

# Chapter 2

## Successive Approximations

In this chapter we continue exploring the mathematical implications of the *S-I-R* model. In the last chapter we calculated future values of *S*, *I*, and *R* by assuming that the rates *S'*, *I'*, and *R'* stayed fixed for a whole day. Since the rates are *not* fixed—they change with *S*, *I*, and *R*—the values of *S*, *I*, and *R* we obtained have to be considered as estimates only. In this chapter we will see how to build a succession of better and better estimates that get us as close as we wish to the true values implied by the model.

This method of **successive approximation** is a basic tool of calculus. It is the one fundamentally new process you will encounter, the ingredient that sets calculus apart from the mathematics you have already studied. With it you will be able to solve a vast array of problems that other methods can't handle.

### 2.1 Making Approximations

In chapter 1 we looked at the specific *S-I-R* model:

$$\begin{aligned}S' &= -.00001 SI, \\I' &= .00001 SI - I/14, \\R' &= I/14,\end{aligned}$$

with initial values at time  $t = 0$ :

$$S = 45400, \quad I = 2100, \quad R = 2500.$$

Rate equations tell us  
where to go next

We originally developed this model as a *description* of the relations among the different components of an epidemic. Almost immediately, though, we began using the rate equations in the model as a *recipe* for predicting what happens over the course of the epidemic: If we know at some time  $t$  the values of  $S(t)$ ,  $I(t)$ , and  $R(t)$ , then the equations tell us how to estimate values of the functions at other times. We used this approach in the last chapter to move backwards and forwards in time, calculating the values of  $S$ ,  $I$ , and  $R$  as we went.

While we got numbers, there were some questions about how accurate these numbers were—that is, how exactly they represented the values implied by the model. In the process we called “there and back again” we used current values of  $S$ ,  $I$ , and  $R$  to find the rates, used these rates to go forward one day, recalculate the rates, and come back to the present—and we got different values from the ones we started with! Resolving this discrepancy will be an important feature of the technique developed in this section.

### The Longest March Begins with a Single Step

So far, in generating numbers from the  $S$ - $I$ - $R$  rate equations, we have assumed that the rates remained constant over an entire day, or longer. Since the rates aren’t constant—they depend on the values of  $S$ ,  $I$ , and  $R$ , which are always changing—the values we calculated for the variables at times other than the given initial time are, at best, estimates. These estimates, while incorrect, are not useless. Let’s see how they behave in the “there and back again” process of chapter 1 as we recalculate the rates more and more frequently, producing a sequence of approximations to the values we are looking for.

#### There and Back Again Again

On page 14 in chapter 1 we used the rate equations to go forward a day and come back again. We started with the initial values

$$S(0) = 45400, \quad I(0) = 2100, \quad R(0) = 2500,$$

calculated the rates, went forward a day to  $t = 1$ , recalculated the rates, and came back a day to  $t = 0$ . We ended up with the estimates

$$S(0) = 45737.1, \quad I(0) = 1820.3, \quad R(0) = 2442.6$$

—which are rather far from the values of  $S(0)$ ,  $I(0)$ , and  $R(0)$  we started with.

A clue to the resolution of this discrepancy appeared in problem 18, page 23. There you were asked to go forward two days and come back again in two different ways, using  $\Delta t = 2$  (a total of 2 steps) in the first case and  $\Delta t = 1$  (a total of 4 steps) in the second. Here are the resulting values calculated for  $S(0)$  in each case and the discrepancy between this value and the original value  $S(0) = 45400$ :

step size	new $S(0)$	discrepancy
$\Delta t = 2$	46717.6	1317.6
$\Delta t = 1$	46021.3	621.3

While the discrepancy is fairly large in either case,  $\Delta t = 1$  clearly does better than  $\Delta t = 2$ . But if smaller is better, why stop at  $\Delta t = 1$ ? What happens if we take even smaller time steps, get the corresponding new values of  $S$ ,  $I$ , and  $R$ , and use these values to recalculate the rates each time?

Smaller steps generate a smaller discrepancy

Recall that the rate  $S'$  (or  $I'$  or  $R'$ ) is simply the multiplier which gives  $\Delta S$ —the (estimated) change in  $S$ —for a given change  $\Delta t$  in  $t$

$$\Delta S = S' \cdot \Delta t.$$

This relation holds for any value of  $\Delta t$ , integer or not. Once we have this value for  $\Delta S$ , we can then calculate

$$\text{new (estimated) } S = \text{current (estimated) } S + \Delta S$$

in the usual way. Note that we have written “(estimated)” throughout to emphasize the fact that if  $S'$  is not constant over the entire time  $\Delta t$ , then the value we get for  $\Delta S$  will typically be only an approximation to the real change in  $S$ .

Let’s try going forward one day and coming back, using different values for  $\Delta t$ . As we reduce  $\Delta t$  the number of calculations will increase. The program SIR we used in the last chapter can still be used to do the tedious calculations. Thus if we decide to use 10 steps of size  $\Delta t = .1$ , we would just change two lines in that program:

```
deltat = .1
FOR k = 1 TO 10
```

If we now run SIR with these modifications we can verify the following sequence of values (The values have been rounded off, and the PRINT statement has been modified to show the new values of the rates at each step as well):

Estimated values of  $S, I,$  and  $R$   
for step sizes  $\Delta t = .1$

$t$	$S(t)$	$I(t)$	$R(t)$	$S'(t)$	$I'(t)$	$R'(t)$
0.0	45 400.0	2100.0	2500.0	-953.4	803.4	150.0
0.1	45 304.7	2180.3	2515.0	-987.8	832.1	155.7
0.2	45 205.9	2263.6	2530.6	-1023.3	861.6	161.7
0.3	45 103.6	2349.7	2546.7	-1059.8	892.0	167.8
$\vdots$						
1.0	44 278.7	3042.9	2678.4	-1347.7	1130.0	217.4

Having arrived at  $t = 1$ , we can now use SIR to turn around and go back to  $t = 0$ . Here's how:

- Change the initial line of the program to  $\mathfrak{t} = 1$  to reflect our new starting time.
- Change the next three lines to use the values we just calculated for  $S(1), I(1),$  and  $R(1)$  as our *starting* values in SIR.
- Change the value of `deltat` to be `-.1` (so each time the program executes the command  $\mathfrak{t} = \mathfrak{t} + \text{deltat}$  it reduces the value of  $\mathfrak{t}$  by `.1`).

The *sign* of `deltat` determines whether we move forward or backward in time

With these changes SIR will yield the desired estimates for  $S(0), I(0),$  and  $R(0)$ , and we get

$$S(0) = 45433.5, \quad I(0) = 2072.3, \quad R(0) = 2494.3.$$

This is clearly a considerable improvement over the values obtained with  $\Delta t = 1$ .

With this promising result, the obvious thing to do is to try even smaller values of  $\Delta t$ , perhaps  $\Delta t = .01$ . We could continue using SIR, making the needed modifications each time. Instead, though, let's rewrite SIR slightly to make it better suited to our current needs. Look at the program SIRVALUE below and compare it with SIR.

**Program: SIRVALUE**

```

tinitial = 0
tfinal = 1
t = tinitial
S = 45400
I = 2100
R = 2500
numberofsteps = 10
deltat = (tfinal - tinitial)/numberofsteps
FOR k = 1 TO numberofsteps
  Sprime = -.00001 * S * I
  Iprime = .00001 * S * I - I / 14
  Rprime = I / 14
  deltaS = Sprime * deltat
  deltaI = Iprime * deltat
  deltaR = Rprime * deltat
  t = t + deltat
  S = S + deltaS
  I = I + deltaI
  R = R + deltaR
NEXT k
PRINT t, S, I, R

```

**Program: SIR**

```

t = 0
S = 45400
I = 2100
R = 2500
deltat = .1
FOR k = 1 TO 10
  Sprime = -.00001 * S * I
  Iprime = .00001 * S * I - I / 14
  Rprime = I / 14
  deltaS = Sprime * deltat
  deltaI = Iprime * deltat
  deltaR = Rprime * deltat
  t = t + deltat
  S = S + deltaS
  I = I + deltaI
  R = R + deltaR
  PRINT t, S, I, R
NEXT k

```

You will see that the major change is to place the `PRINT` statement outside the loop, so only the final values of  $S$ ,  $I$ , and  $R$  get printed. This speeds up the work, since otherwise, with  $\Delta t = .001$ , for instance, we would be asking the computer to print out 1000 lines—about 30 screens of text! Another change is that the value of `deltat` no longer needs to be specified—it is automatically determined by the values of `tinitial`, `tfinal`, and `numberofsteps`.

As written above, `SIR` and `SIRVALUE` both run for 10 steps of size 0.1. By changing the value of `numberofsteps` in the program we can quickly get estimates for  $S(1)$ ,  $I(1)$ , and  $R(1)$  for a wide range of values for  $\Delta t$ . Moreover, once we have these estimates we can use `SIRVALUE` again to go backwards in time to  $t = 0$ , by making changes similar to those we made in `SIR` earlier. First, we need to change the value of `tinitial` to 1 and the value of `tfinal` to 0. Notice that this automatically will make `deltat` a negative quantity, so that each time we run through the loop we step back in time. Second, we need to set the starting values of `S`, `I`, and `R` to the values we just obtained for  $S(1)$ ,  $I(1)$ , and  $R(1)$ . With these changes, `SIRVALUE` will give us the corresponding estimated values for  $S(0)$ ,  $I(0)$ , and  $R(0)$ .

