PART I

Ordinary differential equations

1

Euler's method and beyond

1.1 Ordinary differential equations and the Lipschitz condition

We commence our exposition of the computational aspects of differential equations by examining closely numerical methods for *ordinary differential equations (ODEs).* This is important because of the central role of ODEs in a multitude of applications. Not less crucial is the critical part that numerical ODEs play in the design and analysis of computational methods for *partial differential equations (PDEs).* Thus, even if your main interest is in solving PDEs, ideally you should first master computational ODEs, not just to familiarize yourself with concepts, terminology and ideas but also because (as we will see in what follows) many discretization methods for PDEs reduce the underlying problem to the computation of ODEs.

Our goal is to approximate the solution of the problem

$$\boldsymbol{y}' = \boldsymbol{f}(t, \boldsymbol{y}), \quad t \geq t_0, \qquad \boldsymbol{y}(t_0) = \boldsymbol{y}_0. \tag{1.1}$$

Here $\boldsymbol{f}$ is a sufficiently well-behaved function that maps $[t_0, \infty) \times \mathbb{R}^d$ to $\mathbb{R}^d$ and the initial condition $\boldsymbol{y}_0 \in \mathbb{R}^d$ is a given vector; $\mathbb{R}^d$ denotes here – and elsewhere in this book – the d-dimensional real Euclidean space.

The 'niceness' of $\boldsymbol{f}$ may span a whole range of desirable attributes. At the very least, we insist on $\boldsymbol{f}$ obeying, in a given vector norm $\|\cdot\|$, the *Lipschitz condition*

$$\|\boldsymbol{f}(t, \boldsymbol{x}) - \boldsymbol{f}(t, \boldsymbol{y})\| \leq \lambda \|\boldsymbol{x} - \boldsymbol{y}\| \quad \text{for all} \quad \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d, \ \ t \geq t_0. \tag{1.2}$$

Here $\lambda > 0$ is a real constant that is independent of the choice of $\boldsymbol{x}$ and $\boldsymbol{y}$ – a *Lipschitz constant.* Subject to (1.2), it is possible to prove that the ODE system (1.1) possesses a unique solution.[1] Taking a stronger requirement, we may stipulate that $\boldsymbol{f}$ is an *analytic* function – in other words, that the Taylor series of $\boldsymbol{f}$ about every $(t, \boldsymbol{y}_0) \in [0, \infty) \times \mathbb{R}^d$ has a positive radius of convergence. It is then possible to prove that the solution $\boldsymbol{y}$ itself is analytic. Analyticity comes in handy, since much of our investigation of numerical methods is based on Taylor expansions, but it is often an excessive requirement and excludes many ODEs of practical importance.

In this volume we strive to steer a middle course between the complementary vices of mathematical nitpicking and of hand-waving. We solemnly undertake to avoid any

[1] We refer the reader to the Appendix for a brief refresher course on norms, existence and uniqueness theorems for ODEs and other useful odds and ends of mathematics.

needless mention of exotic function spaces that present the theory in its most general form, whilst desisting from woolly and inexact statements. Thus, we *always* assume that $\boldsymbol{f}$ is Lipschitz and, as necessary, may explicitly stipulate that it is analytic. An intelligent reader could, if the need arose, easily weaken many of our 'analytic' statements so that they are applicable also to sufficiently-differentiable functions.

1.2 Euler's method

Let us ponder briefly the meaning of the ODE (1.1). We possess two items of information: we know the value of $\boldsymbol{y}$ at a single point $t = t_0$ and, given any function value $\boldsymbol{y} \in \mathbb{R}^d$ and time $t \geq t_0$, we can tell the slope from the differential equation. The purpose of the exercise being to guess the value of $\boldsymbol{y}$ at a new point, the most elementary approach is to use linear interpolation. In other words, we estimate $\boldsymbol{y}(t)$ by making the approximation $\boldsymbol{f}(t, \boldsymbol{y}(t)) \approx \boldsymbol{f}(t_0, \boldsymbol{y}(t_0))$ for $t \in [t_0, t_0 + h]$, where $h > 0$ is sufficiently small. Integrating (1.1),

$$\boldsymbol{y}(t) = \boldsymbol{y}(t_0) + \int_{t_0}^{t} \boldsymbol{f}(\tau, \boldsymbol{y}(\tau))\,\mathrm{d}\tau \approx \boldsymbol{y}_0 + (t - t_0)\boldsymbol{f}(t_0, \boldsymbol{y}_0). \tag{1.3}$$

