CHAPTER 5

NUMERICAL SOLUTION OF RATE EQUATIONS

Chapter 1 showed briefly that rate equations (differential equations) could be solved analytically using integral calculus. However, many differential equations of biological interest are not easily solved analytically because of their nonlinear nature. These more complex equations may be solved with numerical integration, which requires a lot of repeated arithmetical calculation. This is the principal reason that biological simulation depends heavily on computers. Some elementary techniques of numerical integration will be introduced below, using simple equations.

This chapter is the key to understanding much of the whole field of biological simulation. Many exercises in following chapters require you to use numerical integration. Try to learn the techniques thoroughly before going on to other material.

5.1 The Fundamental Ideas of Numerical Integration

In its simplest form, numerical integration is easily understood. It employs the straightforward idea of finding an incremental change in a variable, and adding the change to obtain a new value for the variable. A bank does this, for example, when it calculates interest and adds it to your savings account. Likewise, to find the size of a population of organisms at time $t + \Delta t$, you would find the change in population that occurred in the interval Δt, and add it to the size of the population at time t. The size of the population (or of your savings account) may be found after any length of time by repeating the calculation and addition.

The computer is ideally designed to perform repetitive operations of this sort. The results of numerical integration are approximations of the analytical solution, but they can be highly accurate. The difference (error) between the analytical solution and the approximation is produced by various factors. Several exercises in this chapter are concerned with estimating error, and you will find that it can be minimized by selecting appropriate numerical techniques.

In most cases, the errors caused by using numerical integration are not serious in biological simulation, because we are more interested in the behavior of models than in precisely duplicating real data. We will discuss in this chapter a simplified version of only one elementary technique and some of its variations. More complete discussions of these and other methods may be found in Bronson (1973), Carnahan et al. (1969), Macon (1963), and Davies (1971).

5.2 The Finite Difference Approach

You will remember that in Chapter 1 we worked with the derivative of the differential equation, dy/dx. All numerical integration techniques are based on the fact that for small incremental changes in x (the independent variable), the difference quotient $\Delta y/\Delta x$ approximates dy/dx. You may recall from elementary calculus that dy/dx is actually defined as:

$$\frac{dy}{dx} = \lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x}$$

Therefore, if

$$\frac{dy}{dx} = f(y) \tag{5.1}$$

then

$$\frac{\Delta y}{\Delta x} \cong f(y)$$

and with rearrangement

$$\Delta y \cong f(y) \cdot \Delta x \tag{5.2}$$

This leads to the general equation upon which the techniques of numerical integration are based:

$$y_{x+\Delta x} = y_x + \Delta y_x \tag{5.3}$$

Equations 5.2 and 5.3 are used together, with 5.2 defining the change in y for a finite change in x, and 5.3 defining the new value of y after the change.

The mathematically knowledgeable reader will notice that Equation 5.1 is a limited case of the customary expression for the first-order differential equation $dy/dx = f(x, y)$. The simpler form of Equation 5.1 is used here because in most biological simulation models, y is not a function of x.

5.3 The Euler Technique

Equations 5.2 and 5.3 actually describe a technique for solving rate equations called the Euler method, or Euler-Cauchy method. To show how this method would work with a specific example, we will examine the differential equation for exponential growth:

$$\frac{\Delta N}{\Delta t} \cong f(N) = kN \tag{5.4}$$

You will recall from Section 1.1 that N is population density and k is the constant for growth rate. Like many differential equations in biological models, the independent variable is time. This differential equation is approximated by the following difference equation:

$$\frac{\Delta N}{\Delta t} \cong kN \tag{5.5}$$

or

$$\Delta N_t \cong kN_t \Delta t \tag{5.6}$$

and the new value of N is obtained with the equation

$$N_{t+\Delta t} = N_t + \Delta N_t \tag{5.7}$$

To carry out an Euler integration, increment time by units of size Δt, and at each time interval, calculate the change in N, and then the new value of the population density, $N_{t+\Delta t}$. This new value of N then becomes the N_t for the following time interval.

