CHAPTER 21

Numerics for ODEs and PDEs

Ordinary differential equations (ODEs) and partial differential equations (PDEs) play a central role in modeling problems of engineering, mathematics, physics, aeronautics, astronomy, dynamics, elasticity, biology, medicine, chemistry, environmental science, economics, and many other areas. Chapters 1–6 and 12 explained the major approaches to solving ODEs and PDEs analytically. However, in your career as an engineer, applied mathematicians, or physicist you will encounter ODEs and PDEs that *cannot* be solved by those analytic methods or whose solutions are so difficult that other approaches are needed. It is precisely in these real-world projects that numeric methods for ODEs and PDEs are used, often as part of a software package. Indeed, numeric software has become an indispensable tool for the engineer.

This chapter is evenly divided between numerics for ODEs and numerics for PDEs. We start with ODEs and discuss, in Sec. 21.1, methods for first-order ODEs. The main initial idea is that we can obtain approximations to the solution of such an ODE at points that are a distance h apart by using the first two terms of Taylor's formula from calculus. We use these approximations to construct the iteration formula for a method known as Euler's method. While this method is rather unstable and of little practical use, it serves as a pedagogical tool and a starting point toward understanding more sophisticated methods such as the Runge–Kutta method and its variant the Runga–Kutta–Fehlberg (RKF) method, which are popular and useful in practice. As is usual in mathematics, one tends to generalize mathematical ideas. The methods of Sec. 21.1 are one-step methods, that is, the current approximation uses only the approximation from the previous step. Multistep methods, such as the Adams–Bashforth methods and Adams–Moulton methods, use values computed from several previous steps. We conclude numerics for ODEs with applying Runge–Kutta–Nyström methods and other methods to higher order ODEs and systems of ODEs.

Numerics for PDEs are perhaps even more exciting and ingenious than those for ODEs. We first consider PDEs of the elliptic type (Laplace, Poisson). Again, Taylor's formula serves as a starting point and lets us replace partial derivatives by difference quotients. The end result leads to a mesh and an evaluation scheme that uses the Gauss–Seidel method (here also know as Liebmann's method). We continue with methods that use grids to solve Neuman and mixed problems (Sec. 21.5) and conclude with the important Crank–Nicholson method for parabolic PDEs in Sec. 21.6.

***Sections* 21.1 *and* 21.2 *may be studied immediately after Chap.* 1 *and Sec.* 21.3 *immediately after Chaps.* 2–4,** because these sections are independent of Chaps. 19 and 20.

***Sections* 21.4–21.7 *on PDEs may be studied immediately after Chap.* 12** if students have some knowledge of linear systems of algebraic equations.

Prerequisite: Secs. 1.1–1.5 for ODEs, Secs. 12.1–12.3, 12.5, 12.10 for PDEs.
References and Answers to Problems: App. 1 Part E (see also Parts A and C), App. 2.

21.1 Methods for First-Order ODEs

Take a look at Sec. 1.2, where we briefly introduced Euler's method with an example. *We shall develop* ***Euler's method*** *more rigorously.* Pay close attention to the derivation that uses Taylor's formula from calculus to approximate the solution to a first-order ODE at points that are a distance h apart. If you understand this approach, which is typical for numerics for ODEs, then you will understand other methods more easily.

From Chap. 1 we know that an ODE of the first order is of the form $F(x, y, y') = 0$ and can often be written in the explicit form $y' = f(x, y)$. An **initial value problem** for this equation is of the form

$$y' = f(x, y), \qquad y(x_0) = y_0 \tag{1}$$

where x_0 and y_0 are given and we assume that the problem has a unique solution on some open interval $a < x < b$ containing x_0.

In this section we shall discuss methods of computing approximate numeric values of the solution $y(x)$ of (1) at the equidistant points on the x-axis

$$x_1 = x_0 + h, \qquad x_2 = x_0 + 2h, \qquad x_3 = x_0 + 3h, \qquad \cdots$$

where the **step size** h is a fixed number, for instance, 0.2 or 0.1 or 0.01, whose choice we discuss later in this section. Those methods are **step-by-step methods**, using the same formula in each step. Such formulas are suggested by the Taylor series

$$y(x + h) = y(x) + hy'(x) + \frac{h^2}{2}y''(x) + \cdots. \tag{2}$$

Formula (2) is the key idea that lets us develop Euler's method and its variant called—you guessed it—*improved Euler method*, also known as *Heun's method*. Let us start by deriving Euler's method.