Given a sequence t_0, $t_1 = t_0 + h$, $t_2 = t_0 + 2h, \ldots$, where $h > 0$ is the *time step,* we denote by $\boldsymbol{y}_n$ a numerical estimate of the exact solution $\boldsymbol{y}(t_n)$, $n = 0, 1, \ldots$ Motivated by (1.3), we choose

$$\boldsymbol{y}_1 = \boldsymbol{y}_0 + h\boldsymbol{f}(t_0, \boldsymbol{y}_0).$$

This procedure can be continued to produce approximants at t_2, t_3 and so on. In general, we obtain the recursive scheme

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\boldsymbol{f}(t_n, \boldsymbol{y}_n), \qquad n = 0, 1, \ldots, \tag{1.4}$$

the celebrated *Euler method.*

Euler's method is not only the most elementary computational scheme for ODEs and, simplicity notwithstanding, of enduring practical importance. It is also the cornerstone of the numerical analysis of differential equations of evolution. In a deep and profound sense, all the fancy multistep and Runge–Kutta schemes that we shall discuss are nothing but a generalization of the basic paradigm (1.4).

◇ **Graphic interpretation** Euler's method can be illustrated pictorially. Consider, for example, the scalar *logistic equation* $y' = y(1 - y)$, $y(0) = \frac{1}{10}$.

Fig. 1.1 displays the first few steps of Euler's method, with a grotesquely large step $h = 1$. For each step we show the exact solution with initial condition $y(t_n) = y_n$ in the vicinity of $t_n = nh$ (dotted line) and the linear interpolation via Euler's method (1.4) (solid line).

The initial condition being, by definition, exact, so is the slope at t_0. However, instead of following a curved trajectory the numerical solution is piecewise-linear. Having reached t_1, say, we have moved to a wrong trajectory (i.e., corresponding to a different initial condition). The slope at t_1 is wrong – or,

rather, it is the correct slope of the wrong solution! Advancing further, we might well stray even more from the original trajectory.

A realistic goal of numerical solution is not, however, to avoid errors altogether; after all, we approximate since we do not know the exact solution in the first place! An error-generating mechanism exists in every algorithm for numerical ODEs and our purpose is to understand it and to ensure that, in a given implementation, errors do not accumulate beyond a specified tolerance. Remarkably, even the excessive step $h = 1$ leads in Fig. 1.1 to a relatively modest local error. ◇

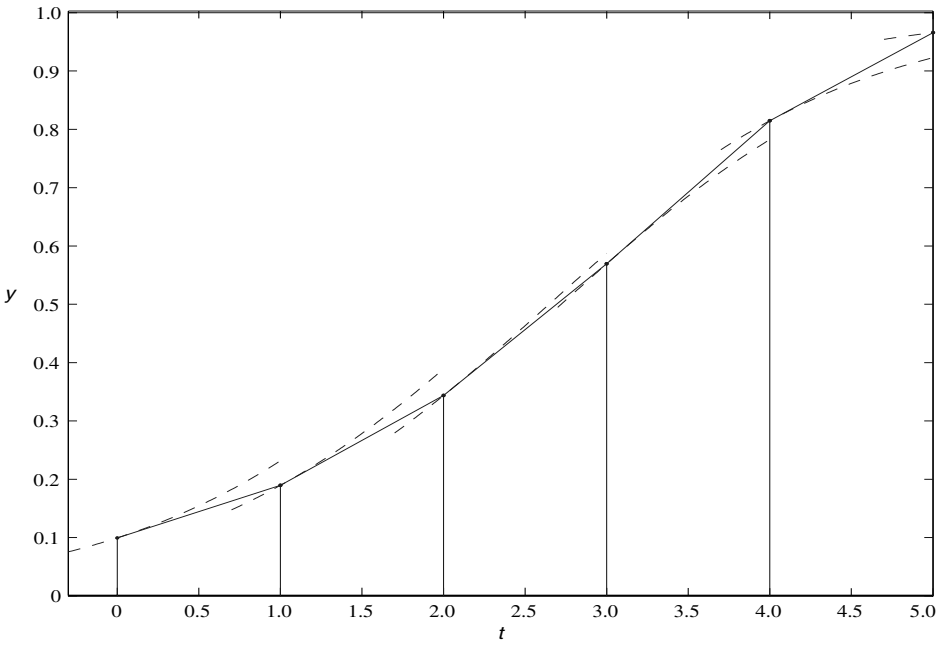

Figure 1.1 Euler's method, as applied to the equation $y' = y(1 - y)$ with initial value $y(0) = \frac{1}{10}$.