This integration is simple enough to be calculated with a single equation combining Equations 5.6 and 5.7. However, it is better to develop the habit of first calculating the change with an equation like 5.6, and then updating the variable with an equation like 5.7. The computer may be programmed to carry out these calculations with two steps:

$$\Delta N \leftarrow k * N * \Delta t$$
$$N \leftarrow N + \Delta N$$

This "two-stage" approach will help to keep your programs clear when the models become more complex in the following chapters. The left-arrow symbol ($\leftarrow$) here indicates the computer operation of performing the calculation on the right-hand side and storing the result under the variable label on the left-hand side. In BASIC this operation is indicated by the equal sign (=).

Exercise 5-1: Examine the model for inhibited growth from Chapter 1. For Equation 1.16 find the difference equations analogous to 5.6 and 5.7. Write and implement a computer program using your equations to find and plot population size from $t = 0$ to 52 weeks.

Use the appropriate constants from Chapter 1, and a time increment Δt of 1. For comparison, have your program also plot values from Equation 1.17, the analytical solution. (Try to plot the Euler values as circles and the analytical values as a continuous line on the same graph.)

5.4 Time Increment and Error in Numerical Integration

The size of Δt is important in determining the error between the analytical solution of a differential equation and the numerical solution. The farther Δt departs from the limit of 0, the farther the results of the simple Euler method will depart from the true value of the integral. This is shown graphically in Figure 5.1, which compares the analytical solution of the exponential growth model (Equation 1.1) with Euler integration, using two different values for Δt.

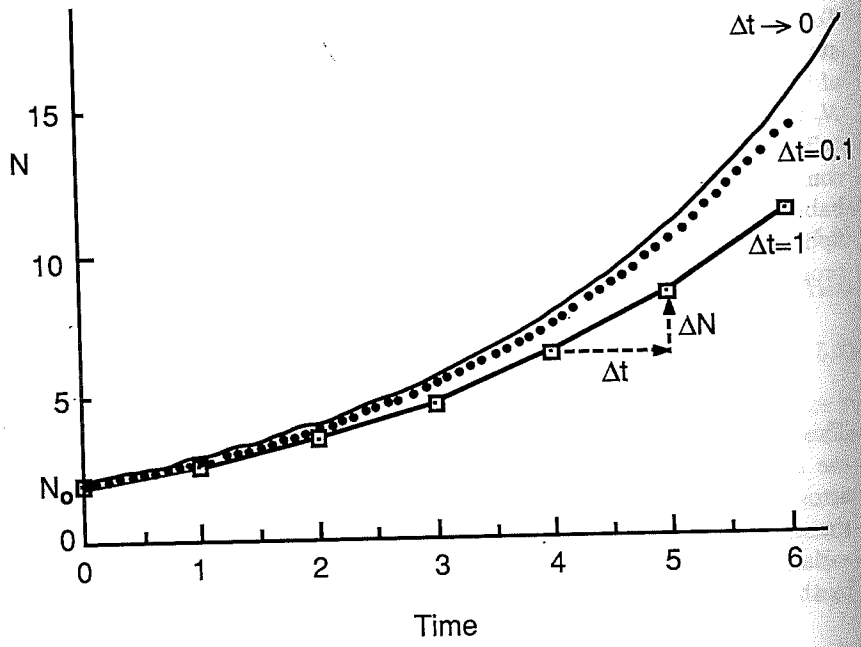

Figure 5.1. The effect of the time increment value on a simple Euler integration.

Error for a single step is directly proportional to the size of Δt. In

addition, error is cumulative, because each new value of N depends on the previous value. Selection of smaller values of Δt will decrease both kinds of error. The error in a single step is termed local error or truncation error, while the cumulative error due to previous steps is termed propagated error. Both can be minimized by selecting more accurate methods, as well as by decreasing Δt. A third type of error, termed rounding or round-off, occurs because computers can hold only a finite number of digits in their memories. The magnitude of this error will vary among types of computers.