With a computer we  
can generate lots of  
data and look for  
patterns

If we use SIRVALUE with  $\Delta t$  ranging from 1 to .00001 (which means letting `numberofsteps` range from 1 to 100,000) we get the table below. This table lists the computed values of  $S(1)$ ,  $I(1)$ , and  $R(1)$  for each  $\Delta t$ , followed by the estimated value of  $S(0)$  obtained by running SIRVALUE backward in time from these new values, and, finally, the discrepancy between this estimated value of  $S(0)$  and the original value  $S(0) = 45400$ .

Estimated values of  $S, I$ , and  $R$  when  $t = 1$ ,  
for step sizes  $\Delta t = 10^{-N}$ ,  $N = 0, \dots, 5$ ,  
together with the corresponding backwards estimate for  $S(0)$ .

$\Delta t$	$S(1)$	$I(1)$	$R(1)$	new $S(0)$	discrepancy
1.0	44446.6	2903.4	2650.0	45737.0626	337.0626
0.1	44278.6648	3042.9241	2678.4111	45433.4741	33.4741
0.01	44257.8301	3060.1948	2681.9751	45403.3615	3.3615
0.001	44255.6960	3061.9633	2682.3406	45400.3363	.3363
0.0001	44255.4821	3062.1406	2682.3773	45400.0336	.0336
0.00001	44255.4607	3062.1584	2682.3809	45400.0034	.0034

There are several striking features of this table. The first is that if we go forward one day and come back again, we can get back as close as we want to our initial value of  $S(0)$  *provided we recalculate the rates frequently enough*. After 200,000 rounds of calculations ( $\Delta t = .00001$ ) we ended up only .0034 away from our starting value. In fact, there is a clear pattern to the values of the errors as we decrease the step size. In the exercises it is left for you to explore this pattern and show that similar results hold for  $I$  and for  $R$ .

A second feature is that as we read down the column under  $S(1)$ , we find each digit **stabilizes**—that is, after changing for a while, it eventually becomes fixed at a particular value. The initial digits 44 are the first to stabilize, and that happens by the time  $\Delta t = 0.1$ . Then the third digit 2 stabilizes, when  $\Delta t = 0.01$ . Roughly speaking, one more digit stabilizes at each successive level. The table is revealing to us, digit by digit, the true value of  $S(1)$ . By the fifth stage we learn that the integer part of  $S(1)$  is 44255. By the sixth stage we can say that the true value of  $S(1)$  is 44255.4 . . .

When we write  $S(1) = 44255.4 . . .$  we are expressing  $S(1)$  to **one decimal place accuracy**. This says, first, that the decimal expansion of  $S(1)$  begins with exactly the six digits shown and, second, that there are further digits after the 4 (represented by the three dots “. . .”). In this case, we can identify

Smaller steps generate a discrepancy which can be made as small as we like

Approximations lead to exact values

further digits simply by continuing the table. Since our step sizes have the form  $\Delta t = 10^{-N}$ , we just need to increase  $N$ . For example, to express  $S(1)$  accurately to six decimal places, we need to stabilize the first eleven digits in our estimates of  $S(1)$ . The table suggests that  $\Delta t$  should probably be about  $10^{-10}$ —i.e.,  $N = 10$ .

The true value of  $S(1)$  emerges through a process that generates a sequence of successive approximations. We say  $S(1) = 44255.4\dots$  is the **limit** of this sequence as  $\Delta t$  is made smaller and smaller or, equivalently, as  $N$  is made larger and larger. We also say that the sequence of successive approximations **converges** to the limit  $S(1)$ . Here is a mathematical notation that expresses these statements more compactly:

$$\begin{aligned} S(1) &= \lim_{\Delta t \rightarrow 0} \{\text{the estimate of } S(1)\} \quad \text{or, equivalently,} \\ &= \lim_{N \rightarrow \infty} \{\text{the estimate of } S(1) \text{ with } \Delta t = 10^{-N}\}. \end{aligned}$$

The symbol  $\infty$  stands for “infinity,” and the expression  $N \rightarrow \infty$  is often pronounced “as  $N$  goes to infinity.” However, it is often more instructive to say “as  $N$  gets larger and larger, without bound.”

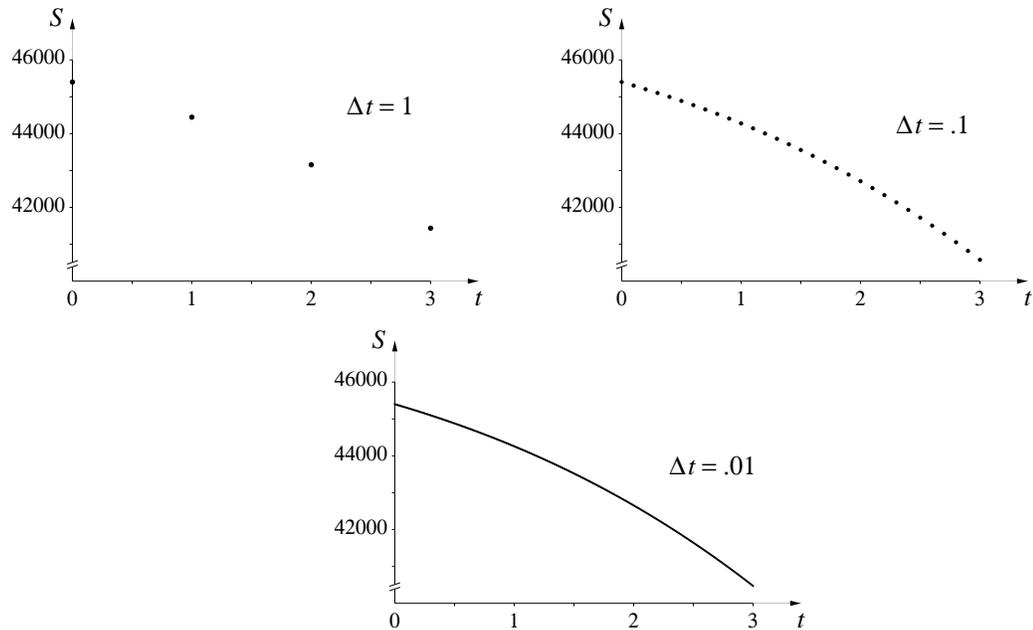
You should check that similar patterns are occurring in the  $I(1)$  and  $R(1)$  columns as well.

The limit concept lies at the heart of calculus. Later on we'll give a precise definition, but you should first see limits at work in a number of contexts and begin to develop some intuitions about what they are. This approach mirrors the historical development of calculus—mathematicians freely used limits for well over a century before a careful, rigorous definition was developed.

## One Picture Is Worth a Hundred Tables

As we noted, the program SIRVALUE prints out only the final values of  $S$ ,  $I$ , and  $R$  because it would typically take too much space to print out the intermediate values. However, if instead of printing these values we plot them graphically, we can convey all this intermediate information in a compact and comprehensible form.

Suppose, for instance, that we wanted to record the calculations leading up to  $S(3)$  by plotting all the points. The graphs below plot all the pairs of values  $(t, S)$  that are calculated along the way for the cases  $\Delta t = 1$  (4 points),  $\Delta t = .1$  (31 points), and  $\Delta t = .01$  (301 points).



By the time we get to steps of size .01, the resulting graph begins to look like a continuous curve. This suggests that instead of simply plotting the points we might want to draw lines connecting the points as they're calculated.

We can easily modify `SIRVALUE` to do this—the only changes will be to replace the `PRINT` command with a command to draw a line and to move this command inside the loop (so that it is executed every time new values are computed). We will also need to add a line or two at the beginning to tell the computer to set up the screen to plot points. This usually involves opening a **window**—i.e., specifying the horizontal and vertical ranges the screen should depict. Since programming languages vary slightly in the way this is done, we use italicized text “*Set up GRAPHICS*” to make clear that this statement is **not** part of the program—you will have to express this in the form your programming language specifies. Similarly, the command

*Plot the line from (t, S) to (t + deltat, S + deltaS)*

will have to be stated in the correct format for your language. The computational core of `SIRVALUE` is unchanged. Here is what the new program looks like if we want to use  $\Delta t = .1$  and connect the points with straight lines:

**Program: SIRPLOT***Set up GRAPHICS*

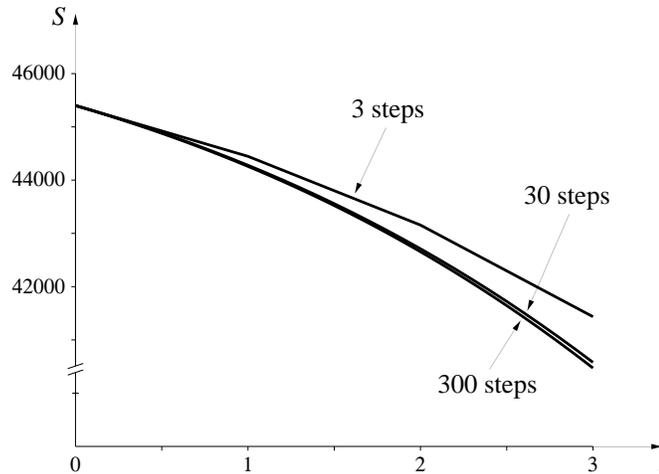
```

tinitial = 0
tfinal = 3
t = tinitial
S = 45400
I = 2100
R = 2500
numberofsteps = 30
deltat = (tfinal - tinitial)/numberofsteps
FOR k = 1 TO numberofsteps
    Sprime = -.00001 * S * I
    Iprime = .00001 * S * I - I / 14
    Rprime = I / 14
    deltaS = Sprime * deltat
    deltaI = Iprime * deltat
    deltaR = Rprime * deltat
    Plot the line from (t, S) to (t + deltat, S + deltaS)
    t = t + deltat
    S = S + deltaS
    I = I + deltaI
    R = R + deltaR
NEXT k

```

If we had wanted just to plot the points, we could have used a command of the form *Plot the point* ( $t$ ,  $S$ ) in place of the command to plot the line, moving this command down two lines so it came after we had computed the new values of  $t$  and  $S$ . We would also need to place that command before the loop so that the initial point corresponding to  $t = 0$  gets plotted.