Euler's method can be easily extended to cater for variable steps. Thus, for a general monotone sequence $t_0 < t_1 < t_2 < \cdots$ we approximate as follows:

$$\boldsymbol{y}(t_{n+1}) \approx \boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h_n \boldsymbol{f}(t_n, \boldsymbol{y}_n),$$

where $h_n = t_{n+1} - t_n$, $n = 0, 1, \ldots$ However, for the time being we restrict ourselves to constant steps.

How good is Euler's method in approximating (1.1)? Before we even attempt to answer this question, we need to formulate it with considerably more rigour. Thus, suppose that we wish to compute a numerical solution of (1.1) in the compact interval

$[t_0, t_0+t^*]$ with some time-stepping numerical method, not necessarily Euler's scheme. In other words, we cover the interval by an equidistant grid and employ the time-stepping procedure to produce a numerical solution. Each grid is associated with a different numerical sequence and the critical question is whether, as $h \to 0$ and the grid is being refined, the numerical solution tends to the exact solution of (1.1). More formally, we express the dependence of the numerical solution upon the step size by the notation $\boldsymbol{y}_n = \boldsymbol{y}_{n,h}$, $n = 0, 1, \ldots, \lfloor t^*/h \rfloor$. A method is said to be *convergent* if, for every ODE (1.1) with a Lipschitz function $\boldsymbol{f}$ and every $t^* > 0$ it is true that

$$\lim_{h\to 0+} \max_{n=0,1,\ldots,\lfloor t^*/h \rfloor} \|\boldsymbol{y}_{n,h} - \boldsymbol{y}(t_n)\| = 0,$$

where $\lfloor \alpha \rfloor \in \mathbb{Z}$ is the integer part of $\alpha \in \mathbb{R}$. Hence, convergence means that, for every Lipschitz function, the numerical solution tends to the true solution as the grid becomes increasingly fine.[2]

In the next few chapters we will mention several desirable attributes of numerical methods for ODEs. It is crucial to understand that convergence is not just another 'desirable' property but, rather, a *sine qua non* of any numerical scheme. *Unless it converges, a numerical method is useless!*

Theorem 1.1 *Euler's method* (1.4) *is convergent.*

Proof We prove this theorem subject to the extra assumption that the function $\boldsymbol{f}$ (and therefore also $\boldsymbol{y}$) is analytic (it is enough, in fact, to stipulate the weaker condition of continuous differentiability).

Given $h > 0$ and $\boldsymbol{y}_n = \boldsymbol{y}_{n,h}$, $n = 0, 1, \ldots, \lfloor t^*/h \rfloor$, we let $\boldsymbol{e}_{n,h} = \boldsymbol{y}_{n,h} - \boldsymbol{y}(t_n)$ denote the numerical error. Thus, we wish to prove that $\lim_{h\to 0+} \max_n \|\boldsymbol{e}_{n,h}\| = 0$.

By Taylor's theorem and the differential equation (1.1),

$$\boldsymbol{y}(t_{n+1}) = \boldsymbol{y}(t_n) + h\boldsymbol{y}'(t_n) + \mathcal{O}\big(h^2\big) = \boldsymbol{y}(t_n) + h\boldsymbol{f}(t_n, \boldsymbol{y}(t_n)) + \mathcal{O}\big(h^2\big), \tag{1.5}$$

and, $\boldsymbol{y}$ being continuously differentiable, the $\mathcal{O}\big(h^2\big)$ term can be bounded (in a given norm) uniformly for all $h > 0$ and $n \le \lfloor t^*/h \rfloor$ by a term of the form ch^2, where $c > 0$ is a constant. We subtract (1.5) from (1.4), giving

$$\boldsymbol{e}_{n+1,h} = \boldsymbol{e}_{n,h} + h[\boldsymbol{f}(t_n, \boldsymbol{y}(t_n) + \boldsymbol{e}_{n,h}) - \boldsymbol{f}(t_n, \boldsymbol{y}(t_n))] + \mathcal{O}\big(h^2\big).$$

