3.7 HIGHER-ORDER NUMERICAL METHODS: TAYLOR AND RUNGE–KUTTA

In Sections 1.4 and 3.6, we discussed a simple numerical procedure, Euler's method, for obtaining a numerical approximation of the solution $\phi(x)$ to the initial value problem

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

Euler's method is easy to implement because it involves only linear approximations to the solution $\phi(x)$. But it suffers from slow convergence, being a method of order 1; that is, the error is $O(h)$. Even the improved Euler's method discussed in Section 3.6 has order of only 2. In this section we present numerical methods that have faster rates of convergence. These include **Taylor methods,** which are natural extensions of the Euler procedure, and **Runge–Kutta methods,** which are the more popular schemes for solving initial value problems because they have fast rates of convergence and are easy to program.

As in the previous section, we assume that f and $\partial f/\partial y$ are continuous and bounded on the vertical strip $\{(x, y)\colon a < x < b, -\infty < y < \infty\}$ and that f possesses as many continuous partial derivatives as needed.

To derive the Taylor methods, let $\phi_n(x)$ be the *exact* solution of the related initial value problem

$$\text{(2)} \qquad \phi_n' = f(x, \phi_n) \ , \qquad \phi_n(x_n) = y_n \ .$$

The Taylor series for $\phi_n(x)$ about the point x_n is

$$\phi_n(x) = \phi_n(x_n) + h\phi_n'(x_n) + \frac{h^2}{2!}\phi_n''(x_n) + \cdots \ ,$$

where $h = x - x_n$. Since ϕ_n satisfies (2), we can write this series in the form

$$\text{(3)} \qquad \phi_n(x) = y_n + hf(x_n, y_n) + \frac{h^2}{2!}\phi_n''(x_n) + \cdots \ .$$

Observe that the recursive formula for y_{n+1} in Euler's method is obtained by truncating the Taylor series after the linear term. For a better approximation, we will use more terms in the Taylor series. This requires that we express the higher-order derivatives of the solution in terms of the function $f(x, y)$.

If y satisfies $y' = f(x, y)$, we can compute y'' by using the chain rule:

$$\text{(4)} \qquad \begin{aligned} y'' &= \frac{\partial f}{\partial x}(x, y) + \frac{\partial f}{\partial y}(x, y)\, y' \\ &= \frac{\partial f}{\partial x}(x, y) + \frac{\partial f}{\partial y}(x, y) f(x, y) \\ &=: f_2(x, y) \ . \end{aligned}$$

In a similar fashion, define $f_3, f_4, \ldots$, that correspond to the expressions $y'''(x)$, $y^{(4)}(x)$, etc. If we truncate the expansion in (3) after the h^p term, then, with the above notation, the recursive formulas for the **Taylor method of order *p*** are

$$\text{(5)} \qquad x_{n+1} = x_n + h \ ,$$

$$\text{(6)} \qquad y_{n+1} = y_n + hf(x_n, y_n) + \frac{h^2}{2!} f_2(x_n, y_n) + \cdots + \frac{h^p}{p!} f_p(x_n, y_n) \ .$$

As before, $y_n \approx \phi(x_n)$, where $\phi(x)$ is the solution to the initial value problem (1). It can be shown[†] that **the Taylor method of order p has the rate of convergence $O(h^p)$.**

Example 1 Determine the recursive formulas for the Taylor method of order 2 for the initial value problem

$$y' = \sin(xy) \ , \qquad y(0) = \pi \ . \tag{7}$$

Solution We must compute $f_2(x, y)$ as defined in (4). Since $f(x, y) = \sin(xy)$,

$$\frac{\partial f}{\partial x}(x, y) = y\cos(xy) \ , \qquad \frac{\partial f}{\partial y}(x, y) = x\cos(xy) \ .$$

Substituting into (4), we have

$$\begin{aligned} f_2(x, y) &= \frac{\partial f}{\partial x}(x, y) + \frac{\partial f}{\partial y}(x, y) f(x, y) \\ &= y\cos(xy) + x\cos(xy)\sin(xy) \\ &= y\cos(xy) + \frac{x}{2}\sin(2xy) \ , \end{aligned}$$

and the recursive formulas (5) and (6) become

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

$$y_{n+1} = y_n + h\sin(x_n y_n) + \frac{h^2}{2}\left[y_n \cos(x_n y_n) + \frac{x_n}{2}\sin(2x_n y_n) \right] \ ,$$

where $x_0 = 0$, $y_0 = \pi$ are the starting values. ◆

The convergence rate, $O(h^p)$, of the pth-order Taylor method raises an interesting question: If we could somehow let p go to infinity, would we obtain **exact** solutions for the interval $[x_0, x_0 + h]$? This possibility is explored in depth in Chapter 8. Of course, a practical difficulty in employing high-order Taylor methods is the tedious computation of the partial derivatives needed to determine f_p (typically these computations grow exponentially with p). One way to circumvent this difficulty is to use one of the **Runge–Kutta methods.**[††]

