CHAPTER 2

Some Fundamental Concepts in the Numerical Solution of Differential Equations

2.1 EULER'S METHOD

Before efficiently using numerical methods to solve d.e.'s, it is necessary to understand some fundamental concepts. The approach to be used here for learning these is to apply Euler's method. Euler's method is not, in my opinion, good enough for useful application today; but it is simple and beautiful, and enables us to understand notions that are very important in all methods.

You will need your d.e. and, to start with, your direction field diagram. You are going to use the little lines in the diagram to approximate a solution. The differential equation with initial conditions that we shall discuss is

$$\frac{dy}{dx} = f(x,y); \quad y(x_0) = y_0. \tag{2.1.1}$$

We shall have the following adventure. First, stand at the point (x_0,y_0). Assuming that the rules guaranteeing existence and uniqueness are satisfied, there is just one solution passing through this point; we want to travel along it. For a start, the direction field will tell us in what direction we must face. We now take a step in that direction, so that at the end of the step the x-coordinate has increased by the quantity h. h is called the *step size*; you are going to be very concerned with it. (h is almost invariably positive, but this is not essential.) Notice that h is not the length of the physical step taken, but is the increment in x, the independent variable. Now, what is the value of y at the end of the step? The increment in y will be the step size h multiplied by the slope of the solution curve at the point (x_0,y_0). This slope is $f(x_0,y_0)$. Then if the coordinates at the end of this step are (x_1,y_1), we have

$$x_1 = x_0 + h, \; y_1 = y_0 + hf(x_0,y_0). \tag{2.1.2}$$

We have just taken one *step* using Euler's method.

A beauty of this method is that once we have reached the point (x_1,y_1), we can follow the procedure all over again, with x_1 and y_1 substituted for x_0 and y_0. Let us take another step, using the same step size h. If this brings us to (x_2,y_2), then

$$x_2 = x_1 + h, \; y_2 = y_1 + hf(x_1,y_1). \tag{2.1.3}$$

Now take another step, and I give you one guess as to where it gets you!

In general, if we are at (x_n,y_n), then the next position will have coordinates

$$x_{n+1} = x_n + h,\ y_{n+1} = y_n + hf(x_n,y_n). \tag{2.1.4}$$

With starting conditions $x = x_0$ and $y = y_0$, (2.1.4) is used recursively for $n = 0,1,2, \ldots$ until we have had enough. This is Euler's method; it is like stepping from line to line in a direction field. It is an example of a *single-step* method, where each step taken is similar to the first. (There also exist *multistep* methods, where some past history of a solution is used when taking a step. It is as if you were looking back over your shoulder to help you get a smoother path.) Euler's method is extravagant and inaccurate, but provides a good learning tool, and we shall use it—once.

First, look at some numbers. Consider

$$y' = y + x;\quad y(0) = -0.5. \tag{2.1.5}$$

Let $h = 0.1$. For successive iterations, we have:

$$x_0 = 0.000000,\quad y_0 = -0.500000,$$

so

$$y_0' = f(x_0,y_0) = y_0 + x_0 = -0.500000.$$

$$x_1 = x_0 + h = 0.100000,\quad y_1 = y_0 + hf(x_0,y_0) = -0.550000,$$

so

$$y_1' = f(x_1,y_1) = y_1 + x_1 = -0.450000.$$

$$x_2 = x_1 + h = 0.200000,\quad y_2 = y_1 + hf(x_1,y_1) = -0.595000,$$

so

$$y_2' = f(x_2,y_2) = y_2 + x_2 = -0.395000.$$

$$x_3 = x_2 + h = 0.300000,\quad y_3 = y_2 + hf(x_2,y_2) = -0.634500.$$

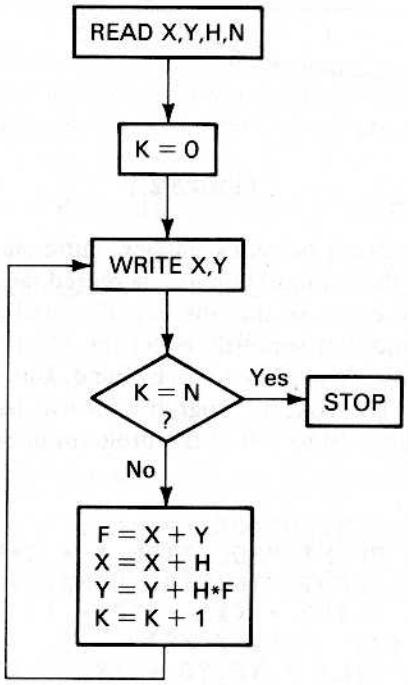
Can you find x_4 and y_4? All right, write them down.