Thus, it follows by the triangle inequality from the Lipschitz condition and the aforementioned bound on the $\mathcal{O}\big(h^2\big)$ reminder term that

$$\begin{aligned} \|\boldsymbol{e}_{n+1,h}\| &\le \|\boldsymbol{e}_{n,h}\| + h\|\boldsymbol{f}(t_n, \boldsymbol{y}(t_n) + \boldsymbol{e}_{n,h}) - \boldsymbol{f}(t_n, \boldsymbol{y}(t_n))\| + ch^2 \\ &\le (1 + h\lambda)\|\boldsymbol{e}_{n,h}\| + ch^2, \qquad n = 0, 1, \ldots, \lfloor t^*/h \rfloor - 1. \end{aligned} \tag{1.6}$$

We now claim that

$$\|\boldsymbol{e}_{n,h}\| \le \frac{c}{\lambda} h\left[(1 + h\lambda)^n - 1\right], \qquad n = 0, 1, \ldots \tag{1.7}$$

[2] We have just introduced a norm through the back door: cf. appendix subsection A.1.3.3 for an exact definition. This, however, should cause no worry, since all norms are equivalent in finite-dimensional spaces. In other words, if a method is convergent in one norm, it converges in all ...

The proof is by induction on n. When $n = 0$ we need to prove that $\|\boldsymbol{e}_{0,h}\| \leq 0$ and hence that $\boldsymbol{e}_{0,h} = \boldsymbol{0}$. This is certainly true, since at t_0 the numerical solution matches the initial condition and the error is zero.

For general $n \geq 0$ we assume that (1.7) is true up to n and use (1.6) to argue that

$$\|\boldsymbol{e}_{n+1,h}\| \leq (1+h\lambda)\frac{c}{\lambda}h\left[(1+h\lambda)^n - 1\right] + ch^2 = \frac{c}{\lambda}h\left[(1+h\lambda)^{n+1} - 1\right].$$

This advances the inductive argument from n to $n+1$ and proves that (1.7) is true. The constant $h\lambda$ is positive, therefore $1 + h\lambda < \mathrm{e}^{h\lambda}$ and we deduce that $(1+h\lambda)^n < \mathrm{e}^{nh\lambda}$. The index n is allowed to range in $\{0, 1, \ldots, \lfloor t^*/h \rfloor\}$, hence $(1+h\lambda)^n < \mathrm{e}^{\lfloor t^*/h \rfloor h\lambda} \leq \mathrm{e}^{t^*\lambda}$. Substituting into (1.7), we obtain the inequality

$$\|\boldsymbol{e}_{n,h}\| \leq \frac{c}{\lambda}(\mathrm{e}^{t^*\lambda} - 1)h, \qquad n = 0, 1, \ldots, \lfloor t^*/h \rfloor.$$

Since $c(\mathrm{e}^{t^*\lambda} - 1)/\lambda$ is independent of h, it follows that

$$\lim_{\substack{h \to 0 \\ 0 \leq nh \leq t^*}} \|\boldsymbol{e}_{n,h}\| = 0.$$

In other words, Euler's method is convergent. ■

◇ **Health warning** At first sight, it might appear that there is more to the last theorem than meets the eye – not just a proof of convergence but also an upper bound on the error. In principle this is perfectly true: the error of Euler's method is indeed always bounded by $hc\mathrm{e}^{t^*\lambda}/\lambda$. Moreover, with very little effort it is possible to demonstrate, e.g. by using the *Peano kernel theorem* (A.2.2.6), that a reasonable choice is $c = \max_{t \in [t_0, t_0 + t^*]} \|\boldsymbol{y}''(t)\|$. The problem with this bound is that, unfortunately, in an overwhelming majority of practical cases it is too large by many orders of magnitude. It falls into the broad category of statements like 'the distance between London and New York is less than 47 light years' which, although manifestly true, fail to contribute significantly to the sum total of human knowledge.