When we “connect the dots” like this we emphasize graphically the underlying assumption we have been making in all our estimates: that the function  $S(t)$  is linear (i.e., it is changing at a constant rate) over each interval  $\Delta t$ . Let’s see what the graphs look like when we do this for the three values of  $\Delta t$  we used above. To compare the results more readily we’ll plot the graphs on the same set of axes. (We will look at a program for doing this in the next section.) We get the following picture:



Graphs made up of line segments look like smooth curves if the segments are short enough

The graphs become indistinguishable from each other and increasingly look like smooth curves as the number of segments increases. If we plotted the 3000-step graph as well, it would be indistinguishable from the 300-step graph at this scale. If we now shift our focus from the end value  $S(3)$  and look at all the intermediate values as well, we find that each graph gives an approximate value for  $S(t)$  for *every* value of  $t$  between 0 and 3. We are seeing the entire function  $S(t)$  over this interval.

Just as we wrote

$$S(3) = \lim_{N \rightarrow \infty} \{\text{the estimate of } S(3) \text{ with } \Delta t = 10^{-N}\}.$$

We can also write

$$\text{graph of } S(t) = \lim_{N \rightarrow \infty} \{\text{line-segment approximations with } \Delta t = 10^{-N}\}.$$

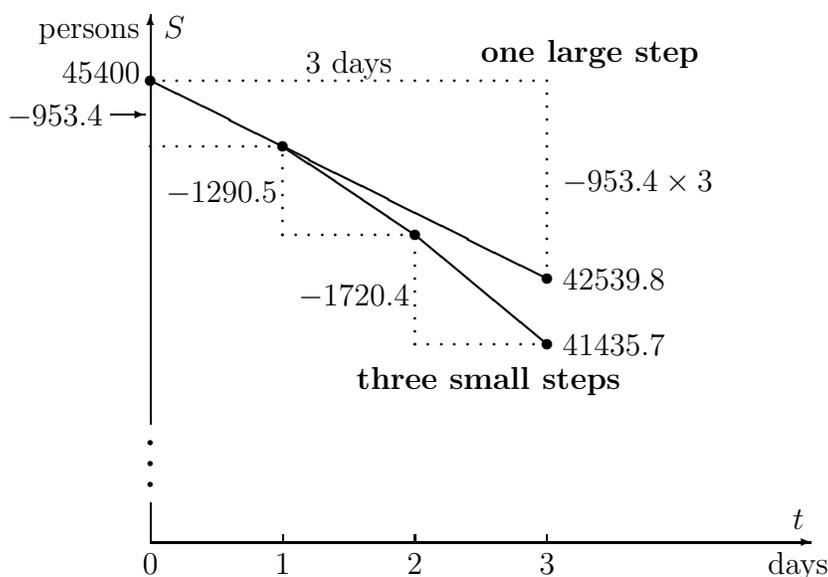
The way we see the graph of  $S(t)$  emerging from successive approximations is our first example of a fundamental result. It has wide-ranging implications which will occupy much of our attention for the rest of the course.

## Piecewise Linear Functions

Let's examine the implications of this approach more closely by considering the “one-step” ( $\Delta t = 3$ ) approximation to  $S(t)$  and the “three-step” ( $\Delta t = 1$ ) approximation over the time interval  $0 \leq t \leq 3$ . In the first case we are making the simplifying assumption that  $S$  decreases at the rate  $S' = -953.4$  persons per day for the entire three days. In the second case we use three shorter steps of length  $\Delta t = 1$ , with the slopes of the corresponding segments given by the table on page 13 in Chapter 1, summarized below (note that since  $\Delta t = 1$  day we have that the magnitude of  $\Delta S = S' \cdot \Delta t$  is the same as the magnitude of  $S'$ ):

$t$	$S$	$S'$
0	45 400.0	-953.4
1	44 446.6	-1 290.5
2	43 156.1	-1 720.4
3	41 435.7	

Here are the corresponding graphs we get:



Two approximations to  $S$  during the first three days

**The “one-step” estimate.** Assuming that  $S$  decreases at the rate  $S' = -953.4$  persons per day for the entire three days is equivalent to assuming that  $S$  follows the upper graph—a straight line with slope  $-953.4$

persons/day. In other words, the one-step approach approximates  $S$  by a **linear function** of  $t$ . If we use the notation  $S_1(t)$  to denote this (one-step) linear approximation we have

$$\text{One-step estimate: } S(t) \approx S_1(t) = 45400 - 953.4t.$$

Because the one-step estimate is actually a function we can find the value of  $S_1(t)$  for *all*  $t$  in the interval  $0 \leq t \leq 3$ , not just  $t = 3$ , and thereby get corresponding estimates for  $S(t)$  as well. For example,

$$\begin{aligned} S_1(2) &= 45400 - 953.4 \times 2 = 43493.2 \\ S_1(1.7) &= 45400 - 953.4 \times 1.7 = 43779.22. \end{aligned}$$

**The “three-step” estimate.** With three smaller steps of size  $\Delta t = 1$ , we get a function whose graph is composed of three line segments, each starting at the  $(t, S)$  point at the beginning of each day and with a slope equal to the corresponding rate of new infections at the beginning of the day. Let’s call this function  $S_3(t)$ . The three-step estimate  $S_3(t)$  is hence not a linear function, strictly speaking. However, since its graph is made up of several straight pieces, it is called a **piecewise linear function**. Recalling that the equation of a line through the point  $(x_0, y_0)$  with slope  $m$  can be written in the form  $y = m(x - x_0) + y_0$ , we can use the values for  $S$  (which correspond to the  $y$ -values) and  $S'$  (which give us the slopes of the segments) at times  $t = 0$ ,  $t = 1$ , and  $t = 2$  calculated above to get an explicit formula for  $S_3(t)$ :

$$S_3(t) = \begin{cases} y = -953.4(t - 0) + 45400 & \text{if } 0 \leq t \leq 1 \\ y = -1290.5(t - 1) + 44446.6 & \text{if } 1 \leq t \leq 2 \\ y = -1720.4(t - 2) + 43156.1 & \text{if } 2 \leq t \leq 3 \end{cases}$$

Note that we have used  $\leq$  in the defining formulas since at the values  $t = 1$  and  $t = 2$  it doesn’t matter which equation we use. This is equivalent to saying that the straight line segments are connected to each other. Since the slopes of the three segments of  $S_3(t)$  are progressively more negative, the piecewise linear graph gets progressively steeper as  $t$  increases. This explains why the value  $S_3(3)$  is *lower* than the value of  $S_1(3)$ . While it is rare that we would actually need to write down an explicit formula like this for the piecewise-linear approximation—it is easier, and usually more informative, just to define  $S_3(t)$  by its graph—it is nevertheless important to

A single function  
may not be specified  
by a single equation

realize that there really is an approximating function defined for all values of  $t$  in the interval  $[0, 3]$ , not just at the finite set of  $t$  values where we make the recalculations.

By the time we are dealing with the 300-step function  $S_{300}(t)$  we can't even tell by its graph that it is piecewise linear unless we zoom in very close. In principle, though, we could still write down a simple linear formula for each of its segments (see the exercises).

**An appraisal.** The graph of  $S_1(t)$  gives us a rough idea of what is happening to the true function  $S(t)$  during the first three days. It starts off at the same rate as  $S'$ , but subsequently the rates move apart. The value of  $S'_1$  never changes, while  $S'$  changes with the (ever-changing) values of  $S$  and  $I$ .

The graph of  $S_3(t)$  is a distinct improvement because it changes its direction twice, modifying its slope at the beginning of each day to come back into agreement with the rate equation. But since the three-step graph is still piecewise linear, it continues to suffer from the same shortcoming as the one-step: once we restrict our attention to a single straight segment (for example, where  $1 \leq t \leq 2$ ), then the three-step graph *also* has a constant slope, while  $S'$  is always changing. Nevertheless,  $S_3(t)$  does satisfy the rate equation in our original model three times—at the beginning of each segment—and isn't too far off at other times. When we get to  $S_{300}(t)$  we have a function which satisfies the rate equation at 300 times and is very close in between.