Observe that the general Taylor method has the form

$$y_{n+1} = y_n + hF(x_n, y_n; h) \ , \tag{8}$$

where the choice of F depends on p. In particular [compare (6)], for

$$p = 1 \ , \qquad F = T_1(x, y; h) := f(x, y) \ ,$$

$$p = 2 \ , \qquad F = T_2(x, y; h) := f(x, y) + \frac{h}{2}\left[\frac{\partial f}{\partial x}(x, y) + \frac{\partial f}{\partial y}(x, y) f(x, y) \right] \ . \tag{9}$$

The idea behind the Runge–Kutta method of order 2 is to choose F in (8) of the form

$$F = K_2(x, y; h) := f\big(x + \alpha h, y + \beta h f(x, y)\big) \ , \tag{10}$$

[†]See *Introduction to Numerical Analysis* by J. Stoer and R. Bulirsch (Springer-Verlag, New York, 2002).

[††]*Historical Footnote:* These methods were developed by C. Runge in 1895 and W. Kutta in 1901.

where the constants α, β are to be selected so that (8) has the rate of convergence $O(h^2)$. The advantage here is that K_2 is computed by two evaluations of the original function $f(x, y)$ and does not involve the derivatives of $f(x, y)$.

To ensure $O(h^2)$ convergence, we compare this new scheme with the Taylor method of order 2 and require

$$T_2(x, y; h) - K_2(x, y; h) = O(h^2) \ , \quad \text{as} \quad h \to 0 \ .$$

That is, we choose α, β so that the Taylor expansions for T_2 and K_2 agree through terms of order h. For (x, y) fixed, when we expand $K_2 = K_2(h)$ as given in (10) about $h = 0$, we find

$$\begin{aligned} K_2(h) &= K_2(0) + \frac{dK_2}{dh}(0)h + O(h^2) \\ &= f(x, y) + \left[\alpha \frac{\partial f}{\partial x}(x, y) + \beta \frac{\partial f}{\partial y}(x, y) f(x, y) \right] h + O(h^2) \ , \end{aligned} \tag{11}$$

where the expression in brackets for dK_2/dh, evaluated at $h = 0$, follows from the chain rule. Comparing (11) with (9), we see that for T_2 and K_2 to agree through terms of order h, we must have $\alpha = \beta = 1/2$. Thus,

$$K_2(x, y; h) = f\left(x + \frac{h}{2}, y + \frac{h}{2} f(x, y) \right) \ .$$

The Runge–Kutta method we have derived is called the **midpoint method** and it has the recursive formulas

$$x_{n+1} = x_n + h \ , \tag{12}$$

$$y_{n+1} = y_n + hf\left(x_n + \frac{h}{2}, \ y_n + \frac{h}{2} f(x_n, y_n) \right) \ . \tag{13}$$

By construction, the midpoint method has the same rate of convergence as the Taylor method of order 2; that is, $O(h^2)$. This is the same rate as the improved Euler's method.

In a similar fashion, one can work with the Taylor method of order 4 and, after some elaborate calculations, obtain the **classical fourth-order Runge–Kutta method.** The recursive formulas for this method are

$$\begin{aligned} x_{n+1} &= x_n + h \ , \\ y_{n+1} &= y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \ , \end{aligned} \tag{14}$$

where

$$\begin{aligned} k_1 &= hf(x_n, y_n) \ , \\ k_2 &= hf\left(x_n + \frac{h}{2}, \ y_n + \frac{k_1}{2} \right) \ , \\ k_3 &= hf\left(x_n + \frac{h}{2}, \ y_n + \frac{k_2}{2} \right) \ , \\ k_4 &= hf(x_n + h, \ y_n + k_3) \ . \end{aligned}$$

The classical fourth-order Runge–Kutta method is one of the more popular methods because its rate of convergence is $O(h^4)$ and it is easy to program. Typically, it produces very accurate approximations even when the number of iterations is reasonably small. However, as the number of iterations becomes large, other types of errors may creep in.

Program outlines for the fourth-order Runge–Kutta method are given below. Just as with the algorithms for the improved Euler's method, the first program (the Runge–Kutta subroutine) is useful for approximating the solution over an interval $[x_0, c]$ and takes the number of steps in the interval as input. As in Section 3.6, the number of steps N is related to the step size h and the interval $[x_0, c]$ by

$$Nh = c - x_0 \ .$$

The subroutine has the option to print out a table of values of x and y. The second algorithm (Runge–Kutta with tolerance) on page 136 is used to approximate, for a given tolerance, the solution at an inputted value $x = c$. This algorithm[†] automatically halves the step sizes successively until the two approximations $y(c; h)$ and $y(c; h/2)$ differ by less than the prescribed tolerance ε. For a stopping procedure based on the relative error, Step 6 of the algorithm should be replaced by

Step 6′ If $\left|\dfrac{y - z}{y}\right| < \varepsilon$, go to Step 10 .