The problem is not with the proof *per se* but with the insensitivity of a Lipschitz constant. A trivial example is the scalar linear equation $y' = -100y$, $y(0) = 1$. Therefore $\lambda = 100$ and, since $y(t) = \mathrm{e}^{-100t}$, $c = \lambda^2$. We thus derive the upper bound of $100h(\mathrm{e}^{100t^*} - 1)$. Letting $t^* = 1$, say, we have

$$|y_n - y(nh)| \leq 2.69 \times 10^{45} h. \tag{1.8}$$

It is easy, however, to show that $y_n = (1 - 100h)^n$, hence to derive the exact expression

$$|y_n - y(nh)| = \left|(1 - 100h)^n - \mathrm{e}^{-100nh}\right|$$

which is smaller by many orders of magnitude than (1.8) (note that, unless nh is very small, to all intents and purposes $\mathrm{e}^{-100nh} \approx 0$).

The moral of our discussion is simple. *The bound from the proof of Theorem 1.1 must not be used in practical estimations of numerical error!* ◇

Euler's method can be rewritten in the form $\boldsymbol{y}_{n+1} - [\boldsymbol{y}_n + h\boldsymbol{f}(t_n, \boldsymbol{y}_n)] = \boldsymbol{0}$. Replacing $\boldsymbol{y}_k$ by the exact solution $\boldsymbol{y}(t_k)$, $k = n, n+1$, and expanding the first few terms of the Taylor series about $t = t_0 + nh$, we obtain

$$\begin{aligned}&\boldsymbol{y}(t_{n+1}) - [\boldsymbol{y}(t_n) + h\boldsymbol{f}(t_n, \boldsymbol{y}(t_n))]\\ &= \left[\boldsymbol{y}(t_n) + h\boldsymbol{y}'(t_n) + \mathcal{O}\left(h^2\right)\right] - [\boldsymbol{y}(t_n) + h\boldsymbol{y}'(t_n)] = \mathcal{O}\left(h^2\right).\end{aligned}$$

We say that the Euler's method (1.4) is of *order* 1. In general, given an arbitrary time-stepping method

$$\boldsymbol{y}_{n+1} = \boldsymbol{\mathcal{Y}}_n(\boldsymbol{f}, h, \boldsymbol{y}_0, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_n), \qquad n = 0, 1, \ldots,$$

for the ODE (1.1), we say that it is of *order* p if

$$\boldsymbol{y}(t_{n+1}) - \boldsymbol{\mathcal{Y}}_n(\boldsymbol{f}, h, \boldsymbol{y}(t_0), \boldsymbol{y}(t_1), \ldots, \boldsymbol{y}(t_n)) = \mathcal{O}\left(h^{p+1}\right)$$

for every analytic $\boldsymbol{f}$ and $n = 0, 1, \ldots$ Alternatively, a method is of order p if it recovers *exactly* every polynomial solution of degree p or less.

The order of a numerical method provides us with information about its *local behaviour* – advancing from t_n to t_{n+1}, where $h > 0$ is sufficiently small, we are incurring an error of $\mathcal{O}\left(h^{p+1}\right)$. Our main interest, however, is in not the local but the *global* behaviour of the method: how well is it doing in a fixed bounded interval of integration as $h \to 0$? Does it converge to the true solution? How fast? Since the local error decays as $\mathcal{O}\left(h^{p+1}\right)$, the number of steps increases as $\mathcal{O}\left(h^{-1}\right)$. The naive expectation is that the global error decreases as $\mathcal{O}(h^p)$, but – as we will see in Chapter 2 – it cannot be taken for granted for each and every numerical method without an additional condition. As far as Euler's method is concerned, Theorem 1.1 demonstrates that all is well and that the error indeed decays as $\mathcal{O}(h)$.

1.3 The trapezoidal rule

Euler's method approximates the derivative by a constant in $[t_n, t_{n+1}]$, namely by its value at t_n (again, we denote $t_k = t_0 + kh$, $k = 0, 1, \ldots$). Clearly, the 'cantilevering' approximation is not very good and it makes more sense to make the constant approximation of the derivative equal to the average of its values at the endpoints. Bearing in mind that derivatives are given by the differential equation, we thus obtain an expression similar to (1.3):

$$\begin{aligned}\boldsymbol{y}(t) &= \boldsymbol{y}(t_n) + \int_{t_n}^{t} \boldsymbol{f}(\tau, \boldsymbol{y}(\tau))\,\mathrm{d}\tau\\ &\approx \boldsymbol{y}(t_n) + \tfrac{1}{2}(t - t_n)[\boldsymbol{f}(t_n, \boldsymbol{y}(t_n)) + \boldsymbol{f}(t, \boldsymbol{y}(t))].\end{aligned}$$