Each of these graphs gives us an idea of the behavior of the true function  $S(t)$  during the time interval  $0 \leq t \leq 3$ . None is strictly correct, but none is hopelessly wrong, either. All are **approximations** to the truth. Moreover,  $S_3(t)$  is a **better** approximation than  $S_1(t)$ —because it reflects at least some of the variability in  $S'$ —and  $S_{300}(t)$  is better still. Thus, even before we have a clear picture of the shape of the true function  $S(t)$ , we would expect it to be closer to  $S_{300}(t)$  than to  $S_3(t)$ . As we saw above, when we take piecewise linear approximations with smaller and smaller step sizes, it is reasonable to think that they will approach the true function  $S$  in the limit. Expressing this in the notation we have used before,

the function  $S(t) = \lim_{N \rightarrow \infty} \{\text{the chain of linear functions with } \Delta t = 10^{-N}\}.$

## Approximate versus Exact

You may find it unsettling that our efforts give us only a sequence of approximations to  $S(3)$ , and not the exact value, or only a sequence of piecewise-linear approximations to  $S(t)$ , not the “real” function itself. In what sense can we say we “know” the number  $S(3)$  or the function  $S(t)$ ? The answer is: in the same sense that we “know” a number like  $\sqrt{2}$  or  $\pi$ . There are two distinct aspects to the way we know a number. On the one hand, we can **characterize** a number precisely and completely:

- $\pi$ : the ratio of the circumference of a circle to its diameter;
- $\sqrt{2}$ : the positive number whose square is 2;

On the other hand, when we try to **construct** the decimal expansion of a number, we usually get only approximate and incomplete results. For example, when we do calculations by hand we might use the rough estimates  $\sqrt{2} \approx 1.414$  and  $\pi \approx 3.1416$ . With a desk-top computer we might have  $\sqrt{2} \approx 1.414\,213\,562\,373\,095$  and  $\pi \approx 3.141\,592\,653\,589\,793$ , but these are still approximations, and we are really saying

$$\begin{aligned}\sqrt{2} &= 1.414\,213\,562\,373\,09\dots \\ \pi &= 3.141\,592\,653\,589\,79\dots\end{aligned}$$

The *complete* decimal expansions for  $\sqrt{2}$  and  $\pi$  are unknown! The exact values exist as limits of approximations that involve successively longer strings of digits, but we never see the limits—only approximations. In the final section of this chapter we will see ways of generating these approximations for  $\sqrt{2}$  and for  $\pi$ .

What we say about  $\pi$  and  $\sqrt{2}$  is true for  $S(3)$  in exactly the same way. We can *characterize* it quite precisely, and we can *construct* approximations to its numerical value to any desired degree of accuracy. Here, for example, is a characterization of  $S(3)$ :

The *S-I-R* problem for which  $a = .00001$  and  $b = 1/14$  and for which  $S = 45400$ ,  $I = 2100$ ,  $R = 2500$  when  $t = 0$  determines three functions  $S(t)$ ,  $I(t)$ , and  $R(t)$ . The number  $S(3)$  is the value that the function  $S(t)$  has when  $t = 3$ .

You should try to extend this argument to describe the sense in which we “know” the function  $S(t)$  by knowing its piecewise-linear approximations. Try to convince yourself that this is operationally no different from the way

What does it mean to “know” a number like  $\pi$ ?

we “know” functions like  $f(x) = \sqrt{x}$ . In each instance we can **characterize** the function completely, but we can only **construct** an approximation to most values of the function or to its graph.

All this discussion of approximations may strike you as an unfortunate departure from the accuracy and precision you may have been led to expect in mathematics up until now. In fact, it is precisely this ability to make quick and accurate approximations to problems that is one of the most powerful features of mathematics. This is what goes on every time you use your calculator to evaluate  $\log 3$  or  $\sin 37$ . Your calculator doesn’t really know what these numbers are—but it does know how to approximate them quickly to 12 decimal places. Similar kinds of approximations are also at the heart of how bridges are built and spaceships are sent to the moon.

Being able to approximate a number to 12 decimal places is usually as good as knowing its value precisely

**A Caution:** The fact that computers and calculators are really only dealing with approximations when we think they are being exact occasionally leads to problems, the most common of which involves **roundoff errors**. You can probably generate a relatively harmless manifestation of this on your computer with the SIRVALUE program. Modify the PRINT line so it prints out the final value of  $t$  to 10 or 12 digits, and try running it with a high value for `numberofsteps`, say 1 million or 10 million. You would expect the final value of  $t$  to be exactly 1 in every case, since you are adding `deltat = 1/numberofsteps` to itself `numberofsteps` times. The catch is that the computer doesn’t store the exact value `1/numberofsteps` unless `numberofsteps` is a power of 2. In all other cases it will only be using an approximation, and if you add up enough quantities that are slightly off, their cumulative error will begin to show. We will encounter a somewhat less benign manifestation of roundoff error in the next chapter.

However ...

## Exercises

### There and back again

1. a) Look at the table on page 66. What is your best guess of the *exact* value of  $I(1)$ ? (Use the “...” notation introduced on page 66.)  
b) What is the exact value of  $R(1)$ ?
2. We noted that the discrepancy (the difference between the new estimate for  $S(0)$  and the original value) seemed to decrease as  $\Delta t$  decreased.
  - a) What is your best estimate (using only the information in the table) for the value of  $\Delta t$  needed to produce a discrepancy of .001 ?

- b) More generally, express as precisely as you can the apparent relation between the size of the discrepancy and the size of  $\Delta t$ .
3. a) Suppose you wanted to try going three days forward and then coming back, using  $\Delta t = .01$ . What changes would you have to make in SIRVALUE to do this?
- b) Make a table similar to the one on page 66 for going three days forward and coming back for  $\Delta t = 1, .1, .01$ , and  $.001$ .
- c) In this new table how does the size of the discrepancy for a given value of  $\Delta t$  compare with the value in the original table?
- d) What value of  $\Delta t$  do you think you would need to determine the integer parts of  $S(3)$  and  $R(3)$  exactly?

#### Piecewise linear functions

4. Using this three-step approximation, what is  $S_3(1.7)$ ? What is  $S_3(2.5)$ ?
5. How would you modify SIRVALUE to get  $S_{3000}(3)$ ? Do it; what do you get?
6. What additional changes would you make to get the values of  $t$ ,  $S$ , and  $S'$  at the beginning of the 193rd segment of  $S_{300}(t)$ ? [HINT: You only need to alter the `FOR k = 1 TO numberofsteps` line (since you don't want to go all the way to the end) and the `PRINT` line. (Note that after running the loop for, say, 20 times, the values of `t` and `S` are the values for the beginning of the 21st segment, while the value of `Sprime` will still be the slope of the 20th segment.)]
7. Suppose we wanted to determine the value of  $S_{300}(2.84135)$ .
- a) In which of the 300 segments of the graph of  $S_{300}(t)$  would we look to find this information?
- b) What are the values of  $t$ ,  $S$ , and  $S'$  at the beginning of this segment?
- c) What is the equation of this segment?
- d) What is  $S_{300}(2.84135)$ ?
8. How would you modify SIRVALUE to calculate estimates for  $S$ ,  $I$ , and  $R$  when  $t = -6$ , using  $\Delta t = .05$ ? Do it; what do you get?

9. We want to use SIRPLOT to look at the graph of  $S(t)$  over the first 20 days, using  $\Delta t = .01$ .
- What changes would we have to make in the program?
  - Sketch the graph you get when you make these changes.
  - If you wanted to plot the graph of  $I(t)$  over this same time interval, what additional modifications to SIRPLOT would be needed? Make them, and sketch the result. When does the infection appear to hit its peak?
  - Modify SIRPLOT to sketch on the same graph all three functions over the first 70 days. Sketch the result.

### The DO-WHILE loop

A difficulty in giving a precise answer to the last question was that we had to get all the values for 20 days, then go back to estimate by eye when the peak occurred. It would be helpful if we could write a program that ran until it reached the point we were looking for, and then stopped. To do this, we need a different kind of loop—a *conditional* loop that keeps looping only while some specified condition is true. A DO-WHILE loop is one useful way to do this. Here's how the modified SIRPLOT program would look:

```

Set up GRAPHICS
tinitial = 0
t = tinitial
S = 45400
I = 2100
R = 2500
Iprime = .00001 * S * I - I / 14
deltat = .01
DO WHILE Iprime > 0
    Sprime = -.00001 * S * I
    Iprime = .00001 * S * I - I / 14
    Rprime = I / 14
    deltaS = Sprime * deltat
    deltaI = Iprime * deltat
    deltaR = Rprime * deltat
    Plot the line from (t, S) to (t + deltat, S + deltaS)
    t = t + deltat
    S = S + deltaS
    I = I + deltaI
    R = R + deltaR
LOOP
PRINT t - deltat

```

The changes we have made are:

- Since we don't know what the final time will be, eliminate the `tfinal = 20` and the `numberofsteps = 2000` commands.
- Since the condition in our loop is keyed to the value of  $I'$ , we have to calculate the initial value of  $I'$  before the loop starts.
- Instead of `deltat = (tfinal - tinitial)/numberofsteps` use the statement `deltat = .01`.
- The key change is to replace the `FOR k = 1 TO numberofsteps` line by the line `DO WHILE Iprime > 0`.
- To denote the end of the loop we replace the `NEXT k` command with the command `LOOP`.
- After the `LOOP` command add the line `PRINT t - deltat`. (The reason we had to back up one step at the end is because due to the way the program was written, the computer takes one final step with a negative value for `Iprime` before it stops.)

The net effect of all this is that the program will continue working as before, calculating values and plotting points  $(t, S)$ , but *only as long as the condition in the DO WHILE statement is true*. The condition we used here was that  $I'$  had to be positive—this is the condition that ensures that values of  $I$  are still getting bigger. While this condition is true, we can always get a larger value for  $I$  by going forward another increment  $\Delta t$ . As soon as the condition is false—i.e., as soon as  $I'$  is negative—the values for  $I$  will be decreasing, which means we have passed the peak and so want to stop.