For small h the higher powers $h^2, h^3, \cdots$ in (2) are very small. Dropping all of them gives the crude approximation

$$\begin{aligned} y(x + h) &\approx y(x) + hy'(x) \\ &= y(x) + hf(x, y) \end{aligned}$$

and the corresponding **Euler method** (or **Euler–Cauchy method)**

$$y_{n+1} = y_n + hf(x_n, y_n) \qquad (n = 0, 1, \cdots) \tag{3}$$

discussed in Sec. 1.2. Geometrically, this is an approximation of the curve of $y(x)$ by a polygon whose first side is tangent to this curve at x_0 (see Fig. 8 in Sec. 1.2).

Error of the Euler Method. Recall from calculus that Taylor's formula with remainder has the form

$$y(x + h) = y(x) + hy'(x) + \tfrac{1}{2}h^2y''(\xi)$$

(where $x \leqq \xi \leqq x + h$). It shows that, in the Euler method, the *truncation error in each step* or **local truncation error** is proportional to h^2, written $O(h^2)$, where O suggests *order* (see also Sec. 20.1). Now, over a fixed x-interval in which we want to solve an ODE, the number of steps is proportional to $1/h$. Hence the *total error* or **global error** is proportional to $h^2(1/h) = h^1$. For this reason, the Euler method is called a **first-order method**. In addition, there are **roundoff errors** in this and other methods, which may affect the accuracy of the values $y_1, y_2, \cdots$ more and more as n increases.

Automatic Variable Step Size Selection in Modern Software. The idea of adaptive integration, as motivated and explained in Sec. 19.5, applies equally well to the numeric solution of ODEs. It now concerns automatically changing the step size h depending on the variability of $y' = f$ determined by

$$y'' = f' = f_x + f_y y' = f_x + f_y f. \tag{4*}$$

Accordingly, modern software automatically selects variable step sizes h_n so that the error of the solution will not exceed a given maximum size TOL (suggesting *tolerance*). Now for the Euler method, when the step size is $h = h_n$, the local error at x_n is about $\frac{1}{2}h_n^2|y''(\xi_n)|$. We require that this be equal to a given tolerance TOL,

$$\text{(a)}\quad \tfrac{1}{2}h_n^2|y''(\xi_n)| = \text{TOL}, \qquad \text{thus} \qquad \text{(b)}\quad h_n = \sqrt{\frac{2\,\text{TOL}}{|y''(\xi_n)|}}. \tag{4}$$

$y''(x)$ must not be zero on the interval J: $x_0 \leqq x = x_N$ on which the solution is wanted. Let K be the minimum of $|y''(x)|$ on J and assume that $K > 0$. Minimum $|y''(x)|$ corresponds to maximum $h = H = \sqrt{2\,\text{TOL}/K}$ by (4). Thus, $\sqrt{2\,\text{TOL}} = H\sqrt{K}$. We can insert this into (4b), obtaining by straightforward algebra

$$h_n = \varphi(x_n)H \qquad \text{where} \qquad \varphi(x_n) = \sqrt{\frac{K}{|y''(\xi_n)|}}. \tag{5}$$

For other methods, automatic step size selection is based on the same principle.

Improved Euler Method. Predictor, Corrector. Euler's method is generally much too inaccurate. For a large h (0.2) this is illustrated in Sec. 1.2 by the computation for

$$y' = y + x, \qquad y(0) = 0. \tag{6}$$

And for small h the computation becomes prohibitive; also, roundoff in so many steps may result in meaningless results. Clearly, methods of higher order and precision are obtained by taking more terms in (2) into account. But this involves an important practical problem. Namely, if we substitute $y' = f(x, y(x))$ into (2), we have

$$y(x + h) = y(x) + hf + \tfrac{1}{2}h^2 f' + \tfrac{1}{6}h^3 f'' + \cdots. \tag{2*}$$

Now y in f depends on x, so that we have f' as shown in (4*) and f'', f''' even much more cumbersome. The ***general strategy*** now is to avoid the computation of these derivatives and to replace it by computing f for one or several suitably chosen auxiliary values of (x, y). "Suitably" means that these values are chosen to make the order of the method as

high as possible (to have high accuracy). Let us discuss two such methods that are of practical importance, namely, the improved Euler method and the (classical) Runge–Kutta method.