This method can be very easily used on a programmable hand calculator. Alternatively, a FORTRAN-like program might go as follows:

```
C     PROGRAM FOR APPLYING EULER'S METHOD TO
C     Y' = F(X,Y), WITH INITIAL CONDITIONS
C     Y(X0) = Y0, AND STEP-SIZE H.
C     N STEPS WILL BE TAKEN.
C     ENTER THE INITIAL CONDITIONS AND STEP SIZE.
      X = 0.0
      Y = - 0.5
      H = 0.1
```

```
C     NOW WRITE OUT X AND Y AT ONCE. YOU CAN'T HAVE
C     TOO MUCH PRINT-OUT.
C     FOR 20 STEPS,
      N = 20
      DO 10 I = 1,N
         F = X + Y
C        THE D.E. HERE IS Y' = X + Y
         X = X + H
         Y = Y + H*F
C        WRITE OUT X AND Y.
   10    CONTINUE
      STOP
      END
```

You may find it preferable to work from a flowchart of the sort shown below.



For your next project, choose initial conditions and then plot the correct solution curve for your own d.e. This is going to tell you how good the approximations are. Now pick a value for the step size h (how about $h = 0.1$ for a start?), and with the same initial conditions, calculate successive steps using Euler's method. Then mark these steps on your figure. Next, do the same for different values of h, both larger and smaller. You will finish up with a figure like Figure 2.1. Note the way in which the accuracy increases as h becomes

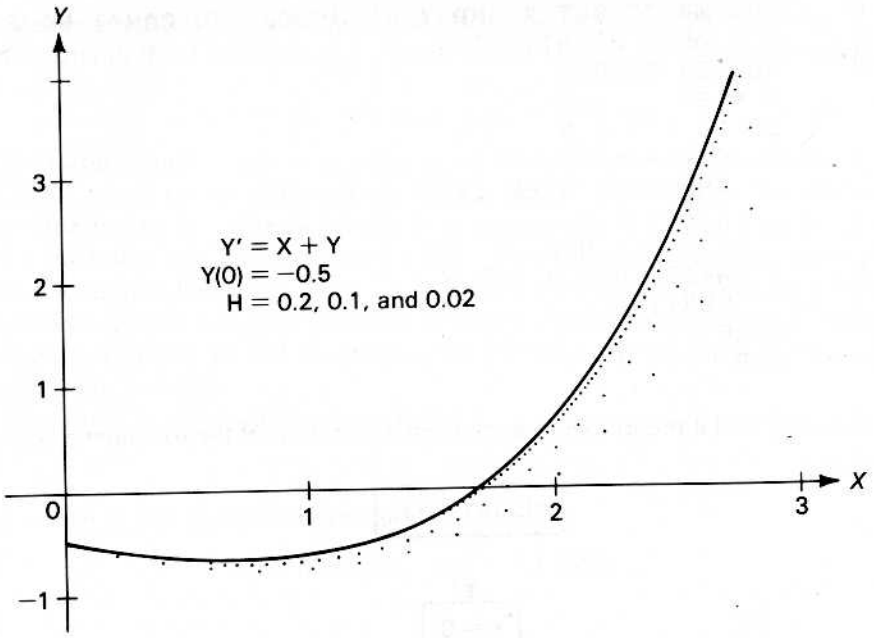


Figure 2.1

smaller. Note also that as h becomes smaller, more steps are needed, so that the time and effort of the computation are increased. Moreover, with each step there will be round-off error, so that the overall round-off error increases with the number of steps and may seriously affect the accuracy of the work.

The current project is a good one to do by hand. But the computer can make experimenting easier. The BASIC program that was used to construct Figure 2.1 consisted first of lines 10 to 160 of the program in Section 1.2, page 7, and then the following:

```
300  REM PLOT A SOLUTION
310  VTAB 23: PRINT "SOLUTION Y = C*EXP(X) - X - 1"
320  VTAB 24: INPUT "INITIAL CONDITIONS ";X0,Y0
330 C = EXP ( - X0) * (X0 + Y0 + 1)
340 N = INT ((XU - X0) / XF)
350 XB = (X0 - XL) / XF:YB = (YU - Y0) / YFAC
360  HPLOT XB,YB
370  FOR I = 1 TO N
380 XB = XB + 1
390 X = XL + XB * XF:Y = C * EXP (X) - X - 1
400 YB = (YU - Y) / YF
410  IF YB < 0 THEN GOTO 450
420  IF YB > 159 THEN GOTO 450
430  HPLOT  TO XB,YB
```

```
440  NEXT I
450  REM  PLOT EULER STEPS
460  INPUT "ENTER STEPSIZE ";H
470 X = X0:Y = Y0
480 Y = Y + H * (Y + X):X = X + H
490 XB = (X - XL) / XF:YB = (YU - Y) / YF
500  IF XB > 279 THEN  GOTO 460
510  IF YB < 0 THEN  GOTO 460
520  IF YB > 159 THEN GOTO 460
530  HPLOT XB,YB: GOTO 480
```

This program can be adapted for your own equation merely by changing lines 330, 390, and 480.