10. Make these modifications; what value for  $t$  do you get?
11. You could modify the `PRINT t` command to also print out other quantities.
  - a) What is the value of  $I$  at its peak?
  - b) What is the value of  $S$  when  $I$  is at its peak? Does this agree with the threshold value we predicted in chapter 1?
12. Suppose you change the initial value of  $S$  to be 5400 and run the previous program. Now what happens? Why?

13. Suppose we wanted to know how long the epidemic lasts. We could use DO-WHILE and keep stepping forward using, say  $\Delta t = .1$ , so long as  $I \geq 1$ . As soon as  $I$  was less than 1 we would want to stop and see what the value of  $t$  was.

a) What modifications would you make in SIRVALUE to get this information?

b) Run your modified program. What value do you get for  $t$ ?

c) Run the program using  $\Delta t = .01$ . Now what is your estimate for the duration of the epidemic? What can you say about the actual time required for  $I$  to drop below 1?

14. a) If we think the epidemic started with a single individual, we can go backwards in time until  $I$  is no longer greater than 1, and see what the corresponding time is. Do this for  $\Delta t = -1, -.1, \text{ and } -.01$ . What is your best estimate for the time that the infection arrived?

b) How many Recovereds were there at the start of the epidemic? This would be the people who had presumably been infected in a previous epidemic and now had immunity.

## 2.2 The Mathematical Implications— Euler's Method

### Approximate Solutions

In the last section we approximated the function  $S(t)$  by piecewise-linear approximations using steps of size  $\Delta t = 1, .1, \text{ and } .01$ . This process can clearly be extended to produce approximations with an *arbitrary* number of steps. For any given step size  $\Delta t$ , the result is a piecewise linear graph whose segments are  $\Delta t$  days wide. This graph then provides us with an estimate for  $S(t)$  for every value of  $t$  in the interval  $0 \leq t \leq 3$ . We call this process of obtaining a function by constructing a sequence of increasingly better approximations **Euler's method**, after the Swiss mathematician Leonhard Euler (1707–1783). Euler was interested in the general problem of finding the functions determined by a set of rate equations, and in 1768 he proposed this method to approximate them. The method is conceptually simple and can indeed be used to get solutions for an enormous range of rate equations. For this reason we will make it a basic tool.

To begin to get a sense of the general utility of Euler's method, let's use it in a new setting. Here is a simple problem that involves just a single variable  $y$  that depends on  $t$ .

$$\begin{aligned} \text{rate equation: } & y' = .1 y \left(1 - \frac{y}{1000}\right), \\ \text{initial condition: } & y = 100 \text{ when } t = 0. \end{aligned}$$

To solve the problem we must find the function  $y(t)$  determined by this rate equation and initial condition. We'll work without a context, in order to emphasize the purely mathematical nature of Euler's method. However, this rate equation is one member of a family called **logistic equations** frequently used in population models. We will explore this context in the exercises.

Let's now construct the function that approximates  $y(t)$  on the interval  $0 \leq t \leq 75$ , using  $\Delta t = .5$ . We can make the suitable modifications in SIRPLOT or SIRVALUE to get this approximation. Suppose we want to view the approximation graphically. Here's what the modified SIRPLOT would look like:

### Program: modified SIRPLOT

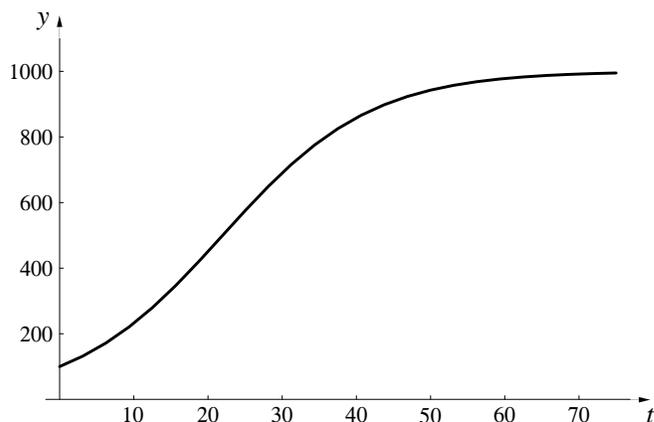
```

Set up GRAPHICS
tinitial = 0
tfinal = 75
t = tinitial
y = 100
numberofsteps = 150
deltat = (tfinal - tinitial)/numberofsteps
FOR k = 1 TO numberofsteps
    yprime = .1 * y * (1 - y / 1000)
    deltay = yprime * deltat
    Plot the line from (t, y) to (t + deltat, y + deltay)
    t = t + deltat
    y = y + deltay
NEXT k

```

As before, words in italics, like "*Plot the line from... to*" need to be translated into the specific formulation required by the computer language you are using.

When we run this program, we get the following graph (axes and scales have been added):



As before, what we see here is only an approximation to the true solution  $y(t)$ . How can we get some idea of how good this approximation is?

## Exact Solutions

For any chosen step size, we can produce an *approximate* solution to a rate equation problem. We will call such an approximation an **Euler approximation**. In the last section we saw that we can improve the accuracy of the approximation by making the steps smaller and using more of them.

For example, we have already found an approximate solution to the problem

$$y' = .1y \left( 1 - \frac{y}{1000} \right); \quad y(0) = 100$$

on the interval  $0 \leq t \leq 75$ , using 150 steps of size  $\Delta t = 1/2$ . Consider a **sequence** of Euler approximations to this problem that are obtained by increasing the number of steps from one stage to the next. To be systematic, let the first approximation have 1 step, the next 2, the next 4, and so on. (The important feature is that the number of steps increases from one approximation to the next, not necessarily that they double—going up by powers of 10 would be just as good. A slight advantage in using powers of 2 is to maximize computer accuracy.) The number of steps thus has the form  $2^{j-1}$ , where  $j = 1, 2, 3, \dots$ . If we use  $y_j(t)$  to denote the approximating



in the graph above, this happens around  $j = 11$ . If we had drawn the graph of  $y_{15}$  or  $y_{20}$ , it would not have been distinguishable from the graph of  $y_{11}$ . It is this entire process of calculating a sequence of successive approximations using increasingly many steps as far as is needed to get the desired level of stabilization that is meant when we talk about Euler's method.

Euler's method is the process of finding solutions through a sequence of successive approximations

The program SEQUENCE shown below plots 14 Euler approximations to  $y(t)$ , increasing the number of steps by a factor of 2 each time. It demonstrates how the graphs of  $y_j(t)$  stabilize to define  $y(t)$  as their limit.

**Program: SEQUENCE**

**A sequence of graphs for  $y' = .1y(1 - y/1000)$ ;  $y(0) = 100$**

*Set up GRAPHICS*

```
FOR j = 1 TO 14
  tinitial = 0
  tfinal = 75
  t = tinitial
  y = 100
  numberofsteps = 2 ^ (j - 1)
  deltat = (tfinal - tinitial) / numberofsteps
  FOR k = 1 TO numberofsteps
    yprime = .1 * y * (1 - y / 1000)
    deltay = yprime * deltat
    Plot the line from (t, y)
      to (t + deltat, y + deltay)
    Color the line with color j
    t = t + deltat
    y = y + deltay
  NEXT k
NEXT j
```

**Program:  
modified SIRPLOT**

Notice that SEQUENCE contains the program SIRPLOT embedded in a loop that executes SIRPLOT 14 times. In this way SEQUENCE plots 14 different graphs. The only new element that has been added to SIRPLOT is “*Color the line with color j*”. When you express this in your programming language it instructs the computer to draw the  $j$ -th graph using color number  $j$  in the computer's “palette.” In the exercises you are asked to use the program SEQUENCE to explore the solutions to a number of rate equation problems.

### Approximate solutions versus exact

By constructing successive approximations to the solution of a rate equation problem, using a sequence of step sizes  $\text{deltat} = \Delta t$  that shrink to 0, we obtain the *exact* solution in the limit.

In practice, though, all we can ever get are particular approximations. However, we can control the level of precision in our approximations by adjusting the step size. If we are dealing with a model of some real process, then this is typically all we need. For example, when it comes to interpreting the *S-I-R* model, we might be satisfied to predict that there will be about 40500 susceptibles remaining in the population after three days. The table on page 66 indicates we would get that level of precision using a step size of about  $\Delta t = 10^{-2}$ . Greater precision than this may be pointless, because the modelling process—which converts reality to mathematics—is itself only an approximation.

The question we asked in the last section—In what sense do we know a number?—applies equally to the functions we obtain using Euler’s method. That is, even if we can characterize a function quite precisely as the solution of a particular rate equation, we may be able to evaluate it only approximately.

### A Caution

We have now seen how to take a set of rate equations and find approximations to the solution of these equations to any degree of accuracy desired. It is important to remember that all these mathematical manipulations are only drawing inferences about the model. We are essentially saying that *if* the original equations capture the internal dynamics of the situation being modelled, *then* here is what we would expect to see. It is still essential at some point to go back to the reality being modeled and check these predictions to see whether our original assumptions were in fact reasonable, or need to be modified. As Alfred North Whitehead has said:

There is no more common error than to assume that, because prolonged and accurate mathematical calculations have been made, the application of the result to some fact of nature is absolutely certain.