In each step of the **improved Euler method** we compute *two* values, first the ***predictor***

(7a) $$y_{n+1}^* = y_n + hf(x_n, y_n),$$

which is an auxiliary value, and then the new y-value, the **corrector**

(7b) $$y_{n+1} = y_n + \tfrac{1}{2}h\,[f(x_n, y_n) + f(x_{n+1}, y_{n+1}^*)].$$

Hence the improved Euler method is a predictor–corrector method: In each step we predict a value (7a) and then we correct it by (7b).

In algorithmic form, using the notations $k_1 = hf(x_n, y_n)$ in (7a) and $k_2 = hf(x_{n+1}, y_{n+1}^*)$ in (7b), we can write this method as shown in Table 21.1.

Table 21.1 Improved Euler Method (Heun's Method)

ALGORITHM EULER (f, x_0, y_0, h, N)

This algorithm computes the solution of the initial value problem $y' = f(x, y)$, $y(x_0) = y_0$ at equidistant points $x_1 = x_0 + h, x_2 = x_0 + 2h, \cdots, x_N = x_0 + Nh$; here f is such that this problem has a unique solution on the interval $[x_0, x_N]$ (see Sec. 1.6).

INPUT: Initial values x_0, y_0, step size h, number of steps N

OUTPUT: Approximation y_{n+1} to the solution $y(x_{n+1})$ at $x_{n+1} = x_0 + (n + 1)h$, where $n = 0, \cdots, N - 1$

For $n = 0, 1, \cdots, N - 1$ do:

$x_{n+1} = x_n + h$

$k_1 = hf(x_n, y_n)$

$k_2 = hf(x_{n+1}, y_n + k_1)$

$y_{n+1} = y_n + \frac{1}{2}(k_1 + k_2)$

OUTPUT x_{n+1}, y_{n+1}

End

Stop

End EULER

EXAMPLE 1 Improved Euler Method. Comparison with Euler Method.

Apply the improved Euler method to the initial value problem (6), choosing $h = 0.2$ as in Sec. 1.2.

Solution. For the present problem we have in Table 21.1

$$k_1 = 0.2(x_n + y_n)$$

$$k_2 = 0.2(x_n + 0.2 + y_n + 0.2(x_n + y_n))$$

$$y_{n+1} = y_n + \frac{0.2}{2}(2.2x_n + 2.2y_n + 0.2) = y_n + 0.22(x_n + y_n) + 0.02.$$

Table 21.2 shows that our present results are much more accurate than those for Euler's method in Table 21.1 but at the cost of more computations. ■

Table 21.2 Improved Euler Method for (6). Errors

n	x_n	y_n	Exact Values (4D)	Error of Improved Euler	Error of Euler
0	0.0	0.0000	0.0000	0.0000	0.000
1	0.2	0.0200	0.0214	0.0014	0.021
2	0.4	0.0884	0.0918	0.0034	0.052
3	0.6	0.2158	0.2221	0.0063	0.094
4	0.8	0.4153	0.4255	0.0102	0.152
5	1.0	0.7027	0.7183	0.0156	0.230

Error of the Improved Euler Method. *The local error is of order h^3 and the global error of order h^2, so that the method is a* **second-order method**.

PROOF Setting $\tilde{f}_n = f(x_n, y(x_n))$ and using (2*) (after (6)), we have

$$y(x_n + h) - y(x_n) = h\tilde{f}_n + \tfrac{1}{2}h^2\tilde{f}_n' + \tfrac{1}{6}h^3\tilde{f}_n'' + \cdots. \tag{8a}$$

Approximating the expression in the brackets in (7b) by $\tilde{f}_n + \tilde{f}_{n+1}$ and again using the Taylor expansion, we obtain from (7b)

$$\begin{aligned} y_{n+1} - y_n &\approx \tfrac{1}{2}h\,[\tilde{f}_n + \tilde{f}_{n+1}] \\ &= \tfrac{1}{2}h\,[\tilde{f}_n + (\tilde{f}_n + h\tilde{f}_n' + \tfrac{1}{2}h^2\tilde{f}_n'' + \cdots)] \\ &= h\tilde{f}_n + \tfrac{1}{2}h^2\tilde{f}_n' + \tfrac{1}{4}h^3\tilde{f}_n'' + \cdots \end{aligned} \tag{8b}$$

(where $' = d/dx_n$, etc.). Subtraction of (8b) from (8a) gives the local error

$$\frac{h^3}{6}\tilde{f}_n'' - \frac{h^3}{4}\tilde{f}_n'' + \cdots = -\frac{h^3}{12}\tilde{f}_n'' + \cdots.$$

Since the number of steps over a fixed x-interval is proportional to $1/h$, the global error is of order $h^3/h = h^2$, so that the method is of second order. ■

Since the Euler method was an attractive pedagogical tool to teach the beginning of solving first-order ODEs numerically but had its drawbacks in terms of accuracy and could even produce wrong answers, we studied the improved Euler method and thereby introduced the idea of a predictor–corrector method. Although improved Euler is better than Euler, there are better methods that are used in industrial settings. Thus the practicing engineer has to know about the Runga–Kutta methods and its variants.