2.2 TRUNCATION ERROR

The discrepancy between the correct solution and a path taken using Euler's method is obvious. For instance, in the numerical example of Section 2.1, for $x = 0.1$, there is a difference between the correct and approximate values of y of

$$(0.5e^{.1} - 0.1 - 1) - (-0.55) = 0.002585 \ldots$$

This difference is called a *truncation error*. To understand the use of the word "truncation," look carefully at the solution of

$$y' = y + x; \quad y(0) = -0.5, \tag{2.2.1}$$

which is

$$y = 0.5e^x - x - 1. \tag{2.2.2}$$

So at the end of just one step with step size h, the correct solution would be

$$y(h) = 0.5e^h - h - 1. \tag{2.2.3}$$

The approximate solution is $y_1 = -0.5 - 0.5h$. Now, $y(h)$ can be expanded in powers of h as

$$y(h) = 0.5\left[1 + h + \frac{h^2}{2!} + \frac{h^3}{3!} + \ldots\right] - h - 1. \tag{2.2.4}$$

In the approximation of Euler's method, all the terms with powers of h higher than the first have been thrown out, so the infinite series of (2.2.4) has been truncated.

The truncation error of just one step is called the *local truncation error*. In the integration method that we are soon to learn, this will be carefully controlled by adjusting the step size h. The *accumulated* truncation error over several steps

is going to depend not only on the method of solution used and the step size, but also on the differential equation being solved. For (1.1.2), the accumulated truncation error is certain to increase without bound as x increases; it will, in fact, increase exponentially. But for (1.1.5), where all the solutions become funnelled together, this will not be the case. What the truncation error does is place us immediately on a different solution from the one having the given initial conditions. For example, if we continue the numerical example of Section 2.1, i.e.,

$$y' = y + x; \quad y(0) = -0.5, \qquad h = 0.1,$$

for ten steps, we find that $x_{10} = 1.0$, $y_{10} = 0.29687\ldots$. The solution that passes through *this* point is $y = Ce^x - x - 1$, where $C = 0.47709\ldots$, and for *this* solution, when $x = 0$, then $y = -0.52291\ldots$.

See if you can make similar comparisons for your own d.e.

2.3 THE ORDER OF A METHOD

Local truncation error depends on both the step size and the method being used. We next need to examine this dependency. For the example discussed in the preceding section, we can see that the local truncation error of the first step can be written as

$$y(h) - y_1 = \text{TE} = 0.5\left[\frac{h^2}{2!} + \frac{h^3}{3!} + \ldots\right]. \tag{2.3.1}$$

The leading term is factored by h^2, a characteristic of Euler's method.

For

$$y' = f(x,y); \quad y(x_0) = y_0, \tag{2.3.2}$$

with step size h, at the end of one step,

$$x_1 = x_0 + h, \quad y_1 = y_0 + hf(x_0,y_0). \tag{2.3.3}$$

If the correct solution is $y = y(x)$, we can write

$$\begin{aligned} y(x_1) &= y(x_0 + h) \\ &= y(x_0) + hy'(x_0) + \frac{1}{2}h^2y''(x_0) + \ldots \\ &= y_0 + hf(x_0,y_0) + \frac{1}{2}h^2y''(x_0) + \ldots \\ &= y_1 + \frac{1}{2}h^2y''(x_0) + \ldots \end{aligned} \tag{2.3.4}$$

So the truncation error is

$$\text{TE} = y(x_1) - y_1 = \frac{1}{2}h^2y''(x_0) + \ldots \tag{2.3.5}$$

The terms omitted on the right all have powers of h higher than two.

Most methods for the numerical solution of d.e.'s proceed in steps and have a local truncation error that depends on the step size: the shorter the step, the smaller the error. Usually, the local truncation error can be expressed as a power series in increasing powers of h, as in (2.3.5), with leading term Ah^n. Then the local truncation error can be approximated by Ah^n in the sense that

$$\lim_{h\to 0}\{\text{TE}(h)/h^n\} = A. \tag{2.3.6}$$

The smaller the value of h, the better is the approximation. Much more to the point, the higher the value of n, the more rapidly does TE become small as h becomes small.

The number $n-1$ is called the *order* of the numerical method. The higher the order is, the more spectacular will be the increase in accuracy when h is diminished. It is clear that for Euler's method $n = 2$, so the order of the method is one. If we were to halve h, then the error of a step would be reduced roughly by a factor of four; but two steps would be needed where one sufficed before, so the overall error would only be halved. If h is reduced too much, then so many steps may be needed that round-off error can become important. Depending on the precision of the calculation, every arithmetical operation introduces round-off error in a random way, and the errors will be compounded. So it may be necessary to balance truncation error (by reducing h) with round-off error (using the highest safe h). In principle, the higher the order of a method, the larger is the permissible step size for a given accuracy requirement, and the lower the dangers from round-off error.