## Exercises

### Approximate solutions

1. Modify SIRVALUE and SIRPLOT to analyze the population of Poland (see exercise 25 of chapter 1, page 47). We assume the population  $P(t)$  satisfies the conditions

$$P' = .009 P \quad \text{and} \quad P(0) = 37,500,000,$$

where  $t$  is years since 1985. We want to know  $P$  100 years into the future; you can assume that  $P$  does not exceed 100,000,000.

- a) Estimate the population in 2085.
- b) Sketch the graph that describes this population growth.

### The Logistic Equation

Suppose we were studying a population of rabbits. If we turn 100 rabbits loose in a field and let  $y(t)$  be the number of rabbits at time  $t$  measured in months, we would like to know how this function behaves. The next several exercises are designed to explore the behavior of the rate equation

$$y' = .1y \left( 1 - \frac{y}{1000} \right); \quad y(0) = 100$$

and see why it might be a reasonable model for this system.

2. By modifying SIRVALUE in the way we modified SIRPLOT to get SEQUENCE, obtain a sequence of estimates for  $y(37)$  that allows you to specify the exact value of  $y(37)$  to two decimal places accuracy.
3.
  - a) Referring to the graph of  $y(t)$  obtained in the text on page 82, what can you say about the behavior of  $y$  as  $t$  gets large?
  - b) Suppose we had started with  $y(0) = 1000$ . How would the population have changed over time? Why?
  - c) Suppose we had started with  $y(0) = 1500$ . How would the population have changed over time? Why?
  - d) Suppose we had started with  $y(0) = 0$ . How would the population have changed over time? Why?

- e) The number 1000 in the denominator of the rate equation is called the **carrying capacity** of the system. Can you give a physical interpretation for this number?
4. Obtain graphical solutions for the rate equation for different values of the carrying capacity. What seems to be happening as the carrying capacity is increased? (Don't restrict yourself to  $t = 37$  here.) In this problem and the next you should sketch the different solutions on the same set of axes.
5. Keep everything in the original problem unchanged except for the constant .1 out front. Obtain graphical solutions with the value of this constant = .05, .2, .3, and .6. How does the behavior of the solution change as this constant changes?
6. Returning to the original logistic equation, modify SIRVALUE or DO-WHILE to find the value for  $t$  such that  $y(t) = 900$ .
7. Suppose we wanted to fit a logistic rate equation to a population, starting with  $y(0) = 100$ . Suppose further that we were comfortable with the 1000 in the denominator of the equation, but weren't sure about the .1 out front. If we knew that  $y(20) = 900$ , what should the value for this constant be?

### Using SEQUENCE

8. Each Euler approximation is made up of a certain number of straight line segments. What instruction in the program SEQUENCE determines the number of segments in a particular approximation? The first graph drawn has only a single segment. How many does the fifth have? How many does the fourteenth have?
9. What is the slope of the first graph? What are the slopes of the two parts of the second graph? [You should be able to answer these questions without resorting to a computer.]
10. Modify the line in the program SEQUENCE which determines the number of steps by having it read `numberofsteps = j`, and run the modified program. Again, we are getting a sequence of approximations, with the number of steps increasing each time, but the approximations don't seem to be getting all that close to anything. Explain why this modified program isn't as effective for our purposes as the original.

11. Modify SEQUENCE to produce a sequence of Euler approximations to the function  $y(t)$  that satisfies the conditions

$$y' = .2y(5 - y) \quad \text{and} \quad y(0) = 1.$$

on the interval  $0 \leq t \leq 10$ . [You need to change the final  $t$  value in the program, and you also need to ensure that the graphs will fit on your screen.]

a) What is  $y(10)$ ? [If you add the line `PRINT j, y` just before the line `NEXT j`, a sequence of 14 estimates for  $y(10)$  will appear on the screen with the graphs.]

b) Make a rough sketch of the graph that is the limit of these approximations. The right half of the limit graph has a distinctive feature; what is it?

c) Without doing any calculations, can you estimate the value of  $y(50)$ ? How did you arrive at this value?

d) Change the **initial condition** from  $y(0) = 1$  to  $y(0) = 9$ . Construct the sequence of Euler approximations beginning with `numberofsteps` being 1, and make a rough sketch of the limit graph. What is  $y(10)$  now? Explain why the first several approximations look so strange.

12. Modify SEQUENCE to construct a sequence of Euler approximations for population of Poland (from exercise 1, above). Sketch the limit graph  $P(t)$ , and mark the values of  $P(0)$  and  $P(100)$  at the two ends.

13. Construct a sequence of Euler approximations to the function  $y(t)$  that satisfies the conditions

$$y' = 2t \quad \text{and} \quad y(0) = 0$$

over the interval  $0 \leq t \leq 2$ . Note that this time the rate  $y'$  is given in terms of  $t$ , not  $y$ . Euler's method works equally well. Using your sequence of approximations, estimate  $y(2)$ . How accurate is your estimate?

14. Construct a sequence of Euler approximations to the function  $y(t)$  that satisfies the conditions

$$y' = \frac{4}{1+t^2} \quad \text{and} \quad y(0) = 0$$

over the interval  $0 \leq t \leq 1$ . Estimate  $y(1)$ . How accurate is your estimate? [Note: the exact value of  $y(1)$  is  $\pi$ , which your estimates may have led you to expect. By using special methods we shall develop much later we can prove that  $y(1) = \pi$ .]

## 2.3 Approximate Solutions

Our efforts to find the functions that were determined by the rate equations for the  $S$ - $I$ - $R$  model have brought to light several important issues:

- We often have to deal with a question that does not have a simple, straightforward answer; perhaps we are trying to determine a quantity (like the square root of 2, or  $S(3)$  in the  $S$ - $I$ - $R$  model), to find some function (like  $S(t)$ ), or to understand a process (like an epidemic, or buying and selling in a market). An **approximation** can get us started.
- In many instances, we can make repeated improvements in the approximation. If these **successive approximations** get arbitrarily close to the unknown, and they do it quickly enough, that may answer the question for all practical purposes. In many cases, there is no alternative.
- The information that successive approximations give us is conveyed in the form of a **limit**.
- The method of successive approximations can be used to evaluate many kinds of mathematical objects, including numbers, graphs, and functions.
- **Limit processes** give us a valuable tool to probe difficult questions. They lie at the heart of calculus.

Even the process of building a mathematical model for a physical system can be seen as an instance of successive approximations. We typically start with a simple model (such as the  $S$ - $I$ - $R$  model) and then add more and more features to it (e.g., in the case of the  $S$ - $I$ - $R$  model we might divide the population into different subgroups, have the parameters in the model depend on the season of the year, make immunity of limited duration, etc.). Is it always possible, at least in theory, to get a sequence of approximating mathematical models that approaches reality in the limit?

In the following chapters we will apply the process of successive approximation to many different kinds of problems. For example, in chapter 3 the problem will be to get a better understanding of the notion of a rate of change of one quantity with respect to another. Then, in chapter 4, we will return to the task of solving rate equations using Euler's method. Chapter 6 introduces the integral, defining it through a sequence of successive approximations. As you study each chapter, pause to identify the places where the method of

successive approximations is being used. This can give you insight into the special role that calculus plays within the broader subject of mathematics.

To illustrate the general utility of the method, we end this chapter by returning to the problem raised in section 1 of constructing the values of  $\sqrt{2}$  and  $\pi$  to an arbitrary number of decimal places.

### Calculating $\pi$ —The Length of a Curve

Humans were grappling with the problem of calculating  $\pi$  at least 3000 years ago. In his work *Measurement of the Circle*, Archimedes (287–212 B.C.) used the method of successive approximations to calculate  $\pi = 3.14\dots$ . He did this by starting with a circle of diameter 1, constructing an inscribed and a circumscribed hexagon, and calculating the lengths of their perimeters. The perimeter of the circumscribed hexagon was clearly an overestimate for  $\pi$ , while the perimeter of the inscribed hexagon was an underestimate. He then improved these estimates by going from hexagons to inscribed and circumscribed 12-sided polygons and again calculating the perimeters. He repeated this process of doubling the number of sides until he had inscribed and circumscribed polygons with 96 sides. These left him with his final estimate

$$3.1409\dots = 3\frac{284\frac{1}{4}}{2017\frac{1}{4}} < \pi < 3\frac{667\frac{1}{2}}{4673\frac{1}{2}} = 3.1428\dots$$

In grade school we learned a nice simple formula for the length of a circle, but that was about it. We were never taught formulas for the lengths of other simple curves like elliptic or parabolic arcs, for a very good reason—there are no such formulas. There are various physical approaches we might take. For example, we could get a rough approximation by laying a piece of string along the curve, then picking up the string and measuring it with a ruler. Instead of a physical solution, we can use the essence of Archimedes' insight of approximating a circle by an inscribed “polygon”—what we have earlier called a piecewise linear graph—to determine the length of any curve. The basic idea is reminiscent of the way we made successive approximations to the functions  $S(t)$ ,  $I(t)$ , and  $R(t)$  in the first section of this chapter. Here is how we will approach the problem:

- approximate the curve by a chain of straight line segments;
- measure the lengths of the segments;

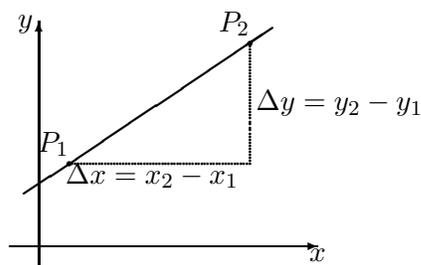
- use the sum of the lengths as an approximation to the true length of the curve.