Runge–Kutta Methods (RK Methods)

A method of great practical importance and much greater accuracy than that of the improved Euler method is the *classical Runge–Kutta method of fourth order*, which we

call briefly the **Runge–Kutta method**.[1] It is shown in Table 21.3. We see that in each step we first compute four auxiliary quantities k_1, k_2, k_3, k_4 and then the new value y_{n+1}. The method is well suited to the computer because it needs no special starting procedure, makes light demand on storage, and repeatedly uses the same straightforward computational procedure. It is numerically stable.

Note that, if f depends only on x, this method reduces to Simpson's rule of integration (Sec. 19.5). Note further that $k_1, \cdots, k_4$ depend on n and generally change from step to step.

Table 21.3 Classical Runge–Kutta Method of Fourth Order

ALGORITHM RUNGE–KUTTA (f, x_0, y_0, h, N).

This algorithm computes the solution of the initial value problem $y' = f(x, y)$, $y(x_0) = y_0$ at equidistant points

$$x_1 = x_0 + h, x_2 = x_0 + 2h, \cdots, x_N = x_0 + Nh; \tag{9}$$

here f is such that this problem has a unique solution on the interval $[x_0, x_N]$ (see Sec. 1.7).

INPUT: Function f, initial values x_0, y_0, step size h, number of steps N

OUTPUT: Approximation y_{n+1} to the solution $y(x_{n+1})$ at $x_{n+1} = x_0 + (n + 1)h$, where $n = 0, 1, \cdots, N - 1$

For $n = 0, 1, \cdots, N - 1$ do:

$$k_1 = hf(x_n, y_n)$$
$$k_2 = hf(x_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}k_1)$$
$$k_3 = hf(x_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}k_2)$$
$$k_4 = hf(x_n + h, y_n + k_3)$$
$$x_{n+1} = x_n + h$$
$$y_{n+1} = y_n + \tfrac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$
OUTPUT x_{n+1}, y_{n+1}

End

Stop

End RUNGE–KUTTA

[1] Named after the German mathematicians KARL RUNGE (Sec. 19.4) and WILHELM KUTTA (1867–1944). Runge [*Math. Annalen* **46** (1895), 167–178], the German mathematician KARL HEUN (1859–1929) [*Zeitschr. Math. Phys.* **45** (1900), 23–38], and Kutta [*Zeitschr. Math. Phys.* **46** (1901), 435–453] developed various similar methods. Theoretically, there are infinitely many fourth-order methods using four function values per step. The method in Table 21.3 is most popular from a practical viewpoint because of its "symmetrical" form and its simple coefficients. It was given by Kutta.

EXAMPLE 2 Classical Runge–Kutta Method

Apply the Runge–Kutta method to the initial value problem in Example 1, choosing $h = 0.2$, as before, and computing five steps.

Solution. For the present problem we have $f(x, y) = x + y$. Hence

$$k_1 = 0.2(x_n + y_n), \qquad k_2 = 0.2(x_n + 0.1 + y_n + 0.5k_1),$$
$$k_3 = 0.2(x_n + 0.1 + y_n + 0.5k_2), \qquad k_4 = 0.2(x_n + 0.2 + y_n + k_3).$$

Table 21.4 shows the results and their errors, which are smaller by factors 10^3 and 10^4 than those for the two Euler methods. See also Table 21.5. We mention in passing that since the present $k_1, \cdots, k_4$ are simple, operations were saved by substituting k_1 into k_2, then k_2 into k_3, etc.; the resulting formula is shown in Column 4 of Table 21.4. Keep in mind that we have four function evaluations at each step. ■