2.4 NUMERICAL CONFIRMATION OF THE ORDER OF EULER'S METHOD

It is very important that you acquire confidence in the order of whatever method you use, and that you be able to control the accuracy of your solutions through judicious handling of the step size. The method that we shall shortly learn makes clever use of the known order. Since the analytical justification of the order can be very complicated, it can be helpful to confirm it numerically. Your next project is to perform an experiment confirming the order of Euler's method.

To do this project, you will need to take several steps. Each step will start with the same initial conditions, but the step size will be varied. Thus, for

$$y' = y + x; \quad y(0) = -0.5, \tag{2.4.1}$$

the outcome of one step is

$$x_1 = h, \quad y_1 = -0.5 - 0.5h. \tag{2.4.2}$$

The correct value at the end of one step is

$$y(x_1) = 0.5e^h - h - 1, \tag{2.4.3}$$

and the truncation error is

$$\mathrm{TE}(h) = y(x_1) - y_1 \tag{2.4.4}$$

In worrying about error, we are concerned with its magnitude rather than its sign; so, consider

$$E(h) = |\, TE(h) \,|\,. \tag{2.4.5}$$

We want to justify the approximation

$$E(h) \simeq Ah^2 \tag{2.4.6}$$

for a small enough h. A standard procedure is to take logs, so that (2.4.6) becomes

$$\log[E(h)] \simeq \log(A) + 2\log(h). \tag{2.4.7}$$

Now if

$$X = \log(h), \quad Y = \log[E(h)], \tag{2.4.8}$$

then

$$Y \simeq \log(A) + 2X, \tag{2.4.9}$$

so that a graph of X and Y as h varies should approximate a straight line having slope 2.

For the example (2.4.1), I used values of

$$h = 0.05, 0.1, 0.15, \ldots, 0.45, 0.5.$$

For each of these values, I computed

$$y_1,\ y(x_1),\ \mathrm{TE}(h),\ E(h),\ X = \log(h) \text{ and } y = \log[E(h)].$$

(The base of the logarithm does not matter.) Figure 2.2 shows how the points appear in the X − Y plane, and how they lie convincingly close to a straight line having slope 2.

There are two ways in which you might run into trouble in this project. Notice that from (2.3.5) we have

$$A = \frac{1}{2}y''(x_0). \tag{2.4.10}$$

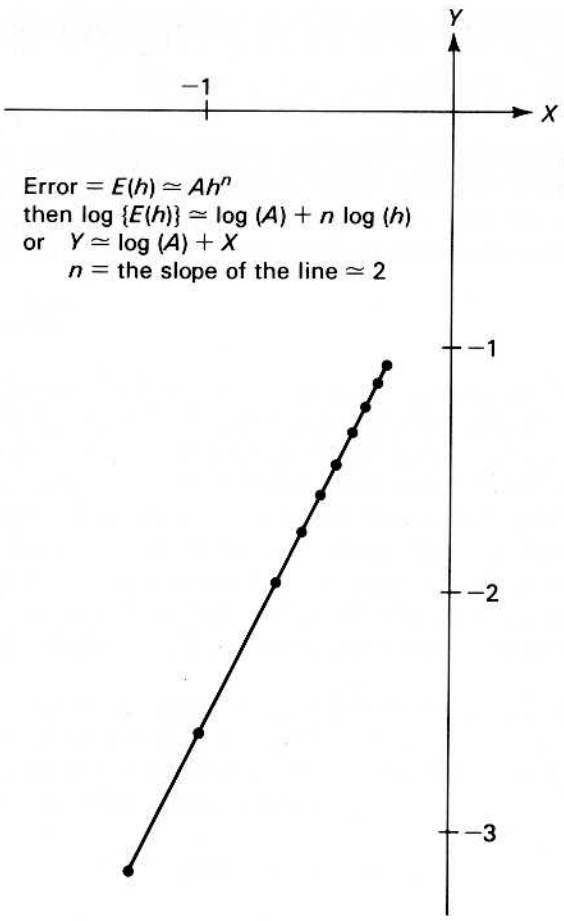


Figure 2.2

This might be very small or even zero. The value of A (when it is not too small) can be found from the intercept of the line given by (2.4.9) with the X-axis. It can also be predicted analytically since

$$y'' = \frac{d}{dx}[f(x,y)] = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial y}\frac{dy}{dx} = f_x + ff_y. \tag{2.4.11}$$

(Come on, it's *good* for you to recollect the chain rule!) Then

$$y''(x_0) = f_x(x_0,y_0) + f(x_0,y_0)f_y(x_0,y_0). \tag{2.4.12}$$

Calculate this value before you start your project, and make sure that it is not very small. If it is zero, then from (2.4.9), you may get a straight line having slope 3.