This is the motivation behind the *trapezoidal rule*

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \tfrac{1}{2}h[\boldsymbol{f}(t_n, \boldsymbol{y}_n) + \boldsymbol{f}(t_{n+1}, \boldsymbol{y}_{n+1})]. \tag{1.9}$$

To obtain the order of (1.9), we substitute the exact solution,

$$\begin{aligned}&\boldsymbol{y}(t_{n+1}) - \left\{\boldsymbol{y}(t_n) + \tfrac{1}{2}h[\boldsymbol{f}(t_n,\boldsymbol{y}(t_n)) + \boldsymbol{f}(t_{n+1},\boldsymbol{y}(t_{n+1}))]\right\}\\ &= \left[\boldsymbol{y}(t_n) + h\boldsymbol{y}'(t_n) + \tfrac{1}{2}h^2\boldsymbol{y}''(t_n) + \mathcal{O}\big(h^3\big)\right]\\ &\quad - \left(\boldsymbol{y}(t_n) + \tfrac{1}{2}h\left\{\boldsymbol{y}'(t_n) + \left[\boldsymbol{y}'(t_n) + h\boldsymbol{y}''(t_n) + \mathcal{O}\big(h^2\big)\right]\right\}\right) = \mathcal{O}\big(h^3\big)\,.\end{aligned}$$

Therefore the trapezoidal rule is of order 2.

Being forewarned of the shortcomings of local analysis, we should not jump to conclusions. Before we infer that the error decays globally as $\mathcal{O}\big(h^2\big)$, we must first prove that the method is convergent. Fortunately, this can be accomplished by a straightforward generalization of the method of proof of Theorem 1.1.

Theorem 1.2 *The trapezoidal rule* (1.9) *is convergent.*

Proof Subtracting

$$\boldsymbol{y}(t_{n+1}) = \boldsymbol{y}(t_n) + \tfrac{1}{2}h\left[\boldsymbol{f}(t_n,\boldsymbol{y}(t_n)) + \boldsymbol{f}(t_{n+1},\boldsymbol{y}(t_{n+1}))\right] + \mathcal{O}\big(h^3\big)$$

from (1.9), we obtain

$$\begin{aligned}\boldsymbol{e}_{n+1,h} = \boldsymbol{e}_{n,h} + \tfrac{1}{2}h\left\{\left[\boldsymbol{f}(t_n,\boldsymbol{y}_n) - \boldsymbol{f}(t_n,\boldsymbol{y}(t_n))\right]\right.\\ \left. + \left[\boldsymbol{f}(t_{n+1},\boldsymbol{y}_{n+1}) - \boldsymbol{f}(t_{n+1},\boldsymbol{y}(t_{n+1}))\right]\right\} + \mathcal{O}\big(h^3\big)\,.\end{aligned}$$

For analytic $\boldsymbol{f}$ we may bound the $\mathcal{O}\big(h^3\big)$ term by ch^3 for some $c > 0$, and this upper bound is valid uniformly throughout $[t_0, t_0 + t^*]$. Therefore, it follows from the Lipschitz condition (1.2) and the triangle inequality that

$$\|\boldsymbol{e}_{n+1,h}\| \le \|\boldsymbol{e}_{n,h}\| + \tfrac{1}{2}h\lambda\left\{\|\boldsymbol{e}_{n,h}\| + \|\boldsymbol{e}_{n+1,h}\|\right\} + ch^3.$$

Since we are ultimately interested in letting $h \to 0$ there is no harm in assuming that $h\lambda < 2$, and we can thus deduce that

$$\|\boldsymbol{e}_{n+1,h}\| \le \left(\frac{1+\frac{1}{2}h\lambda}{1-\frac{1}{2}h\lambda}\right)\|\boldsymbol{e}_{n,h}\| + \left(\frac{c}{1-\frac{1}{2}h\lambda}\right)h^3. \tag{1.10}$$

Our next step closely parallels the derivation of inequality (1.7). We thus argue that

$$\|\boldsymbol{e}_{n,h}\| \le \frac{c}{\lambda}\left[\left(\frac{1+\frac{1}{2}h\lambda}{1-\frac{1}{2}h\lambda}\right)^n - 1\right]h^2. \tag{1.11}$$

This follows by induction on n from (1.10) and is left as an exercise to the reader.