Table 21.4 Runge–Kutta Method Applied to (4)

n	x_n	y_n	$0.2214(x_n + y_n) + 0.0214$	Exact Values (6D) $y = e^x - x - 1$	$10^6 \times$ Error of y_n
0	0.0	0	0.021400	0.000000	0
1	0.2	0.021400	0.070418	0.021403	3
2	0.4	0.091818	0.130289	0.091825	7
3	0.6	0.222107	0.203414	0.222119	12
4	0.8	0.425521	0.292730	0.425541	20
5	1.0	0.718251		0.718282	31

Table 21.5 Comparison of the Accuracy of the Three Methods under Consideration in the Case of the Initial Value Problem (4), with $h = 0.2$

x	$y = e^x - x - 1$	Error		
		Euler (Table 21.1)	Improved Euler (Table 21.3)	Runge–Kutta (Table 21.5)
0.2	0.021403	0.021	0.0014	0.000003
0.4	0.091825	0.052	0.0034	0.000007
0.6	0.222119	0.094	0.0063	0.000011
0.8	0.425541	0.152	0.0102	0.000020
1.0	0.718282	0.230	0.0156	0.000031

Error and Step Size Control. RKF (Runge–Kutta–Fehlberg)

The idea of adaptive integration (Sec. 19.5) has analogs for Runge–Kutta (and other) methods. In Table 21.3 for RK (Runge–Kutta), if we compute in each step approximations $\tilde{y}$ and $\tilde{\tilde{y}}$ with step sizes h and $2h$, respectively, the latter has error per step equal to $2^5 = 32$ times that of the former; however, since we have only half as many steps for $2h$, the actual factor is $2^5/2 = 16$, so that, say,

$$\epsilon^{(2h)} \approx 16\epsilon^{(h)} \qquad \text{and thus} \qquad y^{(h)} - y^{(2h)} = \epsilon^{(2h)} - \epsilon^{(h)} \approx (16 - 1)\epsilon^{(h)}.$$

Hence the error $\epsilon = \epsilon^{(h)}$ for step size h is about

(10) $$\epsilon = \tfrac{1}{15}(\tilde{y} - \tilde{\tilde{y}})$$

where $\tilde{y} - \tilde{\tilde{y}} = y^{(h)} - y^{(2h)}$, as said before. Table 21.6 illustrates (10) for the initial value problem

(11) $$y' = (y - x - 1)^2 + 2, \qquad y(0) = 1,$$

the step size $h = 0.1$ and $0 \leqq x \leqq 0.4$. We see that the estimate is close to the actual error. This method of error estimation is simple but may be unstable.

Table 21.6 Runge–Kutta Method Applied to the Initial Value Problem (11) and Error Estimate (10). Exact Solution $y = \tan x + x + 1$

x	$\tilde{y}$ (Step size h)	$\tilde{\tilde{y}}$ (Step size $2h$)	Error Estimate (10)	Actual Error	Exact Solution (9D)
0.0	1.000000000	1.000000000	0.000000000	0.000000000	1.000000000
0.1	1.200334589			0.000000083	1.200334672
0.2	1.402709878	1.402707408	0.000000165	0.000000157	1.402710036
0.3	1.609336039			0.000000210	1.609336250
0.4	1.822792993	1.822788993	0.000000267	0.000000226	1.822793219

RKF. E. Fehlberg [*Computing* **6** (1970), 61–71] proposed and developed error control by using two RK methods of different orders to go from (x_n, y_n) to (x_{n+1}, y_{n+1}). The difference of the computed y-values at x_{n+1} gives an error estimate to be used for step size control. Fehlberg discovered two RK formulas that together need only six function evaluations per step. We present these formulas here because RKF has become quite popular. For instance, Maple uses it (also for systems of ODEs).

Fehlberg's fifth-order RK method is

(12a) $$y_{n+1} = y_n + \gamma_1 k_1 + \cdots + \gamma_6 k_6$$

with coefficient vector $\gamma = [\gamma_1 \cdots \gamma_6]$,

(12b) $$\gamma = [\tfrac{16}{135} \quad 0 \quad \tfrac{6656}{12{,}825} \quad \tfrac{28{,}561}{56{,}430} \quad -\tfrac{9}{50} \quad \tfrac{2}{55}].$$

His **fourth-order RK method** is

(13a) $$y^*_{n+1} = y_n + \gamma^*_1 k_1 + \cdots + \gamma^*_5 k_5$$

with coefficient vector

(13b) $$\gamma^* = [\tfrac{25}{216} \quad 0 \quad \tfrac{1408}{2565} \quad \tfrac{2197}{4104} \quad -\tfrac{1}{5}].$$