The other cause of trouble may be that you do not make h sufficiently small. In general, we have

$$\mathrm{TE}(h) = Ah^n + Bh^{n+1} + \ldots \tag{2.4.13}$$

For (2.4.6) to be convincing, h must be small enough for the second and subsequent terms on the right of (2.4.13) to be small compared with the first. I suggest that you use the same range of values for h that I used. If you have trouble, try smaller values: you might have an exceptional case where B is much greater than A.

CHAPTER 3

One Approach to Solving a System of Ordinary Differential Equations

3.1 GENERALITIES

Long courses are taught that survey different methods for the numerical solution of ordinary differential equations, but even they have no hope of being comprehensive. (I stress the word *ordinary*; even longer courses are needed for partial differential equations.) Among the treasury of methods, there is none that is "best." The principal method that I have chosen for this text is excellent and up to date; it allows control of the step size so that local truncation error can be bounded. It is, I hope you will come to agree, simple to understand, to program, and to use. But I do not wish to detract in any way from other approaches and other methods: I hope that you will soon have the luck to learn about them.

Not all processes can be derived as easily as Euler's method. It will be helpful to provide some more background. First, let us have a word about *numerical quadrature*, that is, the numerical approximation of a definite integral. Suppose that we had to solve the differential equation

$$\frac{dy}{dx} = f(x); \quad y(x_0) = y_0. \tag{3.1.1}$$

The solution can be formally written down as

$$y = y_0 + \int_{x_0}^{x} f(x)dx. \tag{3.1.2}$$

For most functions $f(x)$, a numerical method would have to be found to approximate the integral for a given value of x.

Consider, for example, the integral

$$I = \int_{a}^{a+h} f(x)dx. \tag{3.1.3}$$

The crudest approximation for this is given by the *rectangular rule*,

$$I \simeq I_R = hf(a). \tag{3.1.4}$$

A little better are the *mid-point rule*,

$$I \simeq I_M = hf(a + \tfrac{1}{2}h), \tag{3.1.5}$$

and the *trapezoidal rule,*

$$I \simeq I_T = \frac{h}{2}[f(a) + f(a + h)]. \tag{3.1.6}$$

A whole lot better is *Simpson's rule,*

$$I \simeq I_S = \frac{h}{6}[f(a) + 4f(a + \tfrac{1}{2}h) + f(a + h)]. \tag{3.1.7}$$

In each case, the error, for small enough h, can be approximated by a term of the sort Ah^n, and the quantity $n - 1$ is the *order* of the method. For the rectangular rule, $n = 2$; for the mid-point and trapezoidal rules, $n = 3$; and for Simpson's rule, $n = 5$. For these formulas, it is rather easy to find bounds for the errors. The following are noted for their interest; they will not be applied here.

$$\left.\begin{aligned} I &= I_R + \frac{1}{2}h^2 f'(z_R), \\ I &= I_M + \frac{1}{24}h^3 f''(z_M), \\ I &= I_T + \frac{1}{12}h^3 f''(z_T), \\ I &= I_S - \frac{1}{90}\left[\frac{h}{2}\right]^5 f^{(4)}(z_S). \end{aligned}\right] \tag{3.1.8}$$

The superiority of Simpson's rule is very clear. In each case the quantity z is some number lying between the two limits a and $a + h$. These formulas can be used repeatedly to build up a quadrature from a to b, where the interval $b - a$ is divided up into n steps, each of width h; so $nh = b - a$, where h is the step size, as before.

Now consider the d.e.

$$\frac{dy}{dx} = f(x,y); \quad y(x_0) = y_0. \tag{3.1.9}$$

Just suppose that we *knew* the solution $y = y(x)$ explicitly as a function of x. Then it would be correct to write down

$$y(x) = y_0 + \int_{x_0}^{x} f[x,y(x)]dx. \tag{3.1.10}$$

In fact this would be an identity. (3.1.10) is correct but not, at first sight, helpful. For in order to evaluate the integral, $y(x)$ must be known; but if $y(x)$ is known, there is no purpose in trying to evaluate the integral. But the expression turns out to be prodigiously helpful.

Suppose that the integral in (3.1.10) is approximated using the rectangular

rule. We shall write

$$x = x_1 = x_0 + h,$$

so we are considering just one *step* in the process. Then

$$y(x_0 + h) \simeq y_1 = y_0 + hf(x_0,y_0), \tag{3.1.11}$$

and we are back with Euler's method. Forget it!