Since $0 < h\lambda < 2$, it is true that

$$\frac{1+\frac{1}{2}h\lambda}{1-\frac{1}{2}h\lambda} = 1 + \frac{h\lambda}{1-\frac{1}{2}h\lambda} \le \sum_{\ell=0}^{\infty}\frac{1}{\ell!}\left(\frac{h\lambda}{1-\frac{1}{2}h\lambda}\right)^{\ell} = \exp\left(\frac{h\lambda}{1-\frac{1}{2}h\lambda}\right).$$

Consequently, (1.11) yields

$$\|\boldsymbol{e}_{n,h}\| \le \frac{ch^2}{\lambda}\left(\frac{1+\frac{1}{2}h\lambda}{1-\frac{1}{2}h\lambda}\right)^n \le \frac{ch^2}{\lambda}\exp\left(\frac{nh\lambda}{1-\frac{1}{2}h\lambda}\right).$$

This bound is true for every nonnegative integer n such that $nh \le t^*$. Therefore

$$\|\boldsymbol{e}_{n,h}\| \le \frac{ch^2}{\lambda} \exp\left(\frac{t^*\lambda}{1-\frac{1}{2}h\lambda}\right)$$

and we deduce that

$$\lim_{\substack{h\to 0\\ 0\le nh\le t^*}} \|\boldsymbol{e}_{n,h}\| = 0.$$

In other words, the trapezoidal rule converges. ■

The number $ch^2 \exp[t^*\lambda/(1-\frac{1}{2}h\lambda)]/\lambda$ is, again, of absolutely no use in practical error bounds. However, a significant difference from Theorem 1.1 is that for the trapezoidal rule the error decays *globally* as $\mathcal{O}(h^2)$. This is to be expected from a second-order method if its convergence has been established.

Another difference between the trapezoidal rule and Euler's method is of an entirely different character. Whereas Euler's method (1.4) can be executed explicitly – knowing $\boldsymbol{y}_n$ we can produce $\boldsymbol{y}_{n+1}$ by computing a value of $\boldsymbol{f}$ and making a few arithmetic operations – this is not the case with (1.9). The vector $\boldsymbol{v} = \boldsymbol{y}_n + \frac{1}{2}h\boldsymbol{f}(t_n, \boldsymbol{y}_n)$ can be evaluated from known data, but that leaves us in each step with the task of finding $\boldsymbol{y}_{n+1}$ as the solution of the system of algebraic equations

$$\boldsymbol{y}_{n+1} - \tfrac{1}{2}h\boldsymbol{f}(t_{n+1}, \boldsymbol{y}_{n+1}) = \boldsymbol{v}.$$

The trapezoidal rule is thus said to be *implicit*, to distinguish it from the *explicit* Euler's method and its ilk.

Solving nonlinear equations is hardly a mission impossible, but we cannot take it for granted either. Only in texts on pure mathematics are we allowed to wave a magic wand, exclaim 'let $\boldsymbol{y}_{n+1}$ be a solution of ...' and assume that all our problems are over. As soon as we come to deal with actual computation, we had better specify how we plan (or our computer plans) to undertake the task of evaluating $\boldsymbol{y}_{n+1}$. This will be a theme of Chapter 7, which deals with the implementation of ODE methods. It suffices to state now that the cost of numerically solving nonlinear equations does not rule out the trapezoidal rule (and other implicit methods) as viable computational instruments. Implicitness is just one attribute of a numerical method and we must weigh it alongside other features.

◇ **A 'good' example** Figure 1.2 displays the (natural) logarithm of the error in the numerical solution of the scalar linear equation $y' = -y + 2\mathrm{e}^{-t}\cos 2t$, $y(0) = 0$ for (in descending order) $h = \frac{1}{2}$, $h = \frac{1}{10}$ and $h = \frac{1}{50}$.

How well does the plot illustrate our main distinction between Euler's method and the trapezoidal rule, namely faster decay of the error for the latter? As often in life, information is somewhat obscured by extraneous 'noise'; in the present case the error oscillates. This can be easily explained by the periodic component of the exact solution $y(t) = \mathrm{e}^{-t}\sin 2t$. Another observation is that, for both Euler's method and the trapezoidal rule, the error, twists and turns notwithstanding, does decay. This, on the face of it, can be explained by the decay of the exact solution but is an important piece of news nonetheless.