In both formulas we use only six different function evaluations altogether, namely,

$$
\begin{aligned}
k_1 &= hf(x_n, y_n) \\
k_2 &= hf(x_n + \tfrac{1}{4}h, \quad y_n + \quad \tfrac{1}{4}k_1) \\
k_3 &= hf(x_n + \tfrac{3}{8}h, \quad y_n + \quad \tfrac{3}{32}k_1 + \quad \tfrac{9}{32}k_2) \\
k_4 &= hf(x_n + \tfrac{12}{13}h, \quad y_n + \tfrac{1932}{2197}k_1 - \tfrac{7200}{2197}k_2 + \tfrac{7296}{2197}k_3) \\
k_5 &= hf(x_n + h, \quad y_n + \tfrac{439}{216}k_1 - \quad 8k_2 + \tfrac{3680}{513}k_3 - \tfrac{845}{4104}k_4) \\
k_6 &= hf(x_n + \tfrac{1}{2}h, \quad y_n - \quad \tfrac{8}{27}k_1 + \quad 2k_2 - \tfrac{3544}{2565}k_3 + \tfrac{1859}{4104}k_4 - \tfrac{11}{40}k_5).
\end{aligned} \tag{14}
$$

The difference of (12) and (13) gives the **error estimate**

$$
\epsilon_{n+1} \approx y_{n+1} - y_{n+1}^* = \tfrac{1}{360}k_1 - \tfrac{128}{4275}k_3 - \tfrac{2197}{75{,}240}k_4 + \tfrac{1}{50}k_5 + \tfrac{2}{55}k_6. \tag{15}
$$

EXAMPLE 3 Runge–Kutta–Fehlberg

For the initial value problem (11) we obtain from (12)–(14) with $h = 0.1$ in the first step the 12S-values

$$
\begin{aligned}
k_1 &= 0.200000000000 & k_2 &= 0.200062500000 \\
k_3 &= 0.200140756867 & k_4 &= 0.200856926154 \\
k_5 &= 0.201006676700 & k_6 &= 0.200250418651
\end{aligned}
$$

$$
\begin{aligned}
y_1^* &= 1.20033466949 \\
y_1 &= 1.20033467253
\end{aligned}
$$

and the error estimate

$$
\epsilon_1 \approx y_1 - y_1^* = 0.00000000304.
$$

The exact 12S-value is $y(0.1) = 1.20033467209$. Hence the actual error of y_1 is $-4.4 \cdot 10^{-10}$, smaller than that in Table 21.6 by a factor of 200. ■

Table 21.7 summarizes essential features of the methods in this section. It can be shown that these methods are *numerically stable* (definition in Sec. 19.1). They are **one-step methods** because in each step we use the data of just *one* preceding step, in contrast to **multistep methods** where in each step we use data from *several* preceding steps, as we shall see in the next section.

Table 21.7 Methods Considered and Their Order (= Their Global Error)

Method	Function Evaluation per Step	Global Error	Local Error
Euler	1	$O(h)$	$O(h^2)$
Improved Euler	2	$O(h^2)$	$O(h^3)$
RK (fourth order)	4	$O(h^4)$	$O(h^5)$
RKF	6	$O(h^5)$	$O(h^6)$

Backward Euler Method. Stiff ODEs

The **backward Euler formula** for numerically solving (1) is

$$\textbf{(16)} \qquad y_{n+1} = y_n + hf(x_{n+1}, y_{n+1}) \qquad (n = 0, 1, \cdots).$$

This formula is obtained by evaluating the right side at the *new* location (x_{n+1}, y_{n+1}); this is called the **backward Euler scheme**. For known y_n it gives y_{n+1} *implicitly,* so it defines an **implicit method**, in contrast to the Euler method (3), which gives y_{n+1} explicitly. Hence (16) must be solved for y_{n+1}. How difficult this is depends on f in (1). For a linear ODE this provides no problem, as Example 4 (below) illustrates. The method is particularly useful for "stiff" ODEs, as they occur quite frequently in the study of vibrations, electric circuits, chemical reactions, etc. The situation of stiffness is roughly as follows; for details, see, for example, [E5], [E25], [E26] in App. 1.