Next, apply the mid-point rule. Then

$$y(x_0 + h) \simeq y_{\frac{1}{2}} + hf[x_0 + \tfrac{1}{2}h,\ y(x_0 + \tfrac{1}{2}h)]. \tag{3.1.12}$$

In order to deal with the term $y(x_0 + \frac{1}{2}h)$, recall the Taylor series

$$y(x_0 + k) = y(x_0) + ky'(x_0) + \frac{1}{2}k^2y''(x_0) + \ldots$$

(If it is rusty in your memory, please review it; it is needed several times in this chapter.) So

$$y(x_0 + \tfrac{1}{2}h) = y(x_0) + (\tfrac{1}{2}h)y'(x_0) + \frac{1}{2}(\tfrac{1}{2}h)^2y''(x_0) + \ldots$$

Remember that in making the mid-point approximation, terms with powers of h greater than the second are thrown out; so, continuing to throw out such terms, we are left with

$$y(x_0 + h) \simeq y_1 = y_0 + hf[x_0 + \tfrac{1}{2}h,\ y_0 + \tfrac{1}{2}hf(x_0,y_0)], \tag{3.1.13}$$

where we have used the relation $y'(x_0) = f(x_0,y_0)$. (3.1.13) provides one step using the *modified Euler's method.*

Next apply the trapezoidal rule to the integral in (3.1.10):

$$\begin{aligned} y(x_0 + h) &\simeq y_0 + \frac{1}{2}h\Big[f(x_0,y_0) + f[x_0 + h,\ y(x_0 + h)]\Big] \\ &= y_0 + \frac{1}{2}h\Big[f(x_0,y_0) + f[x_0 + h,\ y(x_0) + hy'(x_0) + \ldots]\Big] \\ &\simeq y_0 + \frac{1}{2}h\Big[f(x_0,y_0) + f[x_0 + h,\ y_0 + hf(x_0,y_0)]\Big]. \end{aligned}$$

If we write $f(x_0,y_0) = f_0$, the equation can be written as

$$y(x_0 + h) \simeq y_1 = y_0 + \frac{1}{2}h\Big[f_0 + f(x_0 + h,\ y_0 + hf_0)\Big]. \tag{3.1.14}$$

This provides one step in the *improved Euler's method.*

(3.1.13) and (3.1.14) are examples of *Runge-Kutta* methods. It can be shown that if Simpson's rule is applied to the integral in (3.1.10), then the following

formulas result: Let

$$\left.\begin{aligned} f_0 &= f(x_0, y_0), \\ f_1 &= f(x_0 + \tfrac{1}{2}h,\ y_0 + \tfrac{1}{2}hf_0), \\ f_2 &= f(x_0 + \tfrac{1}{2}h,\ y_0 + \tfrac{1}{2}hf_1), \\ f_3 &= f(x_0 + h,\ y_0 + hf_2); \end{aligned}\right] \tag{3.1.15}$$

then

$$y(x_0 + h) \approx y_1 = y_0 + \frac{1}{6}h(f_0 + 2f_1 + 2f_2 + f_3).$$

This is—justly—the most famous of the Runge-Kutta methods. I do not recommend it for this text because of problems with step size control. But it takes only a matter of minutes to program. If you are in a hurry for results and have no available program, then it is a good method to consider. It is a fourth-order method.

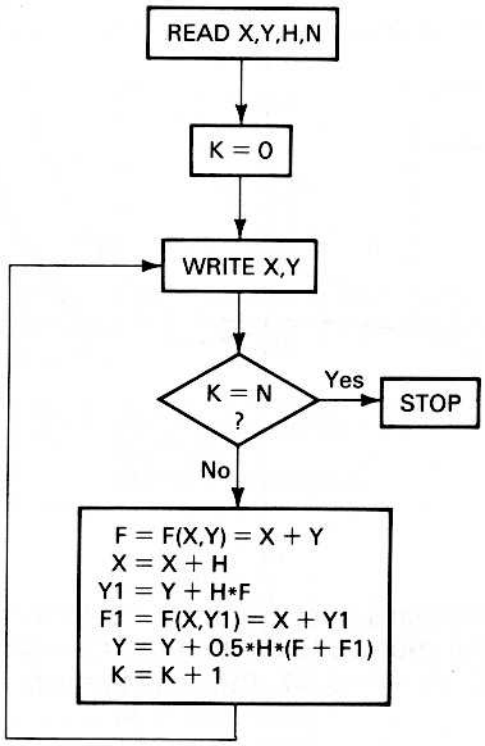
All of the methods looked at so far have been *single-step* methods; these are methods in which each step is like a separate initial-value problem. There is also a large class of *multistep* methods in which the history of the solution preceding the current step is used. For instance, we might use the value of f at the current point and at, say, six preceding points; then we might fit a polynomial in h through these points and this polynomial would provide the integrand in (3.1.10). This procedure would make what is called a *prediction* of the value of y at the end of the step. Once that predicted value is known, it can be incorporated into the polynomial, and this new polynomial can be used in the integral to get a *correction*. These methods employ fewer function evaluations than do Runge-Kutta methods (and the function evaluations may be the most costly part of the program) and can be designed to a high order. They are especially efficient in cases where the step size need not be changed. For instance they are used in orbital calculations for spacecraft. But they are tricky to set up and take some experience to control. Before a multistep method can begin, a set of values of y must already be known: these are usually provided by a Runge-Kutta integration. So I do not recommend multistep methods at this stage.

