[ker, Transpose[
      {RotateLeft[u2], u2, RotateRight[u2], u1}], {1}]}
```

With this approach an approximation to the top example on page 165 can be obtained from

```
PDEEvolveList[PDEKernel[
    (1 - #^2) (1 + #) &, {0.1, 0.05}], Transpose[
    Table[{1, 1} N[Exp[-x^2]], {x, -20, 20, 0.1}]], 400]
```

For both this example and the middle one the results converge rapidly as dx decreases. But for the bottom example, the pictures below show that convergence is not so rapid, and indeed, as is typical in working with PDEs, despite having used large amounts of computer time I do not know whether the details of the picture in the main text are really correct. The energy function (see above) is at least roughly conserved, but it seems quite likely that the "shocks" visible are merely a consequence of the discretization procedure used.

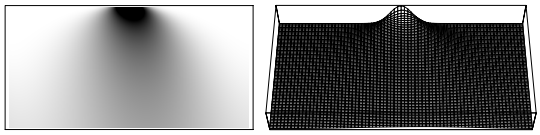


$dx = 0.5$ $dx = 0.2$ $dx = 0.05$

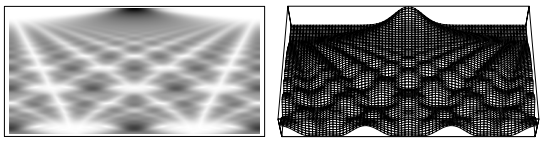
■ **Different powers.** The equations

$\partial_{tt} u[t, x] == \partial_{xx} u[t, x] + (1 - u[t, x]^n)(1 + a\, u[t, x])$

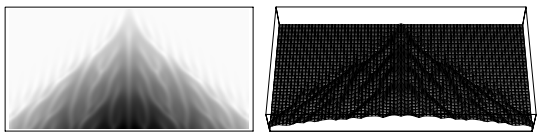
with $n = 4, 6, 8$, etc. appear to show similar behavior to the $n = 2$ equation in the main text.



Burger's equation: $\partial_t u[t, x] = \partial_{xx} u[t, x] - u[t, x]\, \partial_x u[t, x]$



nonlinear Schrödinger equation: $i\, \partial_t u[t, x] = -\partial_{xx} u[t, x] + 4\, \text{Abs}[u[t, x]]^2\, u[t, x]$



Kuramoto-Sivashinsky equation: $\partial_t u[t, x] = -\partial_{xx} u[t, x] - 1/2\, \partial_{xxxx} u[t, x] + (\partial_x u[t, x])^2$

■ **Other PDEs.** The pictures above show three PDEs that have been studied in recent years. All are of the so-called parabolic type, so that, unlike my equation, they have no limit on the rate of information propagation, and thus a solution in any region immediately depends on values on the boundary—which in the pictures below is taken to be periodic. (The deterministic Kardar-Parisi-Zhang equation $\partial_t u[t, x] == a\, \partial_{xx} u[t, x] + 1/2\, b\, (\partial_x u[t, x])^2$ yields behavior like Burger's equation, but symmetrical. Note that *Abs[u]* is plotted in the second picture, while for the last equation a common less symmetrical form replaces the last term by $u[t, x]\, \partial_x u[t, x]$.)

Continuous Versus Discrete Systems

■ **History.** From the late 1600s when calculus was invented it took about two centuries before mathematicians came to terms with the concepts of continuity that it required. And to do so it was necessary to abandon concrete intuition, and instead to rely on abstract mathematical theorems. (See page 1149.) The kind of discrete systems that I consider in this book allow a return to a more concrete form of mathematics, without the necessity for such abstraction.

■ **"Calculus".** It is an irony of language that the word "calculus" now associated with continuous systems comes from the Latin word which means a small pebble of the kind used for doing discrete calculations (same root as "calcium").