CLASSICAL FOURTH-ORDER RUNGE–KUTTA SUBROUTINE

Purpose To approximate the solution to the initial value problem

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

for $x_0 \le x \le c$

INPUT x_0, y_0, c, N (number of steps), PRNTR (=1 to print a table)

Step 1 Set step size $h = (c - x_0)/N$, $x = x_0$, $y = y_0$

Step 2 For $i = 1$ to N, do Steps 3–5

Step 3 Set

$$k_1 = hf(x, y)$$

$$k_2 = hf\left(x + \frac{h}{2}, y + \frac{k_1}{2}\right)$$

$$k_3 = hf\left(x + \frac{h}{2}, y + \frac{k_2}{2}\right)$$

$$k_4 = hf(x + h, y + k_3)$$

Step 4 Set

$$x = x + h$$

$$y = y + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

Step 5 If PRNTR = 1, print x, y

[†]Note that the form of the algorithm on page 136 is the same as that for the improved Euler's method on page 128 except for Step 4, where the Runge–Kutta subroutine is called. More sophisticated stopping procedures are used in production-grade codes.

CLASSICAL FOURTH-ORDER RUNGE–KUTTA ALGORITHM WITH TOLERANCE

Purpose To approximate the solution to the initial value problem

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

at $x = c$, with tolerance ε

INPUT x_0, y_0, c, ε, M (maximum number of iterations)

Step 1 Set $z = y_0$, PRNTR $= 0$

Step 2 For $m = 0$ to M, do Steps 3–7 (or, to save time, start with $m > 0$)

Step 3 Set $N = 2^m$

Step 4 Call FOURTH-ORDER RUNGE–KUTTA SUBROUTINE

Step 5 Print h, y

Step 6 If $|z - y| < \varepsilon$, go to Step 10

Step 7 Set $z = y$

Step 8 Print "$\phi(c)$ is approximately"; y; "but may not be within the tolerance"; ε

Step 9 Go to Step 11

Step 10 Print "$\phi(c)$ is approximately"; y; "with tolerance"; ε

Step 11 STOP

OUTPUT Approximations of the solution to the initial value problem at $x = c$, using 2^m steps.

Example 2 Use the classical fourth-order Runge–Kutta algorithm to approximate the solution $\phi(x)$ of the initial value problem

$$y' = y \ , \qquad y(0) = 1 \ ,$$

at $x = 1$ with a tolerance of 0.001.

Solution The inputs are $x_0 = 0$, $y_0 = 1$, $c = 1$, $\varepsilon = 0.001$, and $M = 25$ (say). Since $f(x, y) = y$, the formulas in Step 3 of the subroutine become

$$k = hy \ , \qquad k_2 = h\left(y + \frac{k_1}{2}\right) \ , \qquad k_3 = h\left(y + \frac{k_2}{2}\right) \ , \qquad k_4 = h(y + k_3) \ .$$

The initial value for N in this algorithm is $N = 1$, so

$$h = (1 - 0)/1 = 1 \ .$$

Thus, in Step 3 of the subroutine, we compute

$$k_1 = (1)(1) = 1 \ , \qquad k_2 = (1)(1 + 0.5) = 1.5 \ ,$$
$$k_3 = (1)(1 + 0.75) = 1.75 \ , \qquad k_4 = (1)(1 + 1.75) = 2.75 \ ,$$

and, in Step 4 of the subroutine, we get for the first approximation

$$\begin{aligned} y &= y_0 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ &= 1 + \frac{1}{6}[1 + 2(1.5) + 2(1.75) + 2.75] \\ &= 2.70833 \ , \end{aligned}$$

where we have rounded to five decimal places. Because

$$|z - y| = |y_0 - y| = |1 - 2.70833| = 1.70833 > \varepsilon \ ,$$

we start over and reset $N = 2$, $h = 0.5$.

Doing Steps 3 and 4 for $i = 1$ and 2, we ultimately obtain (for $i = 2$) the approximation

$$y = 2.71735 \ .$$

Since $|z - y| = |2.70833 - 2.71735| = 0.00902 > \varepsilon$, we again start over and reset $N = 4$, $h = 0.25$. This leads to the approximation

$$y = 2.71821 \ ,$$

so that

$$|z - y| = |2.71735 - 2.71821| = 0.00086 \ ,$$

which is less than $\varepsilon = 0.001$. Hence $\phi(1) = e \approx 2.71821$. ◆

In Example 2 we were able to obtain a better approximation for $\phi(1) = e$ with $h = 0.25$ than we obtained in Section 3.6 using Euler's method with $h = 0.001$ (see Table 3.4, page 124) and roughly the same accuracy as we obtained in Section 3.6 using the improved Euler's method with $h = 0.01$ (see Table 3.5, page 127).