3.2 AN APPLICATION OF THE IMPROVED EULER'S METHOD

If you are in a hurry to reach the principal method of the text, then skip this section. But if you would like to have a go at some of the projects set out in Chapter 4 while learning that method, you may find this discussion helpful.

Having written a program for applying Euler's method, you need to add only two more instructions for the improved method. This is illustrated in the flowchart below. (Compare this flowchart to the one in Section 2.1 (p. 17).)

Your first job is to change your old program. Since the improved Euler's



method is of second order, repeat the project that numerically confirmed the order of Euler's method. Your straight line should now have slope three.

Next, compare results from the old Euler's method with those from the improved method. Such a comparison is shown in Figures 3.1 and 3.2. The aim here is to show the advantage of the improved method and also the accelerated advantage when the step size is reduced. The good effect when the step size is halved is obvious for the improved method; but Euler's method remains a disaster in each case.

Pictorial representation on the screen or in a diagram is all very well; but you should be starting to require results with much higher accuracy. Print out your results, and don't be satisfied unless several decimal places are correct.

The kinds of applications for which this method could be suitable are projects that lead to two simple simultaneous d.e.'s, e.g., some predator–prey models, the spread of disease, or the war models. I do not want to introduce subscripted variables at this stage, although you are welcome to do so if you prefer. Consider the simultaneous equations

$$\left.\begin{aligned} \frac{dy}{dx} &= f(x,y,z), \\ \frac{dz}{dx} &= g(x,y,z), \\ y(x_0) &= y_0, \quad z(x_0) = z_0. \end{aligned}\right] \tag{3.2.1}$$

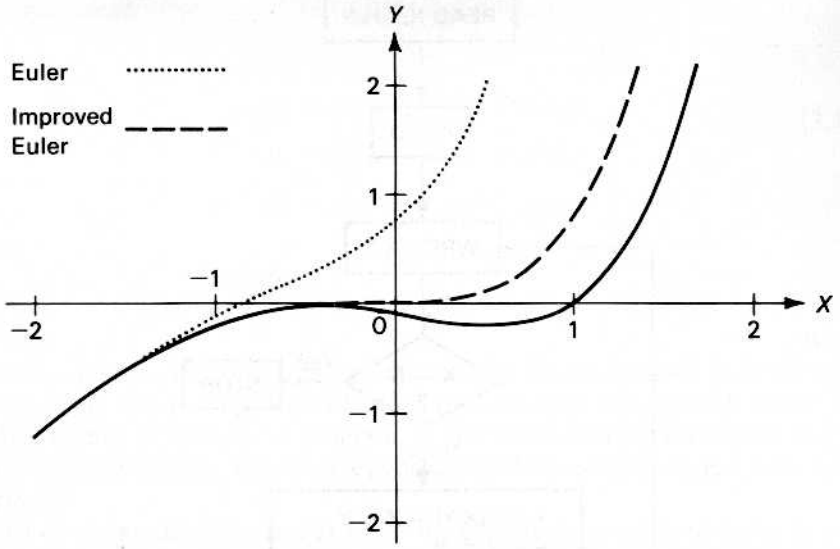


Figure 3.1 Comparison of the use of Euler's method and the improved Euler's method. $y = 2y + y^2$, $y(-2) = -1.247$. The step size is $h = 0.1$.