Exercise 5-2: To estimate the magnitude of the errors due to Δt, write a computer program to compare three solutions for the exponential growth model (Equation 1.3):

(A) the analytical solution (Equation 1.6);

(B) Euler integration with $\Delta t = 1.0$;

(C) Euler integration with $\Delta t = 0.1$.

Use the two-stage approach with Euler integration. Set $N_0 = 10$, $k = 0.10$, and let t run from 0 to 30. Your output should be in the form of a table, with the following information printed out at each whole-integer time interval:

(1) time t;

(2) the true value of N_t calculated with the analytical solution;

(3) N_t from the Euler integration with $\Delta t = 1$;

(4) N_t from the Euler integration with $\Delta t = 0.1$;

(5) the difference between (2) and (3);

(6) the difference between (2) and (4).

Note that your program will not require the graphing subroutines of previous exercises. An easy method of programming this problem is to use nested FOR...NEXT loops, one loop for the integer time interval, and an inner loop for the smaller time increments.

Exercise 5-3: Modify and run the program from Exercise 5-2, using Δt values of 1.0 and 0.01.

5.5 The Improved Euler Method

As you discovered in Exercise 5-3, reducing the time increment Δt decreases the error, but also causes a proportional increase in computer time required to perform a given integration. Complex biological simulations may require highly accurate integration to work properly and will cause great increases in computer times that become inconvenient, expensive, or both. Hence, there has been a lot of mathematical research on methods which reduce the error other than by decreasing Δt.

Most of the error in the simple Euler method applied to the growth model occurs because it proceeds as if the slope in the interval Δt has a constant value which is a function of N_t. In fact, the slope changes continuously between N_t and $N_{t+\Delta t}$.

The Improved Euler is based on finding a better estimate of slope within the interval Δt. (The description here is based on the general notation of Equations 5.1-5.3 above.) First one finds an estimate of slope at y_t and then an estimate of slope at $y_{t+\Delta t}$. The improved slope is the average of these two estimates. The equations which accomplish this estimation are as follows:

$$\Delta y'_t = f(y_t)\,\Delta t \tag{5.8}$$

$$y'_{t+\Delta t} = y_t + \Delta y'_t \tag{5.9}$$

$$\Delta y''_t = f\left(y'_{t+\Delta t}\right)\,\Delta t \tag{5.10}$$

$$\Delta y_t = \frac{\Delta y'_t + \Delta y''_t}{2} \tag{5.11}$$

$$y_{t+\Delta t} = y_t + \Delta y_t \tag{5.12}$$

In this set of equations, $\Delta y'_t$ is the first estimate of the slope and is used to find the second estimate, $\Delta y''_t$. These two estimates are averaged in Equation 5.11, and the average is used to update y_t in Equation 5.12. The technique is illustrated graphically in Figure 5.2. The analogous equations for exponential growth (Equation 1.3) would be:

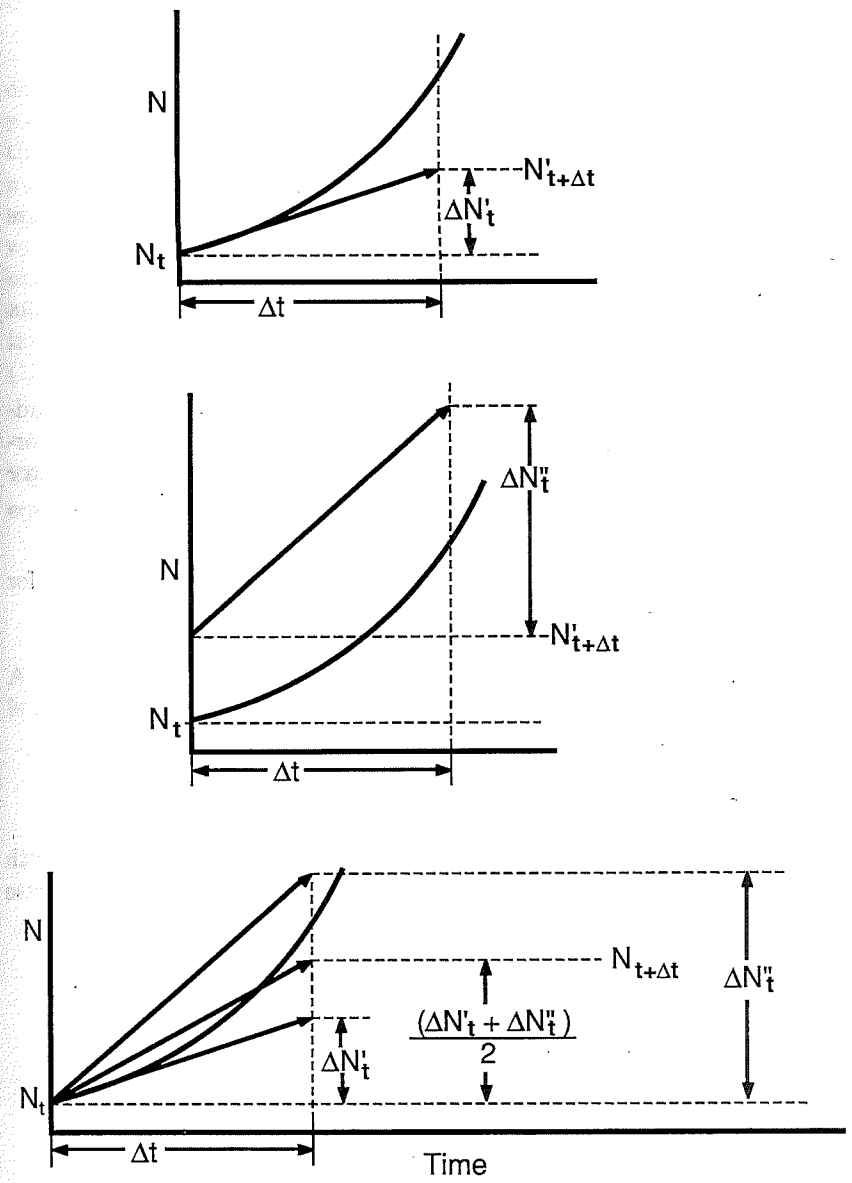

Figure 5.2. Graphical description of the Improved Euler (second-order Runge-Kutta) method. Top: first estimate of slope, $\Delta N'$. Middle: second estimate of slope, $\Delta N''$. Bottom: estimate of slope ΔN from the average of $\Delta N'$ and $\Delta N''$.

$$\Delta N'_t = kN_t \Delta t$$

$$N'_{t+\Delta t} = N_t + \Delta N'_t$$

$$\Delta N''_t = kN'_{t+\Delta t} \Delta t$$

$$\Delta N_t = \frac{\Delta N'_t + \Delta N''_t}{2}$$

$$N_{t+\Delta t} = N_t + \Delta N_t$$

The Improved Euler method has been identified by various terms, including Heun's method, second-order Runge-Kutta, and a simplified predictor-corrector method. In the latter terminology, Equation 5.9 is the predictor equation, and Equations 5.10-5.12 are the corrector step.

Exercise 5-4: Write a computer program to compare three methods for integrating Equation 1.3, the exponential growth model:

(A) The analytical (true) solution;

(B) the simple Euler integration;

(C) Improved Euler integration.

Use the same approach and parameters as in Exercise 5-2, with $\Delta t = 0.1$. Your output should also follow the format of Exercise 5-2, showing a tabulation of:

(1) time (t);

(2) Values of N_t based on method (A) above;

(3) Values of N_t based on method (B) above;

(4) Values of N_t based on method (C) above;

(5) the difference between (A) and (B);

(6) the difference between (A) and (C).