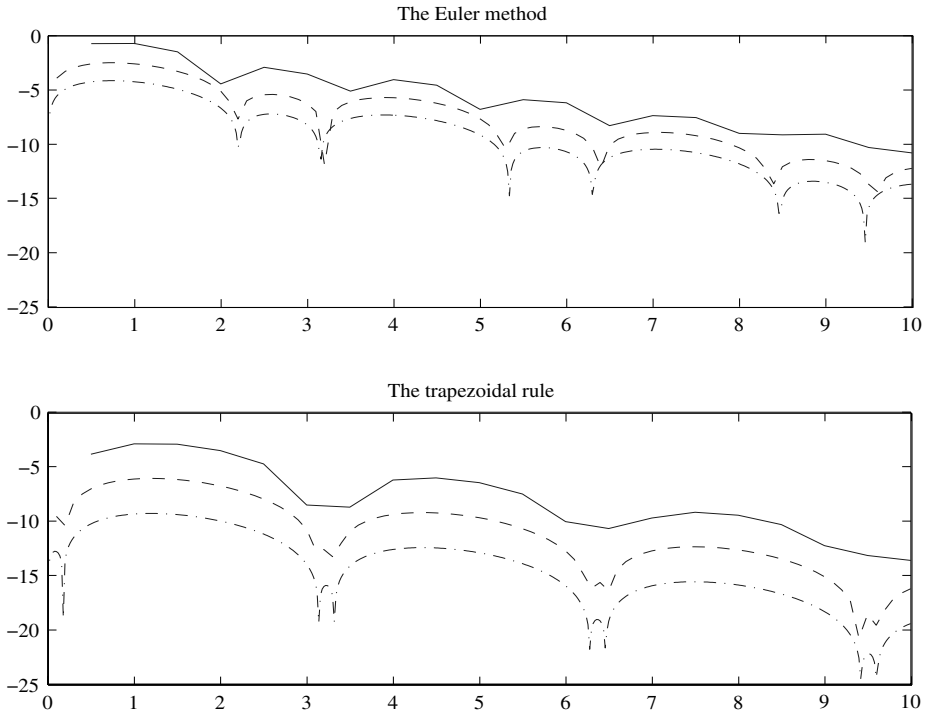

Figure 1.2 Euler's method and the trapezoidal rule, as applied to $y' = -y+2\mathrm{e}^{-t}\cos 2t$, $y(0)=0$. The logarithm of the error, $\ln|y_n-y(t_n)|$, is displayed for $h=\frac{1}{2}$ (solid line), $h=\frac{1}{10}$ (broken line) and $h=\frac{1}{50}$ (broken-and-dotted line).

Our most pessimistic assumption is that errors might accumulate from step to step but, as can be seen from this example, this prophecy of doom is often misplaced. This is a highly nontrivial point, which will be debated at greater length throughout Chapter 4.

Factoring out oscillations and decay, we observe that errors indeed decrease with h. More careful examination verifies that they increase at roughly the rate predicted by order considerations. Specifically, for a convergent method of order p we have $\|\boldsymbol{e}\| \approx ch^p$, hence $\ln\|\boldsymbol{e}\| \approx \ln c + p\ln h$. Denoting by $\boldsymbol{e}^{(1)}$ and $\boldsymbol{e}^{(2)}$ the errors corresponding to step sizes $h^{(1)}$ and $h^{(2)}$ respectively, it follows that $\ln\|\boldsymbol{e}^{(2)}\| \approx \ln\|\boldsymbol{e}^{(1)}\| - p\ln(h^{(2)}/h^{(1)})$. The ratio of consecutive step sizes in Fig. 1.2 being five, we expect the error to decay by (at least) a constant multiple of $\ln 5 \approx 1.6094$ and $2\ln 5 \approx 3.2189$ for Euler and the trapezoidal rule respectively. The actual error decays if anything slightly faster than this. ◇

◇ **A 'bad' example** Theorems 1.1 and 1.2 and, indeed, the whole numerical ODE theory, rest upon the assumption that (1.1) satisfies the Lipschitz con-