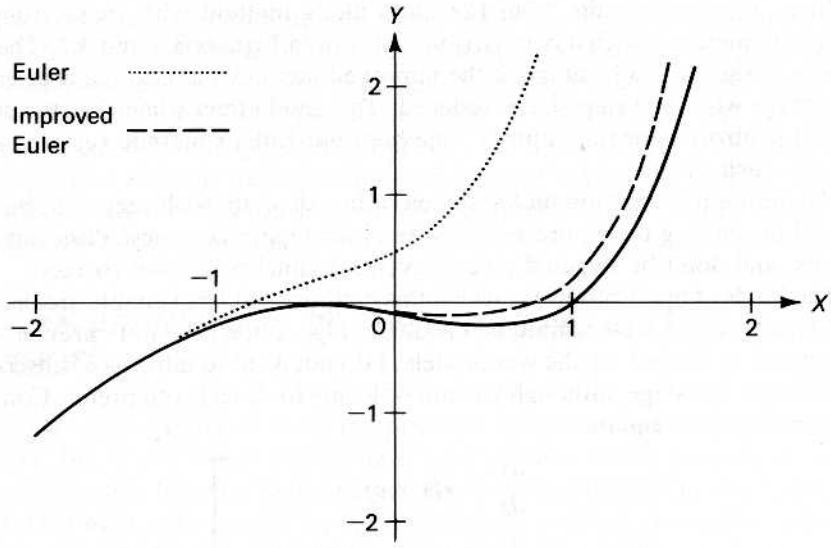
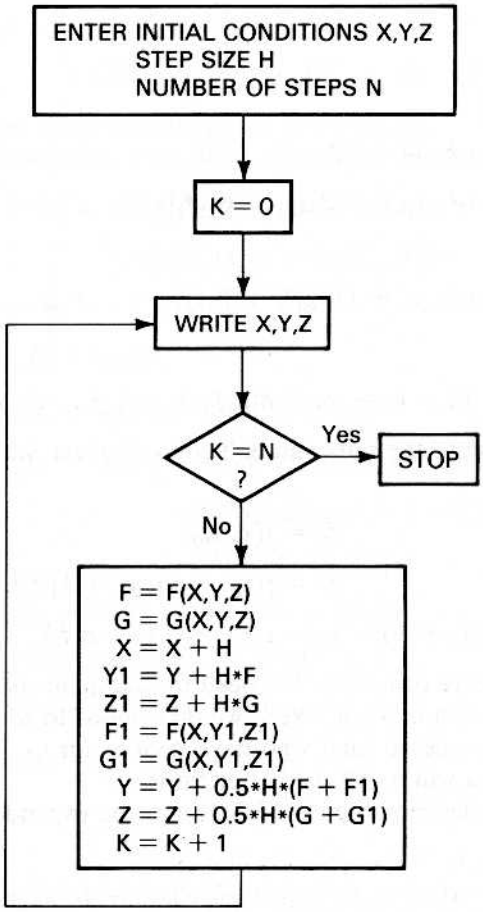


Figure 3.2 Same as Figure 3.1, with step size $h = 0.05$.

In the case of the predator–prey model, these would become

$$\left.\begin{aligned} \frac{dy}{dx} &= ay - byz, \\ \frac{dz}{dx} &= -cz + dyz, \\ y(x_0) &= y_0, \quad z(x_0) = z_0. \end{aligned}\right] \tag{3.2.2}$$

x would represent the time, y the prey population, and z the predator population; a, b, c, and d are constants of the model. Here is a flowchart for a possible treatment of equations (3.2.1).



There are, of course, other ways of terminating a program. For instance, you might enter at the start the greatest value of x for which you want output. If this value is XMAX, then you stop the calculation when X ≥ XMAX. Exper-

iment with different values of the step size; if the choice of h worries you, that is all to the good since you will be in a better frame of mind for what is to come.

The improved Euler's method goes well on a hand calculator. It is useful if you realize its limitations.

3.3 RUNGE-KUTTA FORMULAS

The general explicit Runge-Kutta step can be written as follows. For

$$\frac{dy}{dx} = f(x,y); \quad y(x_0) = y_0.$$

Let

$$\begin{aligned} f_0 &= f(x_0,y_0), \\ f_1 &= f(x_0 + \alpha_1 h,\ y_0 + h\beta_{10} f_0) \\ f_2 &= f(x_0 + \alpha_2 h,\ y_0 + h\{\beta_{20} f_0 + \beta_{21} f_1\}), \\ &\vdots \\ f_k &= f(x_0 + \alpha_k h,\ y_0 + h\{\beta_{k0} f_0 + \beta_{k1} f_1 + \ldots + \beta_{k,k-1} f_{k-1}\}). \end{aligned} \tag{3.3.1}$$

Then

$$y(x_0 + h) \simeq y_1 = y_0 + h(c_0 f_0 + c_1 f_1 + \ldots + c_k f_k).$$

To illustrate some of the nuts and bolts of a formula, let's look at the case $k = 2$:

$$\begin{aligned} f_0 &= f(x_0,y_0), \\ f_1 &= f(x_0 + \alpha h,\ y_0 + h\beta f_0). \\ y(x_0 + h) \simeq y_1 &= y_0 + h(c_0 f_0 + c_1 f_1). \end{aligned} \tag{3.3.2}$$

The α's, β's and c's are constants. We need to find them such that the error in y_1 is as small as we can make it. Well, we don't need to really, because there are heroes of a very special kind who have done it for us. But just relax and watch it happen; you will be all the better for it.

Both sides of the last equation of (3.3.2) must be expanded in powers of h. For a start,

$$y(x_0 + h) = y(x_0) + hy'(x_0) + \tfrac{1}{2}h^2 y''(x_0) + \tfrac{1}{6}h^3 y'''(x_0) + \ldots \tag{3.3.3}$$

Now

$$y(x_0) = y_0, \text{ and } y'(x_0) = f(x_0,y_0) = f_0.$$