5.6 A Fourth-Order Runge-Kutta Method

A great variety of techniques are available for numerical solutions to differential equations, of which Runge-Kutta methods are a particular set. In fact, the Euler method is a first-order Runge-Kutta method and the Improved Euler is a second-order Runge-Kutta. While Runge-Kutta methods of almost any order are possible, the fourth-order is convenient and most commonly used. The estimates it provides can be quite accurate, even with the simplest form of the method given here. The derivation of Runge-Kutta methods may be found in most textbooks of numerical analysis; the practical discussion of Carnahan et al. (1969) is particularly clear. For the following example, we will use the diffusion model (Equation 1.12):

$$\frac{dC}{dt} = -k\,(C - C_x)$$

The simple Euler (first-order Runge-Kutta) two-stage approach with this equation is

$$\Delta C \leftarrow -k(C - C_x)$$
$$C \leftarrow C + \Delta C$$

A BASIC program to accomplish a fourth-order Runge-Kutta integration is given here:

```
100 REM DIFFUSION MODEL
110 REM 4TH-ORDER RUNGE-KUTTA
120 K=0.3 :  C=100 :  CX = 50
130 DT = 0.1
140 FOR T=0 TO 20
150    IF T=0 THEN GOTO 330
160    FOR I=1 TO 10
170:
180      D1 = -K * (C - CX) * DT
190      C1 = C + D1/2
200:
210      D2 = -K * (C1 - CX) * DT
220      C2 = C + D2/2
230:
240      D3 = -K * (C2 - CX) * DT
250      C3 = C + D3
260:
270      D4 = -K * (C3 - CX) * DT
280:
```

```
290      DC = (D1 + (2*D2) + (2*D3) + D4) / 6
300      C = C + DC
310:
320    NEXT I
330    PRINT T,C
340 NEXT T
350 END
```

In essence, this method provides four different estimates of the slope (ΔC) over the increment Δt (lines 180, 210, 240, 270) and then finds a weighted average of these estimates (line 290). The variable is updated with the weighted average in line 300. As part of the updating process, ΔC_1 and ΔC_2 are divided by 2 (lines 190 and 220). Note that this listing is designed for clarity, not computational efficiency.

Exercise 5-5: Write a program to compare four methods for integrating Equation 1.3 (exponential growth):

(A) the analytical (true) solution;

(B) the simple Euler integration;

(C) the Improved Euler integration;

(D) the 4th-order Runge-Kutta method.

Implement your program using the approach and parameters in Exercise 5-2, with $\Delta t = 0.1$. Your output should show a tabulation of time, and the values from methods (A), (B), (C), and (D).

Exercise 5-6: Using the model for exponential growth (Equation 1.3), write a program that will allow you to vary Δt values for numerical integration with the simple Euler method. Find the largest Δt value that will give the same accuracy (rounded to 5 decimal places) as a 4th-order Runge-Kutta solution with $\Delta t = 0.1$, evaluated after 20 time units.

5.7 Systems of Equations

Many of the models to be investigated in subsequent chapters are developed using several equations that contain two or more common variables.

In a simple, two-variable form, these will make up a system of equations:

$$\frac{dy}{dx} = f_1(y, z) \tag{5.13a}$$

$$\frac{dz}{dx} = f_2(y, z) \tag{5.13b}$$

with values available for y and z at $x = 0$. An elementary example of a system like this has already been examined in Section 1.8. Such systems of equations are easily solved with the numerical integration methods you have looked at in the sections above. The only difficulty is to keep in mind the proper sequence.

The fundamental rule for solving systems of equations is that they are taken through the steps of numerical integration in parallel. If you are using a two-equation system with the simple Euler method, both equations will be taken through the first stage (calculating the changes), and then through the update stage (finding the new value).