Repeat this process over and over, each time using a chain that has shorter segments (and therefore more of them) than the last one. The length of the curve emerges as the limit of the sums of the lengths of the successive chains.

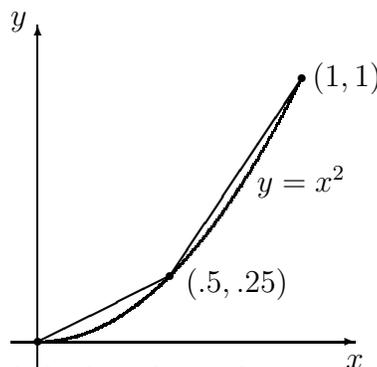
**Distance Formula** If we are given two points  $P_1(x_1, y_1)$  and  $P_2(x_2, y_2)$  in the plane, then the distance between them is just

$$\begin{aligned} d &= \sqrt{\Delta x^2 + \Delta y^2} \\ &= \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \end{aligned}$$

That this follows directly from the Pythagorean theorem can be seen from the picture below:



We'll demonstrate how this process works on a parabola. Specifically, consider the graph of  $y = x^2$  on the interval  $0 \leq x \leq 1$ . At the right we have sketched the graph and our initial approximation. It is a piecewise linear approximation with two segments whose end points have equally spaced  $x$ -coordinates.



We can use the distance formula to find the lengths of the two segments.

$$\begin{aligned} \text{first segment} &: \sqrt{(.5 - 0)^2 + (.25 - 0)^2} = .559016994 \\ \text{second segment} &: \sqrt{(1 - .5)^2 + (1 - .25)^2} = .901387819 \end{aligned}$$

Their total length is the sum

$$.559016994 + .901387819 = 1.460404813.$$

The following program prints out the lengths of the two **segments** and their **total** length.

**Program: LENGTH**  
**Estimating the length of  $y = x^2$  over  $0 \leq x \leq 1$**

```

DEF fnf (x) = x ^ 2
xinitial = 0
xfinal = 1
numberofsteps = 2
deltax = (xfinal - xinitial) / numberofsteps
total = 0
FOR k = 1 TO numberofsteps
  x1 = xinitial + (k - 1) * deltax
  xr = xinitial + k * deltax
  y1 = fnf(x1)
  yr = fnf(xr)
  segment = SQR((xr - x1) ^ 2 + (yr - y1) ^ 2)
  total = total + segment
  PRINT k, segment
NEXT k
PRINT numberofsteps, total

```

## Finding Roots with a Computer

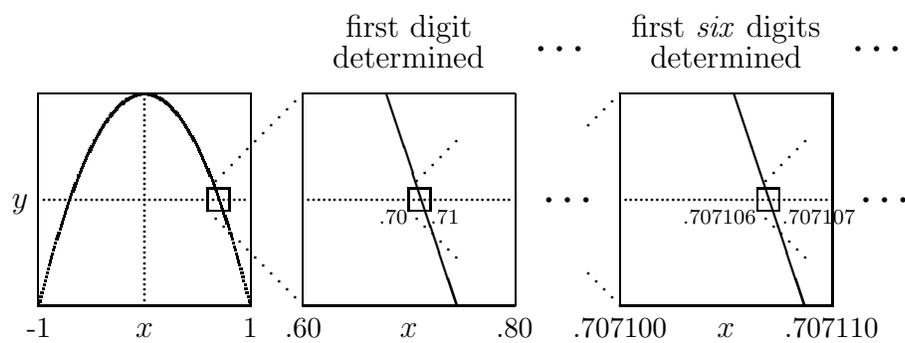
When we casually turn to our calculator and ask it for the value of  $\sqrt{2}$ , what does it really do? Like us, the calculator can only add, subtract, multiply, and divide. Anything else we ask it to do must be reducible to these operations. In particular, the calculator doesn't really "know" the value of  $\sqrt{2}$ . What it does know is how to approximate  $\sqrt{2}$  to, say, 12 significant figures using only elementary arithmetic. In this section we will look at two ways we might do this. Apart from the fact that both approaches use successive approximations, they are remarkably different in flavor. One works graphically, using a computer graphing package, and the other is a numerical algorithm that is about 4000 years old.

Calculators and  
computers really work  
by making  
approximations

### A geometric approach

Exercise 9 on page 40 considered the problem of finding the roots of  $f(x) = 1 - 2x^2$ . A bit of algebra confirms that  $\sqrt{2}/2$  is a root—i.e.,  $f(\sqrt{2}/2) = 0$ . The question is: what is the *numerical value* of  $\sqrt{2}/2$ ?

We'll answer this question by constructing a sequence of approximations that add digits, one at a time, to an estimate for  $\sqrt{2}/2$ . Since the root lies at the point where the graph of  $f$  crosses the  $x$ -axis, we just magnify the graph at this point over and over again, “trapping” the point between  $x$  values that can be made arbitrarily close together.



The graph of  $y = 1 - 2x^2$  under successive magnifications

If we make each stage a ten-fold magnification over the previous one, then, as we zoom in on the next smaller interval that contains the root, one more digit in our estimate will be stabilized. The first six stages are described in the table below. They tell us  $\sqrt{2}/2 = .707106\dots$  to six decimal places accuracy.

Since this method of finding roots requires only that we be able to plot successive magnifications of the graph of  $f$  on a computer screen, the method can be applied to any function that can be entered into a computer.

The positive root of  $1 - 2x^2$

when the root lies between:		the decimal expansion
lower value	upper value	of the root begins with
.70	.80	.7
.700	.710	.70
.7070	.7080	.707
.70710	.70720	.7071
.707100	.707110	.70710
.7071060	.7071070	.707106
⋮	⋮	⋮

### An algebraic approach – the Babylonian algorithm

About 4000 years ago Babylonian builders had a method for constructing the square root of a number from a sequence of successive approximations. To demonstrate the method, we'll construct  $\sqrt{5}$ . We want to find  $x$  so that

$$x^2 = 5 \quad \text{or} \quad x = \frac{5}{x}.$$

The second expression may seem a peculiar way to characterize  $x$ , but it is at least equivalent to the first. The advantage of the second expression is that it gives us *two* numbers to consider:  $x$  and  $5/x$ .

For example, suppose we guess that  $x = 2$ . Of course this is incorrect, because  $2^2 = 4$ , not 5. The two numbers we get from the second expression are 2 and  $5/2 = 2.5$ . One is smaller than  $\sqrt{5}$ , the other is larger (because  $2.5^2 = 6.25$ ). Perhaps their *average* is a better estimate. The average is 2.25, and  $2.25^2 = 5.0625$ .

Although 2.25 is not  $\sqrt{5}$ , it is a better estimate than either 2.5 or 2. If we change  $x$  to 2.25, then

$$x = 2.25, \quad \frac{5}{x} = 2.222, \quad \text{and their average is } 2.236.$$

Is 2.236 a better estimate than 2.5 or 2.222? Indeed it is:  $2.236^2 = 4.999696$ . If we now change  $x$  to 2.236, a remarkable thing happens:

$$x = 2.236, \quad \frac{5}{x} = 2.236, \quad \text{and their average is } 2.236!$$

In other words, if we want accuracy to three decimal places, we have already found  $\sqrt{5}$ . All the digits have stabilized.

Suppose we want *greater* accuracy? Our routine readily obliges. If we set  $x = 2.236000$ , then

$$x = 2.236000, \quad \frac{5}{x} = 2.236136, \quad \text{and their average is } 2.236068.$$

In fact,  $\sqrt{5} = 2.236068\dots$  is accurate to six decimal places.

Here is a summary of the argument we have just developed, expressed in terms of  $\sqrt{a}$  for an arbitrary positive number  $a$ .

If  $x$  is an estimate for  $\sqrt{a}$ ,  
 then the average of  $x$  and  $a/x$   
 is a better estimate.

Once we choose an *initial* estimate, this argument constructs a sequence of successive approximations to  $\sqrt{a}$ . The process of constructing the sequence is called the **Babylonian algorithm** for square roots.

A procedure that tells us how to carry out a sequence of steps, one at a time, to reach a specific goal is called an **algorithm**. Many algebraic processes are algorithms. The word is a Latinization of the name of the Persian astronomer Muhammad al-Khwārizmī (c.780 A.D.–c.850 A.D.), who lived and worked in Baghdad. The title of his seminal book *Hisāb al-jabr wal-muqā bala* (830 A.D.)—usually referred to simply as *al-Jabr*—has a Latin form that is even more familiar to mathematics students.

The Babylonian algorithm takes the current estimate  $x$  for  $\sqrt{a}$  that we have at each stage and says “replace  $x$  by the average of  $x$  and  $a/x$ .” This kind of instruction is ideally suited to a computer, because  $A = B$  in a computer program means “replace the current value of  $A$  by the current value of  $B$ .” In the program BABYLON printed below, the algorithm is realized by a FOR - NEXT loop with a single line that reads “ $x = (x + a / x) / 2$ ”.

The three-step procedure that we used in chapter 1 to obtain values of  $S$ ,  $I$ , and  $R$  in the epidemic problem is also an algorithm, and for that reason it was a straightforward matter to express it as the computer program SIRVALUE.