Error terms of the methods considered so far involve a higher derivative. And we ask what happens if we let h *increase.* Now if the error (the derivative) grows fast but the desired solution also grows fast, nothing will happen. However, if that solution does not grow fast, then with growing h the error term can take over to an extent that the numeric result becomes completely nonsensical, as in Fig. 451. Such an ODE for which h must thus be restricted to small values, and the physical system the ODE models, are called **stiff**. This term is suggested by a mass–spring system with a stiff spring (spring with a large k; see Sec. 2.4). Example 4 illustrates that implicit methods remove the difficulty of increasing h in the case of stiffness: It can be shown that in the application of an implicit method the solution remains stable under any increase of h, although the accuracy decreases with increasing h.

EXAMPLE 4 **Backward Euler Method. Stiff ODE**

The initial value problem

$$y' = f(x, y) = -20hy + 20x^2 + 2x, \quad y(0) = 1$$

has the solution (verify!)

$$y = e^{-20x} + x^2.$$

The backward Euler formula (16) is

$$y_{n+1} = y_n + hf(x_{n+1}, y_{n+1}) = y_n + h(-20y_{n+1} + 20x_{n+1}^2 + 2x_{n+1}).$$

Noting that $x_{n+1} = x_n + h$, taking the term $-20y_{n+1}$ to the left, and dividing, we obtain

$$(16^*) \qquad y_{n+1} = \frac{y_n + h[20(x_n + h)^2 + 2(x_n + h)]}{1 + 20h}.$$

The numeric results in Table 21.8 show the following.

Stability of the backward Euler method for $h = 0.05$ and also for $h = 0.2$ with an error increase by about a factor 4 for $h = 0.2$,

Stability of the Euler method for $h = 0.05$ but instability for $h = 0.1$ (Fig. 451),

Stability of RK for $h = 0.1$ but instability for $h = 0.2$.

This illustrates that the ODE is stiff. Note that even in the case of stability the approximation of the solution near $x = 0$ is poor. ■

Stiffness will be considered further in Sec. 21.3 in connection with systems of ODEs.

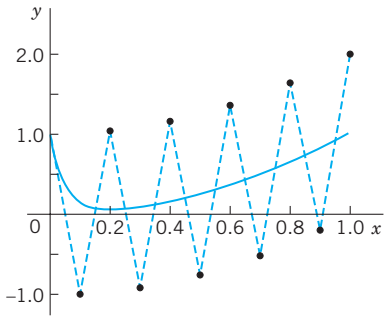

Fig. 451. Euler method with $h = 0.1$ for the stiff ODE in Example 4 and exact solution

Table 21.8 Backward Euler Method (BEM) for Example 6. Comparison with Euler and RK

x	BEM $h = 0.05$	BEM $h = 0.2$	Euler $h = 0.05$	Euler $h = 0.1$	RK $h = 0.1$	RK $h = 0.2$	Exact
0.0	1.00000	1.00000	1.00000	1.00000	1.00000	1.000	1.00000
0.1	0.26188		0.00750	−1.00000	0.34500		0.14534
0.2	0.10484	0.24800	0.03750	1.04000	0.15333	5.093	0.05832
0.3	0.10809		0.08750	−0.92000	0.12944		0.09248
0.4	0.16640	0.20960	0.15750	1.16000	0.17482	25.48	0.16034
0.5	0.25347		0.24750	−0.76000	0.25660		0.25004
0.6	0.36274	0.37792	0.35750	1.36000	0.36387	127.0	0.36001
0.7	0.49256		0.48750	−0.52000	0.49296		0.49001
0.8	0.64252	0.65158	0.63750	1.64000	0.64265	634.0	0.64000
0.9	0.81250		0.80750	−0.20000	0.81255		0.81000
1.0	1.00250	1.01032	0.99750	2.00000	1.00252	3168	1.00000

PROBLEM SET 21.1

1–4 EULER METHOD

Do 10 steps. Solve exactly. Compute the error. Show details.

1. $y' + 0.2y = 0, \quad y(0) = 5, \quad h = 0.2$

2. $y' = \frac{1}{2}\pi\sqrt{1 - y^2}, \quad y(0) = 0, \quad h = 0.1$

3. $y' = (y - x)^2, \quad y(0) = 0, \quad h = 0.1$

4. $y' = (y + x)^2, \quad y(0) = 0, \quad h = 0.1$

5–10 IMPROVED EULER METHOD

Do 10 steps. Solve exactly. Compute the error. Show details.

5. $y' = y, \quad y(0) = 1, \quad h = 0.1$

6. $y' = 2(1 + y^2), \quad y(0) = 0, \quad h = 0.05$

7. $y' - xy^2 = 0, \quad y(0) = 1, \quad h = 0.1$

8. Logistic population model. $y' = y - y^2, \quad y(0) = 0.2, \quad h = 0.1$

9. Do Prob. 7 using Euler's method with $h = 0.1$ and compare the accuracy.

10. Do Prob. 7 using the improved Euler method, 20 steps with $h = 0.05$. Compare.

11–17 CLASSICAL RUNGE–KUTTA METHOD OF FOURTH ORDER

Do 10 steps. Compare as indicated. Show details.

11. $y' - xy^2 = 0, \quad y(0) = 1, \quad h = 0.1$. Compare with Prob. 7. Apply the error estimate (10) to y_{10}.

12. $y' = y - y^2, \quad y(0) = 0.2, \quad h = 0.1$. Compare with Prob. 8.

13. $y' = 1 + y^2, \quad y(0) = 0, \quad h = 0.1$

14. $y' = (1 - x^{-1})y, \quad y(1) = 1, \quad h = 0.1$

15. $y' + y\tan x = \sin 2x, \quad y(0) = 1, \quad h = 0.1$

16. Do Prob. 15 with $h = 0.2$, 5 steps, and compare the errors with those in Prob. 15.

17. $y' = 4x^3y^2, \quad y(0) = 0.5, \quad h = 0.1$

18. Kutta's third-order method is defined by $y_{n+1} = y_n + \frac{1}{6}(k_1 + 4k_2 + k_3^*)$ with k_1 and k_2 as in RK (Table 21.3) and $k_3^* = hf(x_{n+1}, y_n - k_1 + 2k_2)$. Apply this method to (4) in (6). Choose $h = 0.2$ and do 5 steps. Compare with Table 21.5.

19. CAS EXPERIMENT. Euler–Cauchy vs. RK. Consider the initial value problem

(17) $y' = (y - 0.01x^2)^2 \sin(x^2) + 0.02x, \quad y(0) = 0.4$

(solution: $y = 1/[2.5 - \mathrm{S}(x)] + 0.01x^2$ where $\mathrm{S}(x)$ is the Fresnel integral (38) in App. 3.1).

(a) Solve (17) by Euler, improved Euler, and RK methods for $0 \leqq x \leqq 5$ with step $h = 0.2$. Compare the errors for $x = 1, 3, 5$ and comment.

(b) Graph solution curves of the ODE in (17) for various positive and negative initial values.

(c) Do a similar experiment as in (a) for an initial value problem that has a monotone increasing or monotone decreasing solution. Compare the behavior of the error with that in (a). Comment.

20. CAS EXPERIMENT. RKF. (a) Write a program for RKF that gives x_n, y_n, the estimate (10), and, if the solution is known, the actual error ϵ_n.

(b) Apply the program to Example 3 in the text (10 steps, $h = 0.1$).

(c) ϵ_n in (b) gives a relatively good idea of the size of the actual error. Is this typical or accidental? Find out, by experimentation with other problems, on what properties of the ODE or solution this might depend.

21.2 Multistep Methods

In a **one-step method** we compute y_{n+1} using only a single step, namely, the previous value y_n. *One-step methods are* "**self-starting**," they need no help to get going because they obtain y_1 from the initial value y_0, etc. All methods in Sec. 21.1 are one-step.

In contrast, a **multistep method** uses, in each step, values from two or more previous steps. These methods are motivated by the expectation that the additional information will increase accuracy and stability. But to get started, one needs values, say, y_0, y_1, y_2, y_3 in a 4-step method, obtained by Runge–Kutta or another accurate method. Thus, multistep methods are not self-starting. Such methods are obtained as follows.

Adams–Bashforth Methods

We consider an initial value problem

(1) $$y' = f(x, y), \qquad y(x_0) = y_0$$

as before, with f such that the problem has a unique solution on some open interval containing x_0. We integrate $y' = f(x, y)$ from x_n to $x_{n+1} = x_n + h$. This gives

$$\int_{x_n}^{x_{n+1}} y'(x)\,dx = y(x_{n+1}) - y(x_n) = \int_{x_n}^{x_{n+1}} f(x, y(x))\,dx.$$

Now comes the main idea. We replace $f(x, y(x))$ by an interpolation polynomial $p(x)$ (see Sec. 19.3), so that we can later integrate. This gives approximations y_{n+1} of $y(x_{n+1})$ and y_n of $y(x_n)$,

(2) $$y_{n+1} = y_n + \int_{x_n}^{x_{n+1}} p(x)\,dx.$$