Listed below is another example, a program that uses the Improved Euler method with the bimolecular reaction model (Equation 1.18):

$$\frac{d[B]}{dt} = f([A], [B]) = -k[A][B]$$

Note the second equation is

$$\frac{d[A]}{dt} = f([A], [B]) = -k[A][B]$$

In this case the function equations are the same, although this will rarely be the case in the chapters that follow. (For this example, we will assume that the starting concentrations of A and B are similar but not equal, so that neither Equations 1.19 nor 1.22 apply.)

```
100 REM BIMOLECULAR REACTIONS
110 REM IMPROVED EULER METHOD
120 A=100:  B=110:  K=0.1
130 DT = 0.1
140 FOR T = 0 TO 25
150    IF T=0 THEN GOTO 330
160    FOR I = 1 TO 10
170:
180      D1A = -K * A * B * DT
190      D1B = -K * A * B * DT
200:
```

```
210      AX = A + D1A
220      BX = B + D1B
230:
240      D2A = -K * AX * BX * DT
250      D2B = -K * AX * BX * DT
260:
270      DA = (D1A + D2A) / 2
280      DB = (D1B + D2B) / 2
290:
300      A = A + DA
310      B = B + DB
320    NEXT I
330    PRINT T,A,B
340 NEXT T
350 END
```

The key point of this listing is that the two variables $[A]$ and $[B]$ are taken through the updating process in parallel. (As before, this program listing is designed for clarity, rather than efficiency in computation.)

Exercise 5-7: Following the format of the listing of the Improved Euler method above, write and implement a program for the bimolecular reaction model using a fourth-order Runge-Kutta method, with $k = 0.025$, $[A]_0 = 95$, and $[B]_0 = 115$. Your output should be a table of $[A]$ and $[B]$ for values of t from 0 to 20.

5.8 Fifth-Order Runge-Kutta

In most cases that are encountered in biological simulation, the Improved Euler or 4th-order Runge-Kutta methods will be adequate. Where extreme accuracy is required, a 5th-order Runge-Kutta method will usually provide it with minimal increases in computing time. The following listing uses a set of equations developed by Butcher (1964), which Waters (1966) found to have the best balance of accuracy and computing time (James et al. 1977).

```
100 REM EXPONENTIAL GROWTH
110 REM 5TH-ORDER RUNGE-KUTTA
120 N = 10
130 K = 0.10
140 DT = 0.1
150 FOR T=0 TO 30
```



```
160    IF T=0 THEN GOTO 390
170    FOR I = 1 TO 10
180:
190       D1 = K * N * DT
200       N1 = N + D1/4
210:
220       D2 = K * N1 * DT
230       N2 = N + D1/8 + D2/8
240:
250       D3 = K * N2 * DT
260       N3 = N - D2/2 + D3
270:
280       D4 = K * N3 * DT
290       N4 = N + D1*3/16 + D4*9/16
300:
310       D5 = K * N4 * DT
320       N5 = N - D1*3/7 + D2*2/7 + D3*12/7 + D4*12/7 +
330:
340       D6 = K * N5 * DT
350:
360       DN = ( 7*D1 + 32*D3 + 12*D4 + 32*D5 + 7*D6 ) / 90
370       N = N + DN
380    NEXT I
390    PRINT T, N
400 NEXT T
410 END
```

Here again, this listing is designed for clarity, not ease of computation. (The omission of a term containing D2 from line 360 is intentional.) An experienced BASIC programmer can shorten the listing and decrease computational time considerably with algebraic manipulation of the equations and use of the DEF FN statement.

Conclusion

This chapter has described simple numerical methods for solving differential equations. The exercises have been designed to give you some idea of the accuracy you might expect with each method. Most of our simulations in the following chapters can be performed with the simple Euler or Improved Euler techniques. We will be primarily concerned with the behavior of the models rather than precise duplication of complex analytical solutions.