Example 3 Use the fourth-order Runge–Kutta subroutine to approximate the solution $\phi(x)$ of the initial value problem

$$\textbf{(15)} \qquad y' = y^2 \ , \qquad y(0) = 1 \ ,$$

on the interval $0 \le x \le 2$ using $N = 8$ steps (i.e., $h = 0.25$).

Solution Here the starting values are $x_0 = 0$ and $y_0 = 1$. Since $f(x, y) = y^2$, the formulas in Step 3 of the subroutine are

$$k_1 = hy^2 \ , \qquad k_2 = h\left(y + \frac{k_1}{2}\right)^2 \ ,$$

$$k_3 = h\left(y + \frac{k_2}{2}\right)^2 \ , \qquad k_4 = h(y + k_3)^2 \ .$$

From the output, we find

$$\begin{aligned}
x &= 0.25 \qquad y = 1.33322 \ , \\
x &= 0.50 \qquad y = 1.99884 \ , \\
x &= 0.75 \qquad y = 3.97238 \ , \\
x &= 1.00 \qquad y = 32.82820 \ , \\
x &= 1.25 \qquad y = 4.09664 \times 10^{11} \ , \\
x &= 1.50 \qquad y = \text{overflow} \ .
\end{aligned}$$

What happened? Fortunately, the equation in (15) is separable, and, solving for $\phi(x)$, we obtain $\phi(x) = (1 - x)^{-1}$. It is now obvious where the problem lies: The true solution $\phi(x)$ is not defined

at $x = 1$. If we had been more cautious, we would have realized that $\partial f/\partial y = 2y$ is *not* bounded for all y. Hence, the existence of a unique solution is not guaranteed for all x between 0 and 2, and in this case, the method does *not* give meaningful approximations for x near (or greater than) 1. ◆

Example 4 Use the fourth-order Runge–Kutta algorithm to approximate the solution $\phi(x)$ of the initial value problem

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

at $x = 2$ with a tolerance of 0.0001.

Solution This time we check to see whether $\partial f/\partial y$ is bounded. Here $\partial f/\partial y = -2y$, which is certainly unbounded in any vertical strip. However, let's consider the qualitative behavior of the solution $\phi(x)$. The solution curve starts at (0, 1), where $\phi'(0) = 0 - 1 < 0$, so $\phi(x)$ begins decreasing and continues to decrease until it crosses the curve $y = \sqrt{x}$. After crossing this curve, $\phi(x)$ begins to increase, since $\phi'(x) = x - \phi^2(x) > 0$. As $\phi(x)$ increases, it remains below the curve $y = \sqrt{x}$. This is because if the solution were to get "close" to the curve $y = \sqrt{x}$, then the derivative of $\phi(x)$ would approach zero, so that overtaking the function $\sqrt{x}$ is impossible.

Therefore, although the existence-uniqueness theorem does not guarantee a solution, we are inclined to try the algorithm anyway. The above argument shows that $\phi(x)$ probably exists for $x > 0$, so we feel reasonably sure the fourth-order Runge–Kutta method will give a good approximation of the true solution $\phi(x)$. Proceeding with the algorithm, we use the starting values $x_0 = 0$ and $y_0 = 1$. Since $f(x, y) = x - y^2$, the formulas in Step 3 of the subroutine become

$$k_1 = h(x - y^2) , \qquad k_2 = h\left[\left(x + \frac{h}{2}\right) - \left(y + \frac{k_1}{2}\right)^2\right] ,$$

$$k_3 = h\left[\left(x + \frac{h}{2}\right) - \left(y + \frac{k_2}{2}\right)^2\right] , \qquad k_4 = h\left[(x + h) - (y + k_3)^2\right] .$$

In Table 3.6, we give the approximations $y(2; 2^{-m+1})$ for $\phi(2)$ for $m = 0, 1, 2, 3$, and 4. The algorithm stops at $m = 4$, since

$$|y(2; 0.125) - y(2; 0.25)| = 0.00000 .$$

Hence, $\phi(2) \approx 1.25132$ with a tolerance of 0.0001. ◆

TABLE 3.6 Classical Fourth-Order Runge–Kutta Approximation for $\phi(2)$

m	h	Approximation for $\phi(2)$	$\lvert y(2; h) - y(2; 2h)\rvert$
0	2.0	−8.33333	
1	1.0	1.27504	9.60837
2	0.5	1.25170	0.02334
3	0.25	1.25132	0.00038
4	0.125	1.25132	0.00000