**Program: BABYLON**  
**An algorithm to find  $\sqrt{a}$**

a = 5	Output:
x = 2	2.25
n = 6	2.236111
FOR k = 1 TO n	2.236068
x = (x + a / x) / 2	2.236068
PRINT x	2.236068
NEXT k	2.236068

## Exercises

In many of the questions below you are asked to do things like divide an arc into 2,000,000 segments or find a certain number to 8 decimal places. This assumes you have fast computers available. If you are using slower machines or programmable calculators, you should certainly feel free to scale back what is called for—perhaps to 200,000 segments or 6 decimal places. Use your own common sense; there's no value in sitting in front of a screen for an hour waiting for an answer to emerge. To approximate a length by 200,000 segments will take 10 times as long as to approximate it by 20,000 segments, which in turn will take 10 times as long as the 2,000 segment approximation. If you start with the cruder approximations, you should be able to get a good sense of what is reasonable to attempt with the facilities you have available, and modify what the problems call for accordingly.

Use common sense in deciding how close an approximation to make

1. By using a computer to graph  $y = x^2 - 2^x$ , find the solutions of the equation  $x^2 = 2^x$  to four decimal place accuracy.

### The program LENGTH

2. Run the program LENGTH to verify that it gives the lengths of the individual segments and their total length.
3. What line in the program gives the instruction to work with the function  $f(x) = x^2$ ? What line indicates the number of segments to be measured?
4. Each segment has a left and a right endpoint. What lines in the program designate the  $x$ - and  $y$ -coordinates of the left endpoint; the right endpoint?
5. Where in the program is the length of the  $k$ -th segment calculated? The segment is treated as the hypotenuse of a triangle whose length is measured by the Pythagorean theorem. How is the base of that triangle denoted in the program? How is the altitude of that triangle denoted?
6. Modify the program so that it uses 20 segments to estimate the length of the parabola. What is the estimated value?
7. Modify the program so that it estimates the length of the parabola using 200, 2,000, 20,000, 200,000, and 2,000,000 segments. Compare your

results with those tabulated below. [To speed the process up, you will certainly want to delete the `PRINT k, segment` statement that appears inside the loop. Do you see why?]

Number of line segments	Sum of their lengths
2	1.460 404 813
20	1.478 756 512
200	1.478 940 994
2 000	1.478 942 839
20 000	1.478 942 857
200 000	1.478 942 857
2 000 000	1.478 942 857

8. What is the length of the parabola  $y = x^2$  over the interval  $0 \leq x \leq 1$ , correct to 8 decimal places? What is the length, correct to 12 decimal places?

9. Starting at the origin, and moving along the parabola  $y = x^2$ , where are you when you've gone a total distance of 10?

10. Modify the program to find the length of the curve  $y = x^3$  over the interval  $0 \leq x \leq 1$ . Find a value that is correct to 8 decimal places.

11. **Back to the circle.** Consider the unit circle centered at the origin. Pythagoras' Theorem shows that a point  $(x, y)$  is on the circle if and only if  $x^2 + y^2 = 1$ . If we solve this for  $y$  in terms of  $x$ , we get  $y = \pm\sqrt{1 - x^2}$ , where the plus sign gives us the upper half of the circle and the minus sign gives the lower half. This suggests that we look at the function  $g(x) = \sqrt{1 - x^2}$ . The arclength of  $g(x)$  over the interval  $-1 \leq x \leq 1$  should then be exactly  $\pi$ .

- Divide the interval into 100 pieces—what is the corresponding length?
- How many pieces do you have to divide the interval into to get an accuracy equal to that of Archimedes?
- Find the length of the curve  $y = g(x)$  over the interval  $-1 \leq x \leq 1$ , correct to eight decimal places accuracy.

12. This question concerns the function  $h(x) = \frac{4}{1 + x^2}$ .

- a) Sketch the graph of  $y = h(x)$  over the interval  $-2 \leq x \leq 2$ .
  - b) Find the length of the curve  $y = h(x)$  over the interval  $-2 \leq x \leq 2$ .
13. Find the length of the curve  $y = \sin x$  over the interval  $0 \leq x \leq \pi$ .

### The program BABYLON

14. Run the program BABYLON on a computer to verify the tabulated estimates for  $\sqrt{5}$ .

15. In whatever computer language you are using, it should be possible to tell the computer to print out more decimals. Your teacher can tell you how this is handled on your computers. Modify BABYLON to run with at least 14-digit precision in this and the following problems. Also modify the program so it prints out the square of the estimate each time as well. What is the estimated value of  $\sqrt{5}$  in this circumstance? How many steps were needed to get this value? Use the square of this estimate as a measure of its accuracy. What is the square?

16. Use the Babylonian algorithm to find  $\sqrt{80}$ .

a) First use 2 as your initial estimate. How many steps are needed for the calculations to stabilize—that is, to reach a value that doesn't change from one step to the next?

b) Since  $9^2 = 81$ , a good first estimate for  $\sqrt{80}$  is 9. How many steps are needed this time for the calculations to stabilize? Are the final values in (a) and (b) the same?

17. Use the Babylonian algorithm to find  $\sqrt{250}$  and  $\sqrt{1990}$ . If you use 2 as the initial estimate in each case, how many steps are needed for the calculations to stabilize? If you use the integer nearest to the final answer as your initial estimate, then how many steps are needed? Square your answers to measure their accuracy.

18. The Babylonian algorithm is considered to be **very fast**, in the sense that each stage roughly doubles the number of digits that stabilize. Does your work on the preceding exercises confirm this observation? By comparison, is the routine that got the estimates for  $S(3)$  (computed with the program SIRVALUE) faster or slower than the Babylonian algorithm?

## 2.4 Chapter Summary

### The Main Ideas

- The exact numerical value of a quantity may not be known; the value is often given by an **approximation**.
- A numerical quantity is often given as the **limit** of a sequence of **successive approximations**.
- When a particular digit in a sequence of successive approximations **stabilizes**, that digit is assumed to appear in the limit.
- **Euler's method** is a procedure to construct a sequence of increasingly better **approximations** of a function defined by a set of rate equations and initial conditions. Each approximation is a piecewise linear function.
- The exact function defined by a set of rate equations and initial conditions can be expressed as the **limit** of a sequence of **successive Euler approximations** with smaller and smaller step sizes.

### Expectations

- You should be able to use a program to **construct a sequence** of estimates for  $S$ ,  $I$ , and  $R$ , given a specific  $S$ - $I$ - $R$  model with initial conditions.
- You should be able to **modify** the SIRVALUE and SIRPLOT programs to construct a sequence of estimates for the values of functions defined by other rate equations and initial conditions.
- You should be able to use programs that construct a sequence of **Euler approximations** for the function defined by a rate equation with an initial condition. The programs should provide both tabular and graphical output.
- You should be able to estimate the values of the roots of an equation  $f(x) = 0$  using a **computer graphing package**.

- You should be able to find a square root using the **Babylonian algorithm**.
- You should be able to find the length of any piece of any curve.

## Chapter Exercises

1. a) We have considered the logistic equation

$$y' = .1y \left( 1 - \frac{y}{1000} \right)$$

On page 81 we looked at the resulting graph of  $y$  vs.  $t$ , using the starting value  $y(0) = 100$ . There is another graphical interpretation, though, that is also instructive. Note that the logistic equation specifies  $y'$  as a function of  $y$ . Sketch the graph of this function. That is, plot  $y$  values on the horizontal axis and  $y'$  values on the vertical axis; points on the graph will thus be of the form  $(y, y')$ , where the  $y'$ -coordinate is the value given by the logistic equation for the given  $y$  value. What shape does the graph have?

- b) For what value of  $y$  does this graph take on its largest value? Where does this  $y$  value appear in the graph of  $y(t)$  versus  $t$ ?
- c) For what values of  $y$  does the graph of  $y'$  versus  $y$  cross the  $y$ -axis? Where do these  $y$  values appear in the graph of  $y(t)$  versus  $t$ ?

## Grids on Graphs

When writing a graphing program it is often useful to have the computer draw a grid on the screen. This makes it easier to estimate numerical values, for instance. We can use a simple FOR–NEXT loop inside a program to do this. For instance, suppose we had written a program (SIRPLOT or SEQUENCE, for instance) with the graphics already in place. Suppose the screen window covered values from 0 to 100 horizontally, and 0 to 50,000 vertically. If we wanted to draw 21 vertical lines (including both ends) spaced 5 units apart and 11 horizontal lines spaced 5,000 units apart, the following two loops inserted in the program would work:

```

FOR k = 0 TO 20
    Plot the line from (5 * k, 0) to (5 * k, 50000)
NEXT k
FOR k = 0 TO 10
    Plot the line from (0, 5000 * k) to (100, 5000 * k)
NEXT k

```

You should make sure you see how these loops work and that you can modify them as needed.

2. If the screen window runs from  $-20$  to  $120$  horizontally, and  $250$  to  $750$  vertically, how would you modify the loops above to create a vertical grid spaced  $10$  units apart and a horizontal grid spaced  $25$  units apart?
3. Go back to our basic  $S$ - $I$ - $R$  model. Modify SIRPLOT to calculate and plot on the same graph the values of  $S(t)$ ,  $I(t)$ , and  $R(t)$  for  $t$  going from  $0$  to  $120$ , using a stepsize of  $\Delta t = .1$ . Include a grid with a horizontal spacing of  $5$  days, and a vertical spacing of  $2000$  people. You should get something that looks like this:

