AIMS Lecture Notes 2006

Peter J. Olver

10. Numerical Solution of Ordinary Differential Equations

This part is concerned with the numerical solution of initial value problems for systems of ordinary differential equations. We will introduce the most basic one-step methods, beginning with the most basic Euler scheme, and working up to the extremely popular Runge–Kutta fourth order method that can be successfully employed in most situations. We end with a brief discussion of stiff differential equations, which present a more serious challenge to numerical analysts.

10.1. First Order Systems of Ordinary Differential Equations.

Let us begin by reviewing the theory of ordinary differential equations. Many physical applications lead to higher order systems of ordinary differential equations, but there is a simple reformulation that will convert them into equivalent first order systems. Thus, we do not lose any generality by restricting our attention to the first order case throughout. Moreover, numerical solution schemes for higher order initial value problems are entirely based on their reformulation as first order systems.

First Order Systems

A *first order system of ordinary differential equations* has the general form

$$\frac{du_1}{dt} = F_1(t, u_1, \ldots, u_n), \qquad \cdots \qquad \frac{du_n}{dt} = F_n(t, u_1, \ldots, u_n). \tag{10.1}$$

The unknowns $u_1(t), \ldots, u_n(t)$ are scalar functions of the real variable t, which usually represents time. We shall write the system more compactly in vector form

$$\frac{d\mathbf{u}}{dt} = \mathbf{F}(t, \mathbf{u}), \tag{10.2}$$

where $\mathbf{u}(t) = (\, u_1(t), \ldots, u_n(t)\,)^T$, and $\mathbf{F}(t, \mathbf{u}) = (\, F_1(t, u_1, \ldots, u_n), \ldots, F_n(t, u_1, \ldots, u_n)\,)^T$ is a vector-valued function of $n+1$ variables. By a *solution* to the differential equation, we mean a vector-valued function $\mathbf{u}(t)$ that is defined and continuously differentiable on an interval $a < t < b$, and, moreover, satisfies the differential equation on its interval of definition. Each solution $\mathbf{u}(t)$ serves to parametrize a curve $C \subset \mathbb{R}^n$, also known as a *trajectory* or *orbit* of the system.

In this part, we shall concentrate on initial value problems for such first order systems. The general initial conditions are

$$u_1(t_0) = a_1, \qquad u_2(t_0) = a_2, \qquad \cdots \qquad u_n(t_0) = a_n, \tag{10.3}$$

or, in vectorial form,

$$\mathbf{u}(t_0) = \mathbf{a} \tag{10.4}$$

Here t_0 is a prescribed initial time, while the vector $\mathbf{a} = (\, a_1, a_2, \ldots, a_n\,)^T$ fixes the initial position of the desired solution. In favorable situations, as described below, the initial conditions serve to uniquely specify a solution to the differential equations — at least for nearby times. The general issues of existence and uniquenss of solutions will be addressed in the following section.

A system of differential equations is called *autonomous* if the right hand side does not explicitly depend upon the time t, and so takes the form

$$\frac{d\mathbf{u}}{dt} = \mathbf{F}(\mathbf{u}). \tag{10.5}$$

One important class of autonomous first order systems are the steady state fluid flows. Here $\mathbf{F}(\mathbf{u}) = \mathbf{v}$ represents the fluid velocity vector field at the position $\mathbf{u}$. The solution $\mathbf{u}(t)$ to the initial value problem (10.5, 4) describes the motion of a fluid particle that starts at position $\mathbf{a}$ at time t_0. The differential equation tells us that the fluid velocity at each point on the particle's trajectory matches the prescribed vector field.

An *equilibrium solution* is constant: $\mathbf{u}(t) \equiv \mathbf{u}^\star$ for all t. Thus, its derivative must vanish, $d\mathbf{u}/dt \equiv \mathbf{0}$, and hence, every equilibrium solution arises as a solution to the system of algebraic equations

$$\mathbf{F}(\mathbf{u}^\star) = \mathbf{0} \tag{10.6}$$

prescribed by the vanishing of the right hand side of the system (10.5).

Example 10.1. Although a population of people, animals, or bacteria consists of individuals, the aggregate behavior can often be effectively modeled by a dynamical system that involves continuously varying variables. As first proposed by the English economist Thomas Malthus in 1798, the population of a species grows, roughly, in proportion to its size. Thus, the number of individuals $N(t)$ at time t satisfies a first order differential equation of the form

$$\frac{dN}{dt} = \rho N, \tag{10.7}$$

where the proportionality factor $\rho = \beta - \delta$ measures the rate of growth, namely the difference between the birth rate $\beta \geq 0$ and the death rate $\delta \geq 0$. Thus, if births exceed deaths, $\rho > 0$, and the population increases, whereas if $\rho < 0$, more individuals are dying and the population shrinks.

In the very simplest model, the growth rate ρ is assumed to be independent of the population size, and (10.7) reduces to the simple linear ordinary differential equation. The solutions satisfy the Malthusian exponential growth law $N(t) = N_0\, e^{\rho t}$, where $N_0 = N(0)$ is the initial population size. Thus, if $\rho > 0$, the population grows without limit, while if

$\rho < 0$, the population dies out, so $N(t) \to 0$ as $t \to \infty$, at an exponentially fast rate. The Malthusian population model provides a reasonably accurate description of the behavior of an isolated population in an environment with unlimited resources.

In a more realistic scenario, the growth rate will depend upon the size of the population as well as external environmental factors. For example, in the presence of limited resources, relatively small populations will increase, whereas an excessively large population will have insufficient resources to survive, and so its growth rate will be negative. In other words, the growth rate $\rho(N) > 0$ when $N < N^\star$, while $\rho(N) < 0$ when $N > N^\star$, where the *carrying capacity* $N^\star > 0$ depends upon the resource availability. The simplest class of functions that satifies these two inequalities are of the form $\rho(N) = \lambda(N^\star - N)$, where $\lambda > 0$ is a positive constant. This leads us to the nonlinear population model

$$\frac{dN}{dt} = \lambda N\,(N^\star - N). \tag{10.8}$$

In deriving this model, we assumed that the environment is not changing over time; a dynamical environment would require a more complicated non-autonomous differential equation.

Before analyzing the solutions to the nonlinear population model, let us make a preliminary change of variables, and set $u(t) = N(t)/N^\star$, so that u represents the size of the population in proportion to the *carrying capacity* $N^\star$. A straightforward computation shows that $u(t)$ satisfies the so-called *logistic differential equation*

$$\frac{du}{dt} = \lambda\,u\,(1-u), \qquad\qquad u(0) = u_0, \tag{10.9}$$

where we assign the initial time to be $t_0 = 0$ for simplicity. The logistic differential equation can be viewed as the continuous counterpart of the logistic map (2.22). However, unlike its discrete namesake, the logistic differential equation is quite sedate, and its solutions easily understood.

First, there are two equilibrium solutions: $u(t) \equiv 0$ and $u(t) \equiv 1$, obtained by setting the right hand side of the equation equal to zero. The first represents a nonexistent population with no individuals and hence no reproduction. The second equilibrium solution corresponds to a static population $N(t) \equiv N^\star$ that is at the ideal size for the environment, so deaths exactly balance births. In all other situations, the population size will vary over time.

To integrate the logistic differential equation, we proceed as above, first writing it in the separated form

$$\frac{du}{u(1-u)} = \lambda\,dt.$$

Integrating both sides, and using partial fractions,

$$\lambda t + k = \int \frac{du}{u(1-u)} = \int \left[\,\frac{1}{u} + \frac{1}{1-u}\,\right] du = \log \left| \frac{u}{1-u} \right| ,$$

where k is a constant of integration. Therefore

$$\frac{u}{1-u} = c\,e^{\lambda t}, \qquad \text{where} \qquad c = \pm\,e^{k}.$$

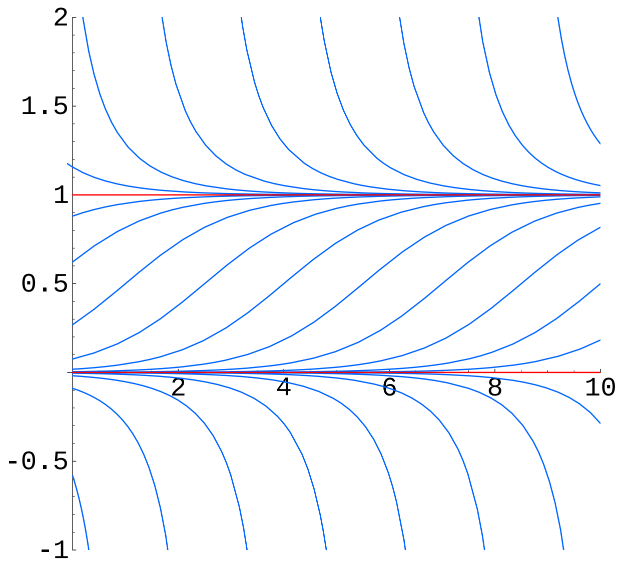


Figure 10.1. Solutions to $u' = u(1-u)$.

Solving for u, we deduce the solution

$$u(t) = \frac{c\,e^{\lambda t}}{1 + c\,e^{\lambda t}}\,. \tag{10.10}$$

The constant of integration is fixed by the initial condition. Solving the algebraic equation

$$u_0 = u(0) = \frac{c}{1+c} \qquad \text{yields} \qquad c = \frac{u_0}{1-u_0}\,.$$

Substituting the result back into the solution formula (10.10) and simplifying, we find

$$u(t) = \frac{u_0\,e^{\lambda t}}{1 - u_0 + u_0\,e^{\lambda t}}\,. \tag{10.11}$$

The resulting solutions are illustrated in Figure 10.1. Interestingly, while the equilibrium solutions are not covered by the integration method, they reappear in the final solution formula, corresponding to initial data $u_0 = 0$ and $u_0 = 1$ respectively. However, this is a lucky accident, and cannot be anticipated in more complicated situations.

When using the logistic equation to model population dynamics, the initial data is assumed to be positive, $u_0 > 0$. As time $t \to \infty$, the solution (10.11) tends to the equilibrium value $u(t) \to 1$ — which corresponds to $N(t) \to N^\star$ approaching the carrying capacity in the original population model. For small initial values $u_0 \ll 1$ the solution initially grows at an exponential rate λ, corresponding to a population with unlimited resources. However, as the population increases, the gradual lack of resources tends to slow down the growth rate, and eventually the population saturates at the equilibrium value. On the other hand, if $u_0 > 1$, the population is too large to be sustained by the available resources, and so dies off until it reaches the same saturation value. If $u_0 = 0$, then the solution remains at equilibrium $u(t) \equiv 0$. Finally, when $u_0 < 0$, the solution only exists for a finite amount of time, with

$$u(t) \longrightarrow -\infty \qquad \text{as} \qquad t \longrightarrow t^\star = \frac{1}{\lambda} \log\left(1 - \frac{1}{u_0}\right).$$

Of course, this final case does appear in the physical world, since we cannot have a negative population!

Example 10.2. A *predator-prey system* is a simplified ecological model of two species: the predators which feed on the prey. For example, the predators might be lions roaming the Serengeti and the prey zebra. We let $u(t)$ represent the number of prey, and $v(t)$ the number of predators at time t. Both species obey a population growth model of the form (10.7), and so the dynamical equations can be written as

$$\frac{du}{dt} = \rho u, \qquad \frac{dv}{dt} = \sigma v,$$

where the growth rates ρ, σ may depend upon the other species. The more prey, i.e., the larger u is, the faster the predators reproduce, while a lack of prey will cause them to die off. On the other hand, the more predators, the faster the prey are consumed and the slower their net rate of growth.

If we assume that the environment has unlimited resources for the prey, which, barring drought, is probably valid in the case of the zebras, then the simplest model that incorporates these assumptions is the *Lotka–Volterra system*

$$\frac{du}{dt} = \alpha u - \delta u v, \qquad \frac{dv}{dt} = -\beta v + \gamma u v, \tag{10.12}$$

corresponding to growth rates $\rho = \alpha - \delta v$, $\sigma = -\beta + \gamma u$. The parameters $\alpha, \beta, \gamma, \delta > 0$ are all positive, and their precise values will depend upon the species involved and how they interact, as indicated by field data, combined with, perhaps, educated guesses. In particular, α represents the unrestrained growth rate of the prey in the absence of predators, while $-\beta$ represents the rate that the predators die off in the absence of their prey. The nonlinear terms model the interaction of the two species: the rate of increase in the predators is proportional to the number of available prey, while the rate of decrese in the prey is proportional to the number of predators. The initial conditions $u(t_0) = u_0$, $v(t_0) = v_0$ represent the initial populations of the two species.

We will discuss the integration of the Lotka–Volterra system (10.12) below. Here, let us content ourselves with determining the possible equilibria. Setting the right hand sides of the system to zero leads to the nonlinear algebraic system

$$0 = \alpha u - \delta u v = u(\alpha - \delta v), \qquad 0 = -\beta v + \gamma u v = v(-\beta + \gamma u).$$

Thus, there are two distinct equilibria, namely

$$u_1^\star = v_1^\star = 0, \qquad u_2^\star = \beta/\gamma, \quad v_2^\star = \alpha/\delta.$$

The first is the uninteresting (or, rather catastropic) situation where there are no animals — no predators and no prey. The second is a nontrivial solution in which both populations maintain a steady value, for which the birth rate of the prey is precisely sufficient to continuously feed the predators. Is this a feasible solution? Or, to state the question more mathematically, is this a stable equilibrium? We shall develop the tools to answer this question below.

Higher Order Systems

A wide variety of physical systems are modeled by nonlinear systems of differential equations depending upon second and, occasionally, even higher order derivatives of the unknowns. But there is an easy device that will reduce any higher order ordinary differential equation or system to an equivalent first order system. "Equivalent" means that each solution to the first order system uniquely corresponds to a solution to the higher order equation and vice versa. The upshot is that, for all practical purposes, one only needs to analyze first order systems. Moreover, the vast majority of numerical solution algorithms are designed for first order systems, and so to numerically integrate a higher order equation, one must place it into an equivalent first order form.

We have already encountered the main idea in our discussion of the phase plane approach to second order scalar equations

$$\frac{d^2u}{dt^2} = F\left(t, u, \frac{du}{dt}\right). \tag{10.13}$$

We introduce a new dependent variable $v = \dfrac{du}{dt}$. Since $\dfrac{dv}{dt} = \dfrac{d^2u}{dt^2}$, the functions u, v satisfy the equivalent first order system

$$\frac{du}{dt} = v, \qquad \frac{dv}{dt} = F(t, u, v). \tag{10.14}$$

Conversely, it is easy to check that if $\mathbf{u}(t) = \left(u(t), v(t)\right)^T$ is any solution to the first order system, then its first component $u(t)$ defines a solution to the scalar equation, which establishes their equivalence. The basic initial conditions $u(t_0) = u_0$, $v(t_0) = v_0$, for the first order system translate into a pair of initial conditions $u(t_0) = u_0$, $\dot u(t_0) = v_0$, specifying the value of the solution and its first order derivative for the second order equation.

Similarly, given a third order equation

$$\frac{d^3u}{dt^3} = F\left(t, u, \frac{du}{dt}, \frac{d^2u}{dt^2}\right),$$

we set

$$v = \frac{du}{dt}, \qquad w = \frac{dv}{dt} = \frac{d^2u}{dt^2}.$$

The variables u, v, w satisfy the equivalent first order system

$$\frac{du}{dt} = v, \qquad \frac{dv}{dt} = w, \qquad \frac{dw}{dt} = F(t, u, v, w).$$

The general technique should now be clear.

Example 10.3. The forced *van der Pol equation*

$$\frac{d^2u}{dt^2} + (u^2 - 1)\,\frac{du}{dt} + u = f(t) \tag{10.15}$$

arises in the modeling of an electrical circuit with a triode whose resistance changes with the current. It also arises in certain chemical reactions and wind-induced motions of structures. To convert the van der Pol equation into an equivalent first order system, we set $v = du/dt$, whence

$$\frac{du}{dt} = v, \qquad\qquad \frac{dv}{dt} = f(t) - (u^2 - 1)\,v - u, \tag{10.16}$$

is the equivalent phase plane system.

Example 10.4. The Newtonian equations for a mass m moving in a potential force field are a second order system of the form

$$m\,\frac{d^2\mathbf{u}}{dt^2} = -\,\nabla F(\mathbf{u})$$

in which $\mathbf{u}(t) = \left(\,u(t), v(t), w(t)\,\right)^T$ represents the position of the mass and $F(\mathbf{u}) = F(u, v, w)$ the potential function. In components,

$$m\,\frac{d^2u}{dt^2} = -\,\frac{\partial F}{\partial u}\,, \qquad m\,\frac{d^2v}{dt^2} = -\,\frac{\partial F}{\partial v}\,, \qquad m\,\frac{d^2w}{dt^2} = -\,\frac{\partial F}{\partial w}\,. \tag{10.17}$$

For example, a planet moving in the sun's gravitational field satisfies the Newtonian system for the gravitational potential

$$F(\mathbf{u}) = -\,\frac{\alpha}{\|\,\mathbf{u}\,\|} = -\,\frac{\alpha}{\sqrt{u^2 + v^2 + w^2}}\,, \tag{10.18}$$

where α depends on the masses and the universal gravitational constant. (This simplified model ignores all interplanetary forces.) Thus, the mass' motion in such a gravitational force field follows the solution to the second order Newtonian system

$$m\,\frac{d^2\mathbf{u}}{dt^2} = -\,\nabla F(\mathbf{u}) = -\,\frac{\alpha\,\mathbf{u}}{\|\,\mathbf{u}\,\|^3} = \frac{\alpha}{(u^2 + v^2 + w^2)^{3/2}}\begin{pmatrix} u \\ v \\ w \end{pmatrix}.$$

The same system of ordinary differential equations describes the motion of a charged particle in a Coulomb electric force field, where the sign of α is positive for attracting opposite charges, and negative for repelling like charges.

To convert the second order Newton equations into a first order system, we set $\mathbf{v} = \dot{\mathbf{u}}$ to be the mass' velocity vector, with components

$$p = \frac{du}{dt}\,, \qquad q = \frac{dv}{dt}\,, \qquad r = \frac{dw}{dt}\,,$$

and so

$$\frac{du}{dt} = p, \qquad \frac{dv}{dt} = q, \qquad \frac{dw}{dt} = r, \tag{10.19}$$
$$\frac{dp}{dt} = -\frac{1}{m}\,\frac{\partial F}{\partial u}\,(u,v,w), \qquad \frac{dq}{dt} = -\frac{1}{m}\,\frac{\partial F}{\partial v}\,(u,v,w), \qquad \frac{dr}{dt} = -\frac{1}{m}\,\frac{\partial F}{\partial w}\,(u,v,w).$$

One of Newton's greatest acheivements was to solve this system in the case of the central gravitational potential (10.18), and thereby confirm the validity of Kepler's laws of planetary motion.

10.2. Existence, Uniqueness, and Continuous Dependence.

It goes without saying that there is no general analytical method that will solve all differential equations. Indeed, even relatively simple first order, scalar, non-autonomous ordinary differential equations cannot be solved in closed form. For example, the solution to the particular *Riccati equation*

$$\frac{du}{dt} = u^2 + t \tag{10.20}$$

cannot be written in terms of elementary functions, although there is a solution formula that relies on Airy functions. The *Abel equation*

$$\frac{du}{dt} = u^3 + t \tag{10.21}$$

fares even worse, since its general solution cannot be written in terms of even standard special functions — although power series solutions can be tediously ground out term by term. Understanding when a given differential equation can be solved in terms of elementary functions or known special functions is an active area of contemporary research, [**6**]. In this vein, we cannot resist mentioning that the most important class of exact solution techniques for differential equations are those based on symmetry. An introduction can be found in the author's graduate level monograph [**37**]; see also [**8**, **26**].

Existence

Before worrying about how to solve a differential equation, either analytically, qualitatively, or numerically, it behooves us to try to resolve the core mathematical issues of existence and uniqueness. First, does a solution exist? If, not, it makes no sense trying to find one. Second, is the solution uniquely determined? Otherwise, the differential equation probably has scant relevance for physical applications since we cannot use it as a predictive tool. Since differential equations inevitably have lots of solutions, the only way in which we can deduce uniqueness is by imposing suitable initial (or boundary) conditions.

Unlike partial differential equations, which must be treated on a case-by-case basis, there are complete general answers to both the existence and uniqueness questions for initial value problems for systems of ordinary differential equations. (Boundary value problems are more subtle.) While obviously important, we will not take the time to present the proofs of these fundamental results, which can be found in most advanced textbooks on the subject, including [**4**, **22**, **25**, **27**].

Let us begin by stating the Fundamental Existence Theorem for initial value problems associated with first order systems of ordinary differential equations.

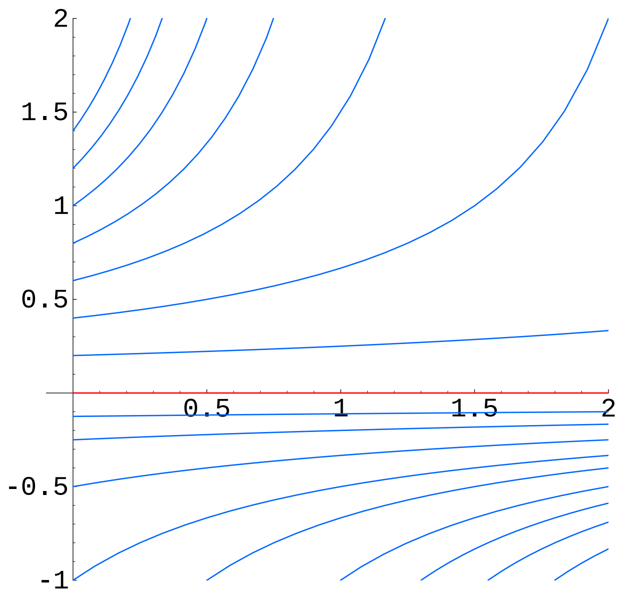


Figure 10.2. Solutions to $\dot{u} = u^2$.

Theorem 10.5. *Let $\mathbf{F}(t, \mathbf{u})$ be a continuous function. Then the initial value problem*[†]

$$\frac{d\mathbf{u}}{dt} = \mathbf{F}(t, \mathbf{u}), \qquad\qquad \mathbf{u}(t_0) = \mathbf{a}, \tag{10.22}$$

admits a solution $\mathbf{u} = \mathbf{f}(t)$ that is, at least, defined for nearby times, i.e., when $|\, t - t_0 \,| < \delta$ for some $\delta > 0$.

Theorem 10.5 guarantees that the solution to the initial value problem exists — at least for times sufficiently close to the initial instant t_0. This may be the most that can be said, although in many cases the maximal interval $\alpha < t < \beta$ of existence of the solution might be much larger — possibly infinite, $-\infty < t < \infty$, resulting in a *global solution*. The interval of existence of a solution typically depends upon both the equation and the particular initial data.

Example 10.6. Consider the autonomous initial value problem

$$\frac{du}{dt} = u^2, \qquad\qquad u(t_0) = u_0. \tag{10.23}$$

To solve the differential equation, we rewrite it in the separated form

$$\frac{du}{u^2} = dt, \qquad \text{and then integrate both sides:} \qquad -\,\frac{1}{u} = \int \frac{du}{u^2} = t + k.$$

[†] If $\mathbf{F}(t, \mathbf{u})$ is only defined on a subdomain $\Omega \subset \mathbb{R}^{n+1}$, then we must assume that the point $(t_0, \mathbf{a}) \in \Omega$ specifying the initial conditions belongs to its domain of definition.

Solving the resulting algebraic equation for u, we deduce the solution formula

$$u = -\,\frac{1}{t+k}\,. \tag{10.24}$$

To specify the integration constant k, we evaluate u at the initial time t_0; this implies

$$u_0 = -\,\frac{1}{t_0+k}\,, \qquad \text{so that} \qquad k = -\,\frac{1}{u_0} - t_0.$$

Therefore, the solution to the initial value problem is

$$u = \frac{u_0}{1-u_0(t-t_0)}\,. \tag{10.25}$$

Figure 10.2 shows the graphs of some typical solutions.

As t approaches the critical value $t^\star = t_0 + 1/u_0$ from below, the solution "blows up", meaning $u(t) \to \infty$ as $t \to t^\star$. The blow-up time $t^\star$ depends upon the initial data — the larger $u_0 > 0$ is, the sooner the solution goes off to infinity. If the initial data is negative, $u_0 < 0$, the solution is well-defined for all $t > t_0$, but has a singularity in the past, at $t^\star = t_0 + 1/u_0 < t_0$. The only solution that exists for all positive and negative time is the constant solution $u(t) \equiv 0$, corresponding to the initial condition $u_0 = 0$.

Thus, even though its right hand side is defined everywhere, the solutions to the scalar initial value problem (10.23) only exist up until time $1/u_0$, and so, the larger the initial data, the shorter the time of existence. In this example, the only global solution is the equilibrium solution $u(t) \equiv 0$. It is worth noting that this short-term existence phenomenon does not appear in the linear regime, where, barring singularities in the equation itself, solutions to a linear ordinary differential equation are guaranteed to exist for all time.

In practice, one always extends a solutions to its maximal interval of existence. The Existence Theorem 10.5 implies that there are only two possible ways in whcih a solution cannot be extended beyond a time $t^\star$: Either

(i) the solution becomes unbounded: $\|\,\mathbf{u}(t)\,\| \to \infty$ as $t \to t^\star$, or

(ii) if the right hand side $F(t,\mathbf{u})$ is only defined on a subset $\Omega \subset \mathbb{R}^{n+1}$, then the solution $\mathbf{u}(t)$ reaches the boundary $\partial\Omega$ as $t \to t^\star$.

If neither occurs in finite time, then the solution is necessarily global. In other words, a solution to an ordinary differential equation cannot suddenly vanish into thin air.

Remark: The existence theorem can be readily adapted to any higher order system of ordinary differential equations through the method of converting it into an equivalent first order system by introducing additional variables. The appropriate initial conditions guaranteeing existence are induced from those of the corresponding first order system, as in the second order example (10.13) discussed above.

Uniqueness and Smoothness

As important as existence is the question of uniqueness. Does the initial value problem have more than one solution? If so, then we cannot use the differential equation to predict

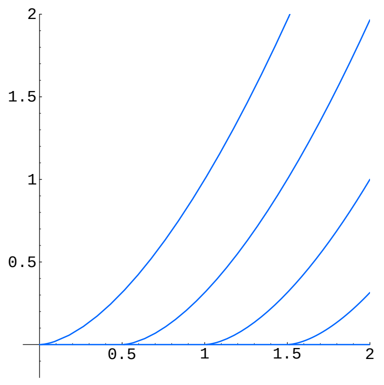


Figure 10.3. Solutions to the Differential Equation $\dot{u} = \frac{5}{3}\, u^{2/5}$.

the future behavior of the system from its current state. While continuity of the right hand side of the differential equation will guarantee that a solution exists, it is not quite sufficient to ensure uniqueness of the solution to the initial value problem. The difficulty can be appreciated by looking at an elementary example.

Example 10.7. Consider the nonlinear initial value problem

$$\frac{du}{dt} = \frac{5}{3}\, u^{2/5}, \qquad\qquad u(0) = 0. \tag{10.26}$$

Since the right hand side is a continuous function, Theorem 10.5 assures us of the existence of a solution — at least for t close to 0. This autonomous scalar equation can be easily solved by the usual method:

$$\int \frac{3}{5}\,\frac{du}{u^{2/5}} = u^{3/5} = t + c, \qquad\qquad \text{and so} \qquad\qquad u = (t+c)^{5/3}.$$

Substituting into the initial condition implies that $c = 0$, and hence $u(t) = t^{5/3}$ is a solution to the initial value problem.

On the other hand, since the right hand side of the differential equation vanishes at $u = 0$, the constant function $u(t) \equiv 0$ is an equilibrium solution to the differential equation. (Here is an example where the integration method fails to recover the equilibrium solution.) Moreover, the equilibrium solution has the same initial value $u(0) = 0$. Therefore, we have constructed two different solutions to the initial value problem (10.26). Uniqueness is *not* valid! Worse yet, there are, in fact, an *infinite* number of solutions to the initial value problem. For *any* $a > 0$, the function

$$u(t) = \begin{cases} 0, & 0 \le t \le a, \\ (t-a)^{5/3}, & t \ge a, \end{cases} \tag{10.27}$$

is differentiable everywhere, even at $t = a$. (Why?) Moreover, it satisfies both the differential equation and the initial condition, and hence defines a solution to the initial value problem. Several of these solutions are plotted in Figure 10.3.

Thus, to ensure uniqueness of solutions, we need to impose a more stringent condition, beyond mere continuity. The proof of the following basic uniqueness theorem can be found in the above references.

Theorem 10.8. *If $\mathbf{F}(t, \mathbf{u}) \in \mathrm{C}^1$ is continuously differentiable, then there exists one and only one solution*[†] *to the initial value problem* (10.22).

Thus, the difficulty with the differential equation (10.26) is that the function $F(u) = \frac{5}{3} u^{2/5}$, although continuous everywhere, is not differentiable at $u = 0$, and hence the Uniqueness Theorem 10.8 does not apply. On the other hand, $F(u)$ is continuously differentiable away from $u = 0$, and so any nonzero initial condition $u(t_0) = u_0 \neq 0$ will produce a unique solution — for as long as it remains away from the problematic value $u = 0$.

Blanket Hypothesis: From now on, all differential equations must satisfy the uniqueness criterion that their right hand side is continuously differentiable.

While continuous differentiability is sufficient to guarantee uniqueness of solutions, the smoother the right hand side of the system, the smoother the solutions. Specifically:

Theorem 10.9. *If $\mathbf{F} \in \mathrm{C}^n$ for $n \geq 1$, then any solution to the system $\dot{\mathbf{u}} = \mathbf{F}(t, \mathbf{u})$ is of class $\mathbf{u} \in \mathrm{C}^{n+1}$. If $\mathbf{F}(t, \mathbf{u})$ is an analytic function, then all solutions $\mathbf{u}(t)$ are analytic.*

One important consequence of uniqueness is that the solution trajectories of an autonomous system do not vary over time.

Proposition 10.10. *Consider an autonomous system $\dot{\mathbf{u}} = \mathbf{F}(\mathbf{u})$ whose right hand side $\mathbf{F} \in \mathrm{C}^1$. If $\mathbf{u}(t)$ is the solution to with initial condition $\mathbf{u}(t_0) = \mathbf{u}_0$, then the solution to the initial value problem $\widetilde{\mathbf{u}}(t_1) = \mathbf{u}_0$ is $\widetilde{\mathbf{u}}(t) = \mathbf{u}(t - t_1 + t_0)$.*

Note that the two solutions $\mathbf{u}(t)$ and $\widetilde{\mathbf{u}}(t)$ parametrize the *same* curve in $\mathbb{R}^n$, differing only by an overall "phase shift", $t_1 - t_0$, in their parametrizations. Thus, all solutions passing through the point $\mathbf{u}_0$ follow the same trajectory, irrespective of the time they arrive there. Indeed, not only is the trajectory the same, but the solutions have identical speeds at each point along the trajectory curve. For instance, if the right hand side of the system represents the velocity vector field of steady state fluid flow, Proposition 10.10 implies that the stream lines — the paths followed by the individual fluid particles — do not change in time, even though the fluid itself is in motion. This, indeed, is the meaning of the term "steady state" in fluid mechanics.

Continuous Dependence

In a real-world applications, initial conditions are almost never known exactly. Rather, experimental and physical errors will only allow us to say that their values are approximately equal to those in our mathematical model. Thus, to retain physical relevance, we need to be sure that small errors in our initial measurements do not induce a large change in the solution. A similar argument can be made for any physical parameters, e.g., masses,

[†] As noted earlier, we extend all solutions to their maximal interval of existence.

charges, spring stiffnesses, frictional coefficients, etc., that appear in the differential equation itself. A slight change in the parameters should not have a dramatic effect on the solution.

Mathematically, what we are after is a criterion of *continuous dependence* of solutions upon both initial data and parameters. Fortunately, the desired result holds without any additional assumptions, beyond requiring that the parameters appear continuously in the differential equation. We state both results in a single theorem.

Theorem 10.11. *Consider an initial value problem problem*

$$\frac{d\mathbf{u}}{dt} = \mathbf{F}(t, \mathbf{u}, \boldsymbol{\mu}), \qquad \mathbf{u}(t_0) = \mathbf{a}(\boldsymbol{\mu}), \tag{10.28}$$

in which the differential equation and/or the initial conditions depend continuously on one or more parameters $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_k)$. Then the unique[†] *solution $\mathbf{u}(t, \boldsymbol{\mu})$ depends continuously upon the parameters.*

Example 10.12. Let us look at a perturbed version

$$\frac{du}{dt} = \alpha\, u^2, \qquad u(0) = u_0 + \varepsilon,$$

of the initial value problem that we considered in Example 10.6. We regard ε as a small perturbation of our original initial data u_0, and α as a variable parameter in the equation. The solution is

$$u(t, \varepsilon) = \frac{u_0 + \varepsilon}{1 - \alpha\,(u_0 + \varepsilon)\, t}\,. \tag{10.29}$$

Note that, where defined, this is a continuous function of both parameters α, ε. Thus, a small change in the initial data, or in the equation, produces a small change in the solution — at least for times near the initial time.

Continuous dependence *does not* preclude nearby solutions from eventually becoming far apart. Indeed, the blow-up time $t^\star = 1/\big[\,\alpha\,(u_0 + \varepsilon)\,\big]$ for the solution (10.29) depends upon both the initial data and the parameter in the equation. Thus, as we approach the singularity, solutions that started out very close to each other will get arbitrarily far apart; see Figure 10.2 for an illustration.

An even simpler example is the linear model of exponential growth $\dot{u} = \alpha u$ when $\alpha > 0$. A very tiny change in the initial conditions has a negligible short term effect upon the solution, but over longer time intervals, the differences between the two solutions will be dramatic. Thus, the "sensitive dependence" of solutions on initial conditions already appears in very simple linear equations. For similar reasons, sontinuous dependence does *not* prevent solutions from exhibiting chaotic behavior. Further development of these ideas can be found in [**1**, **14**] and elsewhere.

[†] We continue to impose our blanket uniqueness hypothesis.

10.3. Numerical Methods.

Since we have no hope of solving the vast majority of differential equations in explicit, analytic form, the design of suitable numerical algorithms for accurately approximating solutions is essential. The ubiquity of differential equations throughout mathematics and its applications has driven the tremendous research effort devoted to numerical solution schemes, some dating back to the beginnings of the calculus. Nowadays, one has the luxury of choosing from a wide range of excellent software packages that provide reliable and accurate results for a broad range of systems, at least for solutions over moderately long time periods. However, all of these packages, and the underlying methods, have their limitations, and it is essential that one be able to to recognize when the software is working as advertised, and when it produces spurious results! Here is where the theory, particularly the classification of equilibria and their stability properties, as well as first integrals and Lyapunov functions, can play an essential role. Explicit solutions, when known, can also be used as test cases for tracking the reliability and accuracy of a chosen numerical scheme.

In this section, we survey the most basic numerical methods for solving initial value problems. For brevity, we shall only consider so-called single step schemes, culminating in the very popular and versatile fourth order Runge–Kutta Method. This should only serve as a extremely basic introduction to the subject, and many other important and useful methods can be found in more specialized texts, [**21**, **28**]. It goes without saying that some equations are more difficult to accurately approximate than others, and a variety of more specialized techniques are employed when confronted with a recalcitrant system. But all of the more advanced developments build on the basic schemes and ideas laid out in this section.

Euler's Method

The key issues confronting the numerical analyst of ordinary differential equations already appear in the simplest first order ordinary differential equation. Our goal is to calculate a decent approxiomation to the (unique) solution to the initial value problem

$$\frac{du}{dt} = F(t,u), \qquad\qquad u(t_0) = u_0. \tag{10.30}$$

To keep matters simple, we will focus our attention on the scalar case; however, all formulas and results written in a manner that can be readily adapted to first order systems — just replace the scalar functions $u(t)$ and $F(t,u)$ by vector-valued functions $\mathbf{u}$ and $\mathbf{F}(t,\mathbf{u})$ throughout. (The time t, of course, remains a scalar.) Higher order ordinary differential equations are inevitably handled by first converting them into an equivalent first order system, as discussed in Section 10.1, and then applying the numerical scheme.

The very simplest numerical solution method is named after Leonhard Euler — although Newton and his contemporaries were well aware of such a simple technique. Euler's Method is rarely used in practice because much more efficient and accurate techniques can be implemented with minimal additional work. Nevertheless, the method lies at the core of the entire subject, and must be thoroughly understood before progressing on to the more sophisticated algorithms that arise in real-world computations.

Starting at the initial time t_0, we introduce successive *mesh points* (or sample times)

$$t_0 < t_1 < t_2 < t_3 < \cdots ,$$

continuing on until we reach a desired final time $t_n = t^\star$. The mesh points should be fairly closely spaced. To keep the analysis as simple as possible, we will always use a uniform *step size*, and so

$$h = t_{k+1} - t_k > 0, \tag{10.31}$$

does not depend on k and is assumed to be relatively small. This assumption serves to simplify the analysis, and does not significantly affect the underlying ideas. For a uniform step size, the k^{th} mesh point is at $t_k = t_0 + k\,h$. More sophisticated *adaptive* methods, in which the step size is adjusted in order to maintain accuracy of the numerical solution, can be found in more specialized texts, e.g., [**21**, **28**]. Our numerical algorithm will recursively compute approximations $u_k \approx u(t_k)$, for $k = 0, 1, 2, 3, \ldots$, to the sampled values of the solution $u(t)$ at the chosen mesh points. Our goal is to make the *error* $E_k = u_k - u(t_k)$ in the approximation at each time t_k as small as possible. If required, the values of the solution $u(t)$ between mesh points may be computed by a subsequent interpolation procedure, e.g., the cubic splines of Section 13.3.

As you learned in first year calculus, the simplest approximation to a (continuously differentiable) function $u(t)$ is provided by its tangent line or first order Taylor polynomial. Thus, near the mesh point t_k

$$u(t) \;\approx\; u(t_k) + (t - t_k)\,\frac{du}{dt}(t_k) \;=\; u(t_k) + (t - t_k)\,F(t_k, u(t_k)),$$

in which we replace the derivative du/dt of the solution by the right hand side of the governing differential equation (10.30). In particular, the approximate value of the solution at the subsequent mesh point is

$$u(t_{k+1}) \;\approx\; u(t_k) + (t_{k+1} - t_k)\,F(t_k, u(t_k)). \tag{10.32}$$

This simple idea forms the basis of Euler's Method.

Since in practice we only know the approximation u_k to the value of $u(t_k)$ at the current mesh point, we are forced to replace $u(t_k)$ by its approximation u_k in the preceding formula. We thereby convert (10.32) into the iterative scheme

$$u_{k+1} = u_k + (t_{k+1} - t_k)\,F(t_k, u_k). \tag{10.33}$$

In particular, when based on a uniform step size (10.31), *Euler's Method* takes the simple form

$$u_{k+1} = u_k + h\,F(t_k, u_k). \tag{10.34}$$

As sketched in Figure 10.4, the method starts off approximating the solution reasonably well, but gradually loses accuracy as the errors accumulate.

Euler's Method is the simplest example of a *one-step* numerical scheme for integrating an ordinary differential equation. This refers to the fact that the succeeding approximation, $u_{k+1} \approx u(t_{k+1})$, depends only upon the current value, $u_k \approx u(t_k)$, which is one mesh point or "step" behind.

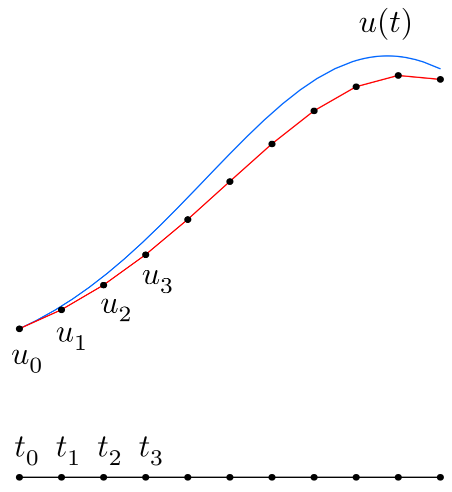


Figure 10.4. Euler's Method.

To begin to understand how Euler's Method works in practice, let us test it on a problem we know how to solve, since this will allow us to precisely monitor the resulting errors in our numerical approximation to the solution.

Example 10.13. The simplest "nontrivial" initial value problem is

$$\frac{du}{dt} = u, \qquad\qquad u(0) = 1,$$

whose solution is, of course, the exponential function $u(t) = e^t$. Since $F(t,u) = u$, Euler's Method (10.34) with a fixed step size $h > 0$ takes the form

$$u_{k+1} = u_k + h\,u_k = (1+h)\,u_k.$$

This is a linear iterative equation, and hence easy to solve:

$$u_k = (1+h)^k u_0 = (1+h)^k$$

is our proposed approximation to the solution $u(t_k) = e^{t_k}$ at the mesh point $t_k = k\,h$. Therefore, the Euler scheme to solve the differential equation, we are effectively approximating the exponential by a power function:

$$e^{t_k} = e^{k\,h} \;\approx\; (1+h)^k$$

When we use simply t to indicate the mesh time $t_k = k\,h$, we recover, in the limit, a well-known calculus formula:

$$e^t = \lim_{h\to 0}\,(1+h)^{t/h} = \lim_{k\to\infty}\left(1+\frac{t}{k}\right)^k. \tag{10.35}$$

A reader familiar with the computation of compound interest will recognize this particular approximation. As the time interval of compounding, h, gets smaller and smaller, the amount in the savings account approaches an exponential.

How good is the resulting approximation? The *error*

$$E(t_k) = E_k = u_k - e^{t_k}$$

measures the difference between the true solution and its numerical approximation at time $t = t_k = k\,h$. Let us tabulate the error at the particular times $t = 1, 2$ and 3 for various values of the step size h. The actual solution values are

$$e^1 = e = 2.718281828\ldots\,, \qquad e^2 = 7.389056096\ldots\,, \qquad e^3 = 20.085536912\ldots\,.$$

In this case, the numerical approximation always underestimates the true solution.

h	$E(1)$	$E(2)$	$E(3)$
.1	$-.125$	$-.662$	-2.636
.01	$-.0134$	$-.0730$	$-\ .297$
.001	$-.00135$	$-.00738$	$-\ .0301$
.0001	$-.000136$	$-.000739$	$-\ .00301$
.00001	$-.0000136$	$-.0000739$	$-\ .000301$

Some key observations:

- For a fixed step size h, the further we go from the initial point $t_0 = 0$, the larger the magnitude of the error.
- On the other hand, the smaller the step size, the smaller the error at a fixed value of t. The trade-off is that more steps, and hence more computational effort[†] is required to produce the numerical approximation. For instance, we need $k = 10$ steps of size $h = .1$, but $k = 1000$ steps of size $h = .001$ to compute an approximation to $u(t)$ at time $t = 1$.
- The error is more or less in proportion to the step size. Decreasing the step size by a factor of $\frac{1}{10}$ decreases the error by a similar amount, but simultaneously increases the amount of computation by a factor of 10.

The final observation indicates that the Euler Method is of *first order*, which means that the error depends *linearly*[‡] on the step size h. More specifically, at a fixed time t, the error is bounded by

$$|\,E(t)\,| = |\,u_k - u(t)\,| \leq C(t)\,h, \qquad \text{when} \qquad t = t_k = k\,h, \tag{10.36}$$

for some positive $C(t) > 0$ that depends upon the time, and the initial condition, but not on the step size.

[†] In this case, there happens to be an explicit formula for the numerical solution which can be used to bypass the iterations. However, in almost any other situation, one cannot compute the approximation u_k without having first determined the intermediate values $u_0, \ldots, u_{k-1}$.

[‡] See the discussion of the order of iterative methods in Section 2.1 for motivation.

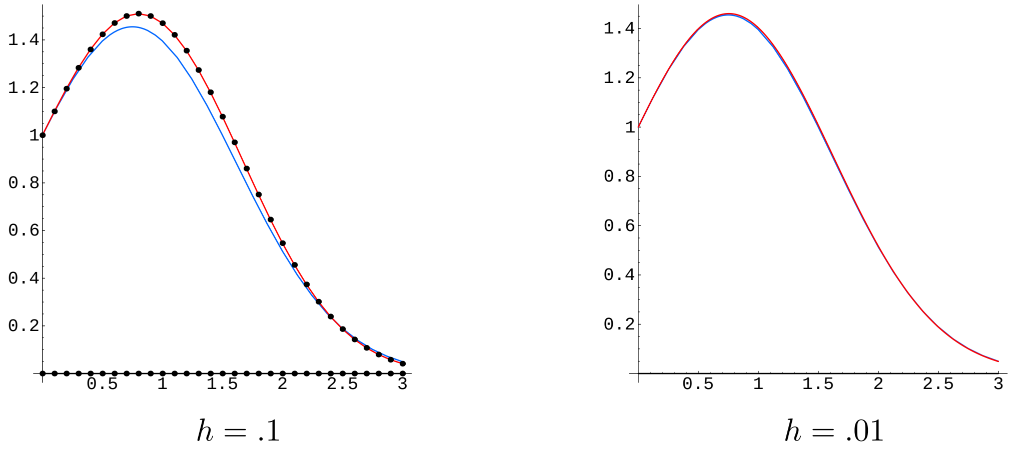


Figure 10.5. Euler's Method for $\dot{u} = \left(1 - \frac{4}{3}\,t\right) u$.

Example 10.14. The solution to the initial value problem

$$\frac{du}{dt} = \left(1 - \tfrac{4}{3}\,t\right) u, \qquad u(0) = 1, \tag{10.37}$$

is found by the method of separation of variables:

$$u(t) = \exp\left(t - \tfrac{2}{3}\,t^2\right). \tag{10.38}$$

Euler's Method leads to the iterative numerical scheme

$$u_{k+1} = u_k + h\left(1 - \tfrac{4}{3}\,t_k\right) u_k, \qquad u_0 = 1.$$

In Figure 10.5 we compare the graphs of the actual and numerical solutions for step sizes $h = .1$ and $.01$. In the former plot, we expliticly show the mesh points, but not in the latter, since they are too dense; moreover, the graphs of the numerical and true solutions are almost indistinguishable at this resolution.

The following table lists the numerical errors $E(t_k) = u_k - u(t_k)$ between the computed and actual solution values

$$u(1) = 1.395612425\ldots\,, \qquad u(2) = .513417119\ldots\,, \qquad u(3) = .049787068\ldots\,,$$

for several different step sizes:

h	$E(1)$	$E(2)$	$E(3)$
.1000	.07461761	.03357536	$-$.00845267
.0100	.00749258	.00324416	$-$.00075619
.0010	.00074947	.00032338	$-$.00007477
.0001	.00007495	.00003233	$-$.00000747

As in the previous example, each decrease in step size by a factor of 10 leads to one additional decimal digit of accuracy in the computed solution.

Taylor Methods

In general, the order of a numerical solution method governs both the accuracy of its approximations and the speed at which they converge to the true solution as the step size is decreased. Although the Euler Method is simple and easy to implement, it is only a first order scheme, and therefore of limited utility in serious computations. So, the goal is to devise simple numerical methods that enjoy a much higher order of accuracy.

Our derivation of the Euler Method was based on a first order Taylor approximation to the solution. So, an evident way to design a higher order method is to employ a higher order Taylor approximation. The Taylor series expansion for the solution $u(t)$ at the succeeding mesh point $t_{k+1} = t_k + h$ has the form

$$u(t_{k+1}) = u(t_k + h) = u(t_k) + h\,\frac{du}{dt}(t_k) + \frac{h^2}{2}\,\frac{d^2u}{dt^2}(t_k) + \frac{h^3}{6}\,\frac{d^3u}{dt^3}(t_k) + \ \cdots\ . \tag{10.39}$$

As we just saw, we can evaluate the first derivative term through use of the underlying differential equation:

$$\frac{du}{dt} = F(t,u). \tag{10.40}$$

The second derivative term can be found by differentiating the equation with respect to t. Invoking the chain rule[†],

$$\begin{aligned}\frac{d^2u}{dt^2} = \frac{d}{dt}\,\frac{du}{dt} = \frac{d}{dt}\,F(t,u(t)) &= \frac{\partial F}{\partial t}(t,u) + \frac{\partial F}{\partial u}(t,u)\,\frac{du}{dt} \\ &= \frac{\partial F}{\partial t}(t,u) + \frac{\partial F}{\partial u}(t,u)\,F(t,u) \equiv F^{(2)}(t,u).\end{aligned} \tag{10.41}$$

This operation is known as the *total derivative*, indicating that that we must treat the second variable u as a function of t when differentiating. Substituting (10.40–41) into (10.39) and truncating at order h^2 leads to the *Second Order Taylor Method*

$$\begin{aligned}u_{k+1} &= u_k + h\,F(t_k,u_k) + \frac{h^2}{2}\,F^{(2)}(t_k,u_k) \\ &= u_k + h\,F(t_k,u_k) + \frac{h^2}{2}\left(\frac{\partial F}{\partial t}(t_k,u_k) + \frac{\partial F}{\partial u}(t_k,u_k)\,F(t_k,u_k)\right),\end{aligned} \tag{10.42}$$

in which, as before, we replace the solution value $u(t_k)$ by its computed approximation u_k. The resulting method is of second order, meaning that the error function satisfies the quadratic error estimate

$$|\,E(t)\,| = |\,u_k - u(t)\,| \le C(t)\,h^2 \qquad \text{when} \qquad t = t_k = k\,h. \tag{10.43}$$

[†] We assume throughout that F has as many continuous derivatives as needed.

Example 10.15. Let us explicitly formulate the second order Taylor Method for the initial value problem (10.37). Here

$$\begin{aligned}\frac{du}{dt} &= F(t,u) = \left(1 - \tfrac{4}{3}\,t\right) u,\\ \frac{d^2u}{dt^2} &= \frac{d}{dt}\,F(t,u) = -\tfrac{4}{3}\,u + \left(1 - \tfrac{4}{3}\,t\right)\,\frac{du}{dt} = -\tfrac{4}{3}\,u + \left(1 - \tfrac{4}{3}\,t\right)^2 u,\end{aligned}$$

and so (10.42) becomes

$$u_{k+1} = u_k + h\left(1 - \tfrac{4}{3}\,t_k\right) u_k + \tfrac{1}{2}\,h^2\left[-\tfrac{4}{3}\,u_k + \left(1 - \tfrac{4}{3}\,t_k\right)^2 u_k\right], \qquad u_0 = 1.$$

The following table lists the errors between the values computed by the second order Taylor scheme and the actual solution values, as given in Example 10.14.

h	$E(1)$	$E(2)$	$E(3)$
.100	.00276995	−.00133328	.00027753
.010	.00002680	−.00001216	.00000252
.001	.00000027	−.00000012	.00000002

In accordance with the quadratic error estimate (10.43), a decrease in the step size by a factor of $\frac{1}{10}$ leads in an increase in accuracy of the solution by a factor $\frac{1}{100}$, i.e., an increase in 2 significant decimal places in the numerical approximation of the solution.

Higher order Taylor Methods can be readily established by including further terms in the expansion (10.39). However, they are rarely used in practice, for two reasons:

- Owing to their dependence upon the partial derivatives of $F(t,u)$, the right hand side of the differential equation needs to be rather smooth.
- Even worse, the explicit formulae become exceedingly complicated, even for relatively simple functions $F(t,u)$. Efficient evaluation of the multiplicity of terms in the Taylor approximation and avoidance of round off errors become significant concerns.

As a result, mathematicians soon abandoned the Taylor series approach, and began to look elsewhere for high order, efficient integration methods.

Error Analysis

Before pressing on, we need to engage in a more serious discussion of the error in a numerical scheme. A general *one-step* numerical method can be written in the form

$$u_{k+1} = G(h, t_k, u_k), \tag{10.44}$$

where G is a prescribed function of the current approximate solution value $u_k \approx u(t_k)$, the time t_k, and the step size $h = t_{k+1} - t_k$, which, for illustrative purposes, we assume to be fixed. (We leave the discussion of *multi-step methods*, in which G could also depend upon the earlier values $u_{k-1}, u_{k-2}, \ldots$, to more advanced texts, e.g., [**21**, **28**].)

In any numerical integration scheme there are, in general, three sources of error.

- The first is the *local error* committed in the current step of the algorithm. Even if we have managed to compute a completely accurate value of the solution $u_k = u(t_k)$ at time t_k, the numerical approximation scheme (10.44) is almost certainly not exact, and will therefore introduce an error into the next computed value $u_{k+1} \approx u(t_{k+1})$. Round-off errors, resulting from the finite precision of the computer arithmetic, will also contribute to the local error.
- The second is due to the error that is already present in the current approximation $u_k \approx u(t_k)$. The local errors tend to accumulate as we continue to run the iteration, and the net result is the *global error*, which is what we actually observe when compaing the numerical apporximation with the exact solution.
- Finally, if the initial condition $u_0 \approx u(t_0)$ is not computed accurately, this *initial error* will also make a contribution. For example, if $u(t_0) = \pi$, then we introduce some initial error by using a decimal approximation, say $\pi \approx 3.14159$.

The third error source will, for simplicity, be ignored in our discussion, i.e., we will assume $u_0 = u(t_0)$ is exact. Further, for simplicity we will assume that round-off errors do not play any significant role — although one must always keep them in mind when analyzing the computation. Since the global error is entirely due to the accumulation of successive local errors, we must first understand the local error in detail.

To measure the local error in going from t_k to t_{k+1}, we compare the exact solution value $u(t_{k+1})$ with its numerical approximation (10.44) under the assumption that the current computed value is correct: $u_k = u(t_k)$. Of course, in practice this is never the case, and so the local error is an artificial quantity. Be that as it may, in most circumstances the local error is (*a*) easy to estimate, and, (*b*) provides a reliable guide to the global accuracy of the numerical scheme. To estimate the local error, we assume that the step size h is small and approximate the solution $u(t)$ by its Taylor expansion[†]

$$\begin{aligned} u(t_{k+1}) &= u(t_k) + h\,\frac{du}{dt}(t_k) + \frac{h^2}{2}\,\frac{d^2u}{dt^2}(t_k) + \cdots \\ &= u_k + h\,F(t_k,u_k) + \frac{h^2}{2}\,F^{(2)}(t_k,u_k) + \cdots . \end{aligned} \tag{10.45}$$

In the second expression, we have employed (10.41) and its higher order analogs to evaluate the derivative terms, and then invoked our local accuracy assumption to replace $u(t_k)$ by u_k. On the other hand, a direct Taylor expansion, in h, of the numerical scheme produces

$$u_{k+1} = G(h,t_k,u_k) = G(0,t_k,u_k) + h\,\frac{\partial G}{\partial h}\,(0,t_k,u_k) + \frac{h^2}{2}\,\frac{\partial^2 G}{\partial h^2}\,(0,t_k,u_k) + \cdots . \tag{10.46}$$

The local error is obtained by comparing these two Taylor expansions.

Definition 10.16. A numerical integration method is of *order* n if the Taylor expansions (10.45, 46) of the exact and numerical solutions agree up to order h^n.

[†] In our analysis, we assume that the differential equation, and hence the solution, has sufficient smoothness to justify the relevant Taylor approximation.

For example, the Euler Method

$$u_{k+1} = G(h, t_k, u_k) = u_k + h\,F(t_k, u_k),$$

is already in the form of a Taylor expansion — that has no terms involving $h^2, h^3, \ldots$. Comparing with the exact expansion (10.45), we see that the constant and order h terms are the same, but the order h^2 terms differ (unless $F^{(2)} \equiv 0$). Thus, according to the definition, the Euler Method is a first order method. Similarly, the Taylor Method (10.42) is a second order method, because it was explicitly designed to match the constant, h and h^2 terms in the Taylor expansion of the solution. The general Taylor Method of order n sets $G(h, t_k, u_k)$ to be exactly the order n Taylor polynomial, differing from the full Taylor expansion at order h^{n+1}.

Under fairly general hypotheses, it can be proved that, if the numerical scheme has order n as measured by the local error, then the *global error* is bounded by a multiple of h^n. In other words, assuming no round-off or initial error, the computed value u_k and the solution at time t_k can be bounded by

$$|\,u_k - u(t_k)\,| \leq M\,h^n, \tag{10.47}$$

where the constant $M > 0$ may depend on the time t_k and the particular solution $u(t)$. The error bound (10.47) serves to justify our numerical observations. For a method of order n, decreasing the step size by a factor of $\frac{1}{10}$ will decrease the overall error by a factor of about 10^{-n}, and so, roughly speaking, we anticipate gaining an additional n digits of accuracy — at least up until the point that round-off errors begin to play a role. Readers interested in a more complete error analysis of numerical integration schemes should consult a specialized text, e.g., [**21**, **28**].

The bottom line is the higher its order, the more accurate the numerical scheme, and hence the larger the step size that can be used to produce the solution to a desired accuracy. However, this must be balanced with the fact that higher order methods inevitably require more computational effort at each step. If the total amount of computation has also decreased, then the high order method is to be preferred over a simpler, lower order method. Our goal now is to find another route to the design of higher order methods that avoids the complications inherent in a direct Taylor expansion.

An Equivalent Integral Equation

The secret to the design of higher order numerical algorithms is to replace the differential equation by an equivalent integral equation. By way of motivation, recall that, in general, differentiation is a badly behaved process; a reasonable function can have an unreasonable derivative. On the other hand, integration ameliorates; even quite nasty functions have relatively well-behaved integrals. For the same reason, accurate numerical integration is relatively painless, whereas numerical differentiation should be avoided unless necessary. While we have not dealt directly with integral equations in this text, the subject has been extensively developed by mathematicians, [**10**], and has many important physical applications.

Conversion of an initial value problem (10.30) to an integral equation is straightforward. We integrate both sides of the differential equation from the initial point t_0 to a variable time t. The Fundamental Theorem of Calculus is used to explicitly evaluate the left hand integral:

$$u(t) - u(t_0) = \int_{t_0}^{t} \dot{u}(s)\, ds = \int_{t_0}^{t} F(s, u(s))\, ds.$$

Rearranging terms, we arrive at the key result.

Lemma 10.17. *The solution $u(t)$ to the the integral equation*

$$u(t) = u(t_0) + \int_{t_0}^{t} F(s, u(s))\, ds \tag{10.48}$$

coincides with the solution to the initial value problem $\dfrac{du}{dt} = F(t,u)$, $u(t_0) = u_0$.

Proof: Our derivation already showed that the solution to the initial value problem satisfies the integral equation (10.48). Conversely, suppose that $u(t)$ solves the integral equation. Since $u(t_0) = u_0$ is constant, the Fundamental Theorem of Calculus tells us that the derivative of the right hand side of (10.48) is equal to the integrand, so $\dfrac{du}{dt} = F(t, u(t))$. Moreover, at $t = t_0$, the upper and lower limits of the integral coincide, and so it vanishes, whence $u(t) = u(t_0) = u_0$ has the correct initial conditions. *Q.E.D.*

Observe that, unlike the differential equation, the integral equation (10.48) requires no additional initial condition — it is automatically built into the equation. The proofs of the fundamental existence and uniqueness Theorems 10.5 and 10.8 for ordinary differential equations are, in fact, based on the integral equation reformulation of the initial value problem; see [**22**, **25**] for details.

The integral equation reformulation is equally valid for systems of first order ordinary differential equations. As noted above, $\mathbf{u}(t)$ and $\mathbf{F}(t, \mathbf{u}(t))$ become vector-valued functions. Integrating a vector-valued function is accomplished by integrating its individual components. Complete details are left to the exercises.

Implicit and Predictor–Corrector Methods

From this point onwards, we shall abandon the original initial value problem, and turn our attention to numerically solving the equivalent integral equation (10.48). Let us rewrite the integral equation, starting at the mesh point t_k instead of t_0, and integrating until time $t = t_{k+1}$. The result is the basic integral formula

$$u(t_{k+1}) = u(t_k) + \int_{t_k}^{t_{k+1}} F(s, u(s))\, ds \tag{10.49}$$

that (implicitly) computes the value of the solution at the subsequent mesh point. Comparing this formula with the Euler Method

$$u_{k+1} = u_k + h\, F(t_k, u_k), \qquad \text{where} \qquad h = t_{k+1} - t_k,$$

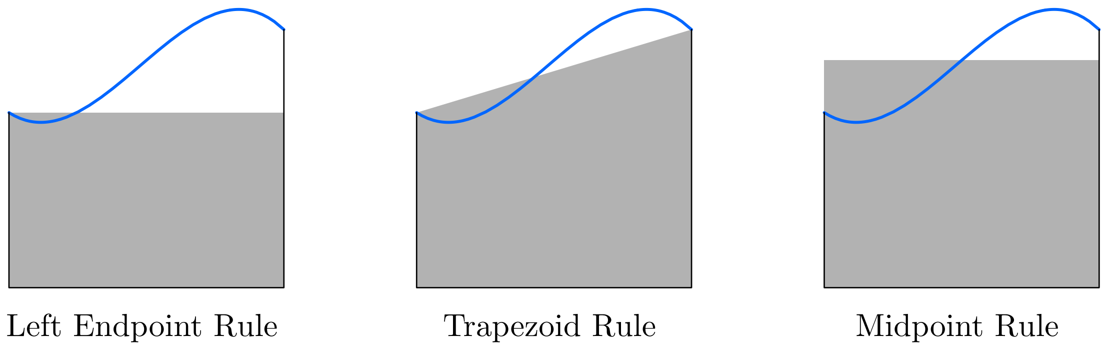


Figure 10.6. Numerical Integration Methods.

and assuming for the moment that $u_k = u(t_k)$ is exact, we discover that we are merely approximating the integral by

$$\int_{t_k}^{t_{k+1}} F(s,u(s))\,ds \;\approx\; h\,F(t_k,u(t_k)). \tag{10.50}$$

This is the Left Endpoint Rule for numerical integration — that approximates the area under the curve $g(t) = F(t,u(t))$ between $t_k \le t \le t_{k+1}$ by the area of a rectangle whose height $g(t_k) = F(t_k,u(t_k)) \approx F(t_k,u_k)$ is prescribed by the left-hand endpoint of the graph. As indicated in Figure 10.6, this is a reasonable, but not especially accurate method of numerical integration.

In first year calculus, you no doubt encountered much better methods of approximating the integral of a function. One of these is the *Trapezoid Rule*, which approximates the integral of the function $g(t)$ by the area of a trapezoid obtained by connecting the two points $(t_k, g(t_k))$ and $(t_{k+1}, g(t_{k+1}))$ on the graph of g by a straight line, as in the second Figure 10.6. Let us therefore try replacing (10.50) by the more accurate trapezoidal approximation

$$\int_{t_k}^{t_{k+1}} F(s,u(s))\,ds \;\approx\; \tfrac{1}{2}\,h\left[\,F(t_k,u(t_k)) + F(t_{k+1},u(t_{k+1}))\,\right]. \tag{10.51}$$

Substituting this approximation into the integral formula (10.49), and replacing the solution values $u(t_k), u(t_{k+1})$ by their numerical approximations, leads to the (hopefully) more accurate numerical scheme

$$u_{k+1} = u_k + \tfrac{1}{2}\,h\left[\,F(t_k,u_k) + F(t_{k+1},u_{k+1})\,\right], \tag{10.52}$$

known as the *Trapezoid Method*. It is an *implicit scheme*, since the updated value u_{k+1} appears on both sides of the equation, and hence is only defined implicitly.

Example 10.18. Consider the differential equation $\dot u = \left(1 - \frac{4}{3}\,t\right) u$ studied in Examples 10.14 and 10.15. The Trapezoid Method with a fixed step size h takes the form

$$u_{k+1} = u_k + \tfrac{1}{2}\,h\left[\,\left(1 - \tfrac{4}{3}\,t_k\right)u_k + \left(1 - \tfrac{4}{3}\,t_{k+1}\right)u_{k+1}\,\right].$$

In this case, we can explicit solve for the updated solution value, leading to the recursive formula

$$u_{k+1} = \frac{1 + \frac{1}{2}h\left(1 - \frac{4}{3}t_k\right)}{1 - \frac{1}{2}h\left(1 - \frac{4}{3}t_{k+1}\right)}\, u_k = \frac{1 + \frac{1}{2}h - \frac{2}{3}h\,t_k}{1 - \frac{1}{2}h + \frac{2}{3}h\,(t_k + h)}\, u_k. \tag{10.53}$$

Implementing this scheme for three different step sizes gives the following errors between the computed and true solutions at times $t = 1, 2, 3$.

h	$E(1)$	$E(2)$	$E(3)$
.100	−.00133315	.00060372	−.00012486
.010	−.00001335	.00000602	−.00000124
.001	−.00000013	.00000006	−.00000001

The numerical data indicates that the Trapezoid Method is of second order. For each reduction in step size by $\frac{1}{10}$, the accuracy in the solution increases by, roughly, a factor of $\frac{1}{100} = \frac{1}{10^2}$; that is, the numerical solution acquires two additional accurate decimal digits.

The main difficulty with the Trapezoid Method (and any other implicit scheme) is immediately apparent. The updated approximate value for the solution u_{k+1} appears on both sides of the equation (10.52). Only for very simple functions $F(t, u)$ can one expect to solve (10.52) explicitly for u_{k+1} in terms of the known quantities t_k, u_k and $t_{k+1} = t_k + h$. The alternative is to employ a numerical equation solver, such as the bisection algorithm or Newton's Method, to compute u_{k+1}. In the case of Newton's Method, one would use the current approximation u_k as a first guess for the new approximation u_{k+1} — as in the continuation method discussed in Example 2.20. The resulting scheme requires some work to program, but can be effective in certain situations.

An alternative, less involved strategy is based on the following far-reaching idea. We already know a half-way decent approximation to the solution value u_{k+1} — namely that provided by the more primitive Euler scheme

$$\widetilde{u}_{k+1} = u_k + h\,F(t_k, u_k). \tag{10.54}$$

Let's use this estimated value in place of u_{k+1} on the right hand side of the implicit equation (10.52). The result

$$\begin{aligned} u_{k+1} &= u_k + \tfrac{1}{2}h\left[\,F(t_k, u_k) + F(t_k + h, \widetilde{u}_{k+1})\,\right] \\ &= u_k + \tfrac{1}{2}h\left[\,F(t_k, u_k) + F\big(\,t_k + h, u_k + h\,F(t_k, u_k)\,\big)\,\right]. \end{aligned} \tag{10.55}$$

is known as the *Improved Euler Method*. It is a completely explicit scheme since there is no need to solve any equation to find the updated value u_{k+1}.

Example 10.19. For our favorite equation $\dot{u} = \left(1 - \frac{4}{3}t\right)u$, the Improved Euler Method begins with the Euler approximation

$$\widetilde{u}_{k+1} = u_k + h\left(1 - \tfrac{4}{3}t_k\right)u_k,$$

and then replaces it by the improved value

$$\begin{aligned} u_{k+1} &= u_k + \tfrac{1}{2}h\left[\left(1-\tfrac{4}{3}t_k\right)u_k + \left(1-\tfrac{4}{3}t_{k+1}\right)\widetilde{u}_{k+1}\right] \\ &= u_k + \tfrac{1}{2}h\left[\left(1-\tfrac{4}{3}t_k\right)u_k + \left(1-\tfrac{4}{3}(t_k+h)\right)\left(u_k + h\left(1-\tfrac{4}{3}t_k\right)u_k\right)\right] \\ &= \left[\left(1-\tfrac{2}{3}h^2\right)\left[1+h\left(1-\tfrac{4}{3}t_k\right)\right] + \tfrac{1}{2}h^2\left(1-\tfrac{4}{3}t_k\right)^2\right]u_k. \end{aligned}$$

Implementing this scheme leads to the following errors in the numerical solution at the indicated times. The Improved Euler Method performs comparably to the fully implicit scheme (10.53), and significantly better than the original Euler Method.

h	$E(1)$	$E(2)$	$E(3)$
.100	−.00070230	.00097842	.00147748
.010	−.00000459	.00001068	.00001264
.001	−.00000004	.00000011	.00000012

The Improved Euler Method is the simplest of a large family of so-called *predictor–corrector algorithms*. In general, one begins a relatively crude method — in this case the Euler Method — to *predict* a first approximation $\widetilde{u}_{k+1}$ to the desired solution value u_{k+1}. One then employs a more sophisticated, typically implicit, method to *correct* the original prediction, by replacing the required update u_{k+1} on the right hand side of the implicit scheme by the less accurate prediction $\widetilde{u}_{k+1}$. The resulting explicit, corrected value u_{k+1} will, provided the method has been designed with due care, be an improved approximation to the true solution.

The numerical data in Example 10.19 indicates that the Improved Euler Method is of second order since each reduction in step size by $\frac{1}{10}$ improves the solution accuracy by, roughly, a factor of $\frac{1}{100}$. To verify this prediction, we expand the right hand side of (10.55) in a Taylor series in h, and then compare, term by term, with the solution expansion (10.45). First,

$$F\left(t_k+h, u_k+h\,F(t_k,u_k)\right) = F + h\left(F_t + F\,F_u\right) + \tfrac{1}{2}h^2\left(F_{tt} + 2F\,F_{tu} + F^2\,F_{uu}\right) + \cdots,$$

where all the terms involving F and its partial derivatives on the right hand side are evaluated at t_k, u_k. Substituting into (10.55), we find

$$u_{k+1} = u_k + h\,F + \tfrac{1}{2}h^2\left(F_t + F\,F_u\right) + \tfrac{1}{4}h^3\left(F_{tt} + 2F\,F_{tu} + F^2\,F_{uu}\right) + \cdots. \tag{10.56}$$

The two Taylor expansions (10.45) and (10.56) agree in their order $1, h$ and h^2 terms, but differ at order h^3. This confirms our experimental observation that the Improved Euler Method is of second order.

We can design a range of numerical solution schemes by implementing alternative numerical approximations to the basic integral equation (10.49). For example, the Midpoint

Rule approximates the integral of the function $g(t)$ by the area of the rectangle whose height is the value of the function at the midpoint:

$$\int_{t_k}^{t_{k+1}} g(s)\,ds \;\approx\; h\,g\left(t_k + \tfrac{1}{2}\,h\right), \qquad \text{where} \qquad h = t_{k+1} - t_k. \tag{10.57}$$

See Figure 10.6 for an illustration. The Midpoint Rule is known to have the same order of accuracy as the Trapezoid Rule, [**2**, **7**]. Substituting into (10.49) leads to the approximation

$$u_{k+1} = u_k + \int_{t_k}^{t_{k+1}} F(s, u(s))\,ds \;\approx\; u_k + h\,F\left(t_k + \tfrac{1}{2}\,h, u\left(t_k + \tfrac{1}{2}\,h\right)\right).$$

Of course, we don't know the value of the solution $u\left(t_k + \frac{1}{2}\,h\right)$ at the midpoint, but can predict it through a straightforward adaptation of the basic Euler approximation:

$$u\left(t_k + \tfrac{1}{2}\,h\right) \;\approx\; u_k + \tfrac{1}{2}\,h\,F(t_k, u_k).$$

The result is the *Midpoint Method*

$$u_{k+1} = u_k + h\,F\left(t_k + \tfrac{1}{2}\,h, u_k + \tfrac{1}{2}\,h\,F(t_k, u_k)\right). \tag{10.58}$$

A comparison of the terms in the Taylor expansions of (10.45), (10.58) reveals that the Midpoint Method is also of second order.

Runge–Kutta Methods

The Improved Euler and Midpoint Methods are the most elementary incarnations of a general class of numerical schemes for ordinary differential equations that were first systematically studied by the German mathematicians Carle Runge and Martin Kutta in the late nineteenth century. Runge–Kutta Methods are by far the most popular and powerful general-purpose numerical methods for integrating ordinary differential equations. While not appropriate in all possible situations, Runge–Kutta schemes are surprisingly robust, performing efficiently and accurately in a wide variety of problems. Barring significant complications, they are the method of choice in most basic applications. They comprise the engine that powers most computer software for solving general initial value problems for systems of ordinary differential equations.

The most general *Runge–Kutta Method* takes the form

$$u_{k+1} = u_k + h \sum_{i=1}^{m} c_i\,F(t_{i,k}, u_{i,k}), \tag{10.59}$$

where m counts the number of *terms* in the method. Each $t_{i,k}$ denotes a point in the k^{th} mesh interval, so $t_k \le t_{i,k} \le t_{k+1}$. The second argument $u_{i,k} \approx u(t_{i,k})$ should be viewed as an approximation to the solution at the point $t_{i,k}$, and so is computed by a simpler Runge–Kutta scheme of the same general format. There is a lot of flexibility in the design of the method, through choosing the coefficients c_i, the times $t_{i,k}$, as well as the scheme (and all parameters therein) used to compute each of the intermediate approximations $u_{i,k}$. As always, the *order* of the method is fixed by the power of h to which the Taylor expansions of the numerical method (10.59) and the actual solution (10.45) agree. Clearly, the more terms we include in the Runge–Kutta formula (10.59), the more free parameters available to match terms in the solution's Taylor series, and so the higher the potential order of the method. Thus, the goal is to arrange the parameters so that the method has a high order of accuracy, while, simultaneously, avoiding unduly complicated, and hence computationally costly, formulae.

Both the Improved Euler and Midpoint Methods are instances of a family of two term Runge–Kutta Methods

$$\begin{aligned} u_{k+1} &= u_k + h\left[\, a\,F(t_k,u_k) + b\,F\big(\,t_{k,2}, u_{k,2}\,\big)\,\right] \\ &= u_k + h\left[\, a\,F(t_k,u_k) + b\,F\big(\,t_k + \lambda\,h, u_k + \lambda\,h\,F(t_k,u_k)\,\big)\,\right], \end{aligned} \tag{10.60}$$

based on the current mesh point, so $t_{k,1} = t_k$, and one intermediate point $t_{k,2} = t_k + \lambda\,h$ with $0 \le \lambda \le 1$. The basic Euler Method is used to approximate the solution value

$$u_{k,2} = u_k + \lambda\,h\,F(t_k,u_k)$$

at $t_{k,2}$. The Improved Euler Method sets $a = b = \frac{1}{2}$ and $\lambda = 1$, while the Midpoint Method corresponds to $a = 0$, $b = 1$, $\lambda = \frac{1}{2}$. The range of possible values for a, b and λ is found by matching the Taylor expansion

$$\begin{aligned} u_{k+1} &= u_k + h\left[\, a\,F(t_k,u_k) + b\,F\big(\,t_k + \lambda\,h, u_k + \lambda\,h\,F(t_k,u_k)\,\big)\,\right] \\ &= u_k + h\,(a+b)\,F(t_k,u_k) + h^2\,b\,\lambda\left[\frac{\partial F}{\partial t}\,(t_k,u_k) + F(t_k,u_k)\,\frac{\partial F}{\partial u}\,(t_k,u_k)\right] + \;\cdots\;. \end{aligned}$$

(in powers of h) of the right hand side of (10.60) with the Taylor expansion (10.45) of the solution, namely

$$u(t_{k+1}) = u_k + h\,F(t_k,u_k) + \frac{h^2}{2}\,\left[\,F_t(t_k,u_k) + F(t_k,u_k)\,F_u(t_k,u_k)\,\right] + \;\cdots\;,$$

to as high an order as possible. First, the constant terms, u_k, are the same. For the order h and order h^2 terms to agree, we must have, respectively,

$$a + b = 1, \qquad\qquad b\,\lambda = \tfrac{1}{2}.$$

Therefore, setting

$$a = 1 - \mu, \qquad b = \mu, \qquad \text{and} \qquad \lambda = \frac{1}{2\,\mu}\,, \qquad \text{where } \mu \text{ is arbitrary}^{\dagger},$$

leads to the following family of two term, second order Runge–Kutta Methods:

$$u_{k+1} = u_k + h\left[\,(1-\mu)\,F(t_k,u_k) + \mu\,F\left(\,t_k + \frac{h}{2\,\mu}, u_k + \frac{h}{2\,\mu}\,F(t_k,u_k)\right)\right]. \tag{10.61}$$

† Although we should restrict $\mu \ge \frac{1}{2}$ in order that $0 \le \lambda \le 1$.

The case $\mu = \frac{1}{2}$ corresponds to the Improved Euler Method (10.55), while $\mu = 1$ yields the Midpoint Method (10.58). Unfortunately, none of these methods are able to match all of the third order terms in the Taylor expansion for the solution, and so we are left with a one-parameter family of two step Runge–Kutta Methods, all of second order, that include the Improved Euler and Midpoint schemes as particular instances. The methods with $\frac{1}{2} \leq \mu \leq 1$ all perform more or less comparably, and there is no special reason to prefer one over the other.

To construct a third order Runge–Kutta Method, we need to take at least $m \geq 3$ terms in (10.59). A rather intricate computation (best done with the aid of computer algebra) will produce a range of valid schemes; the results can be found in [**21**, **28**]. The algebraic manipulations are rather tedious, and we leave a complete discussion of the available options to a more advanced treatment. In practical applications, a particularly simple fourth order, four term formula has become the most used. The method, often abbreviated as RK4, takes the form

$$u_{k+1} = u_k + \frac{h}{6}\left[\, F(t_k, u_k) + 2\,F(t_{2,k}, u_{2,k}) + 2\,F(t_{3,k}, u_{3,k}) + F(t_{4,k}, u_{4,k})\,\right], \tag{10.62}$$

where the times and function values are successively computed according to the following procedure:

$$\begin{aligned} t_{2,k} &= t_k + \tfrac{1}{2}\,h, \qquad & u_{2,k} &= u_k + \tfrac{1}{2}\,h\,F(t_k, u_k), \\ t_{3,k} &= t_k + \tfrac{1}{2}\,h, \qquad & u_{3,k} &= u_k + \tfrac{1}{2}\,h\,F(t_{2,k}, u_{2,k}), \\ t_{4,k} &= t_k + h, \qquad & u_{4,k} &= u_k + h\,F(t_{3,k}, u_{3,k}). \end{aligned} \tag{10.63}$$

The four term RK4 scheme (10.62–63) is, in fact, a fourth order method. This is confirmed by demonstrating that the Taylor expansion of the right hand side of (10.62) in powers of h matches all of the terms in the Taylor series for the solution (10.45) up to and including those of order h^4, and hence the local error is of order h^5. This is not a computation for the faint-hearted — bring lots of paper and erasers, or, better yet, a good computer algebra package! The RK4 scheme is but one instance of a large family of fourth order, four term Runge–Kutta Methods, and by far the most popular owing to its relative simplicity.

Example 10.20. Application of the RK4 Method (10.62–63) to our favorite initial value problem (10.37) leads to the following errors at the indicated times:

h	$E(1)$	$E(2)$	$E(3)$
.100	-1.944×10^{-7}	1.086×10^{-6}	4.592×10^{-6}
.010	-1.508×10^{-11}	1.093×10^{-10}	3.851×10^{-10}
.001	-1.332×10^{-15}	-4.741×10^{-14}	1.932×10^{-14}

The accuracy is phenomenally good — much better than any of our earlier numerical schemes. Each decrease in the step size by a factor of $\frac{1}{10}$ results in about 4 additional decimal digits of accuracy in the computed solution, in complete accordance with its status as a fourth order method.

Actually, it is not entirely fair to compare the accuracy of the methods using the same step size. Each iteration of the RK4 Method requires four evaluations of the function $F(t,u)$, and hence takes the same computational effort as four Euler iterations, or, equivalently, two Improved Euler iterations. Thus, the more revealing comparison would be between RK4 at step size h, Euler at step size $\frac{1}{4}h$, and Improved Euler at step size $\frac{1}{2}h$, as these involve roughly the same amount of computational effort. The resulting errors $E(1)$ at time $t=1$ are listed in the following table.

Thus, even taking computational effort into account, the Runge–Kutta Method continues to outperform its rivals. At a step size of .1, it is almost as accurate as the Improved Euler Method with step size .0005, and hence 200 times as much computation, while the Euler Method would require a step size of approximately $.24\times 10^{-6}$, and would be $4,000,000$ times as slow as Runge–Kutta! With a step size of .001, RK4 computes a solution value that is near the limits imposed by machine accuracy (in single precision arithmetic). The superb performance level and accuracy of the RK4 Method immediately explains its popularity for a broad range of applications.

h	Euler	Improved Euler	Runge–Kutta 4
.1	1.872×10^{-2}	-1.424×10^{-4}	-1.944×10^{-7}
.01	1.874×10^{-3}	-1.112×10^{-6}	-1.508×10^{-11}
.001	1.870×10^{-4}	-1.080×10^{-8}	-1.332×10^{-15}

Example 10.21. As noted earlier, by writing the function values as vectors $\mathbf{u}_k \approx \mathbf{u}(t_k)$, one can immediately use all of the preceding methods to integrate initial value problems for first order systems of ordinary differential equations $\dot{\mathbf{u}} = \mathbf{F}(\mathbf{u})$. Consider, by way of example, the Lotka–Volterra system

$$\frac{du}{dt} = 2u - uv, \qquad \frac{dv}{dt} = -9v + 3uv. \tag{10.64}$$

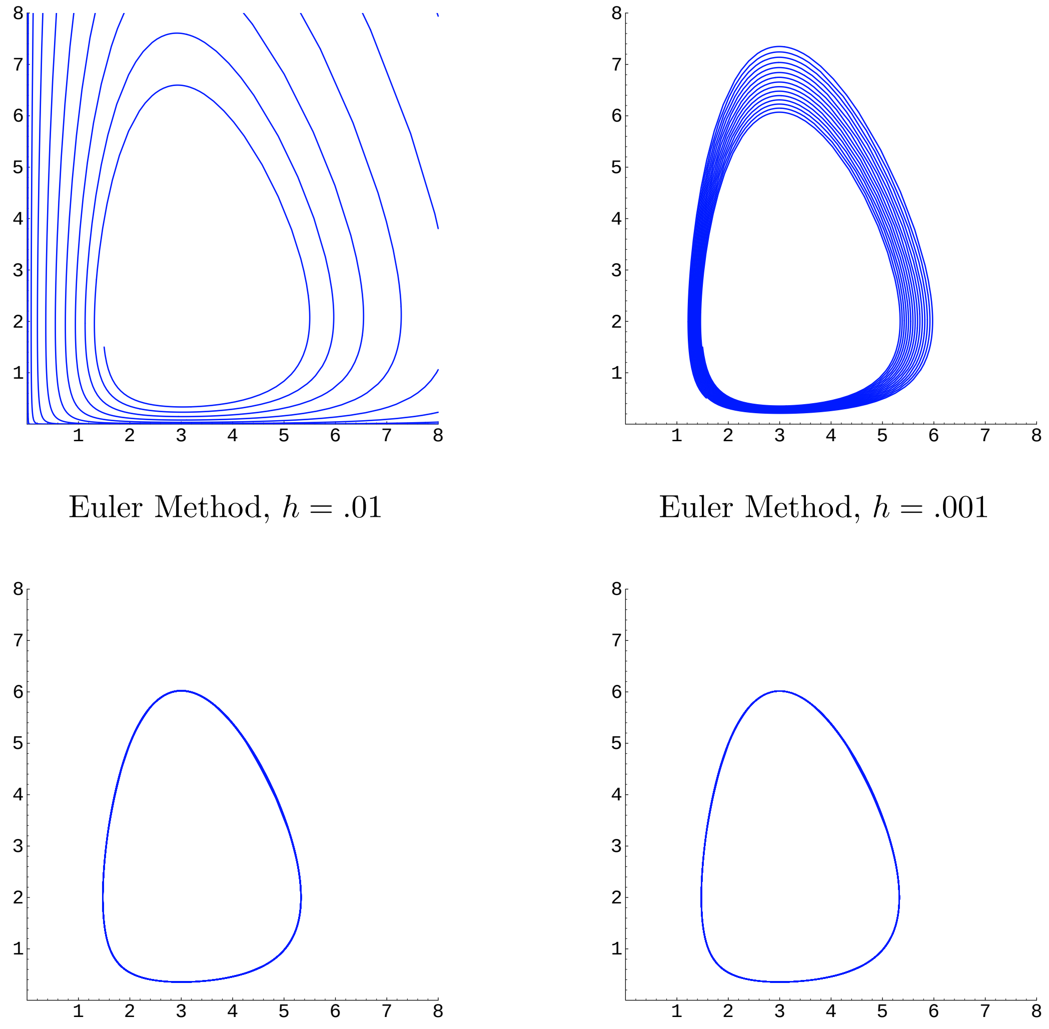
To find a numerical solution, we write $\mathbf{u} = (u,v)^T$ for the solution vector, while $\mathbf{F}(\mathbf{u}) = (2u-uv, -9v+3uv)^T$ is the right hand side of the system. The Euler Method with step size h is given by

$$\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + h\,\mathbf{F}(\mathbf{u}^{(k)}),$$

or, explicitly, as a first order nonlinear iterative system

$$u^{(k+1)} = u^{(k)} + h\,(2u^{(k)} - u^{(k)}\,v^{(k)}), \qquad v^{(k+1)} = v^{(k)} + h\,(-9v^{(k)} + 3u^{(k)}\,v^{(k)}).$$

The Improved Euler and Runge–Kutta schemes are implemented in a similar fashion. Phase portraits of the three numerical algorithms starting with initial conditions $u^{(0)} = v^{(0)} = 1.5$, and up to time $t=25$ in the case of the Euler Method, and $t=50$ for the other



Euler Method, $h = .01$ Euler Method, $h = .001$

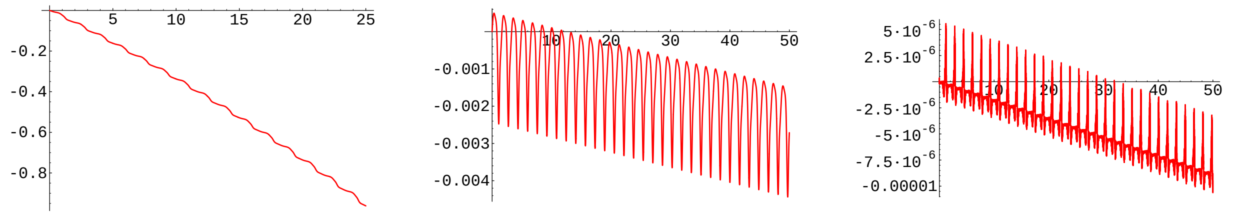
Improved Euler Method, $h = .01$ RK4 Method, $h = .01$

Figure 10.7. Numerical Solutions of Predator–Prey Model.

two, appear in Figure 10.7. In fact, the solution is supposed to travel periodically around a closed curve, which is the level set

$$I(u,v) = 9\,\log u - 3\,u + 2\,\log v - v = I(1.5,1.5) = -1.53988$$

of the first integral. The Euler Method spirals away from the exact periodic solution, whereas the Improved Euler and RK4 Methods perform rather well. Since we do not have an analytic formula for the solution, we cannot measure the exact error in the methods. However, the known first integral is supposed to remain constant on the solution trajectories, and so one means of monitoring the accuracy of the solution is by the variation in the numerical values of $I(u^{(k)}, v^{(k)})$. These are graphed in Figure 10.8; the Improved Euler keeps the vlue within .0005, while for the RK4 solution, only the fifth decimal place in the value of the first integral experiences any change over the indicated time period. Of course, the ononger one continues to integrate, the more error will gradually creep into the



Euler Method $h = .001$ Improved Euler Method $h = .01$ RK4 Method $h = .01$

Figure 10.8. Numerical Evaluation of Lotka–Volterra First Integral.

numerical solution. Still, for most practical purposes, the RK4 solution is indistinguishable from the exact solution.

In practical implementations, it is important to monitor the accuracy of the numerical solution, so to gauge when to abandon an insufficiently precise computation. Since accuracy is dependent upon the step size h, one may try adjusting h so as stay within a preassigned error. *Adaptive methods*, allow one to change the step size during the course of the computation, in response to some estimation of the overall error. Insufficiently accurate numerical solutions would necessitate a suitable reduction in step size (or increase in the order of the scheme). On the other hand, if the solution is more accurate than the application requires, one could increase the step size so as to reduce the total amount of computational effort.

How might one decide when a method is giving inaccurate results, since one presumably does not know the true solution and so has nothing to directly test the numerical approximation against? One useful idea is to integrate the differential equation using two different numerical schemes, usually of different orders of accuracy, and then compare the results. If the two solution values are reasonably close, then one is usually safe in assuming that the methods are both giving accurate results, while in the event that they differ beyond some preassigned tolerance, then one needs to re-evaluate the step size. The required adjustment to the step size relies on a more detailed analysis of the error terms. Several well-studied methods are employed in practical situations; the most popular is the Runge–Kutta–Fehlberg Method, which combines a fourth and a fifth order Runge–Kutta scheme for error control. Details can be found in more advanced treatments of the subject, e.g., [**21**, **28**].

Stiff Differential Equations

While the fourth order Runge–Kutta Method with a sufficiently small step size will successfully integrate a broad range of differential equations — at least over not unduly long time intervals — it does occasionally experience unexpected difficulties. While we have not developed sufficiently sophisticated analytical tools to conduct a thorough analysis, it will be instructive to look at why a breakdown might occur in a simpler context.

Example 10.22. The elementary linear initial value problem

$$\frac{du}{dt} = -250\,u, \qquad u(0) = 1, \tag{10.65}$$

is an instructive and sobering example. The explicit solution is easily found; it is a very rapidly decreasing exponential: $u(t) = e^{-250t}$.

$$u(t) = e^{-250t} \qquad \text{with} \qquad u(1) \approx 2.69 \times 10^{-109}.$$

The following table gives the result of approximating the solution value $u(1) \approx 2.69 \times 10^{-109}$ at time $t = 1$ using three of our numerical integration schemes for various step sizes:

h	Euler	Improved Euler	RK4
.1	6.34×10^{13}	3.99×10^{24}	2.81×10^{41}
.01	4.07×10^{17}	1.22×10^{21}	1.53×10^{-19}
.001	1.15×10^{-125}	6.17×10^{-108}	2.69×10^{-109}

The results are not misprints! When the step size is .1, the computed solution values are perplexingly large, and appear to represent an exponentially growing solution — the complete opposite of the rapidly decaying true solution. Reducing the step size beyond a critical threshold suddenly transforms the numerical solution to an exponentially decaying function. Only the fourth order RK4 Method with step size $h = .001$ — and hence a total of $1,000$ steps — does a reasonable job at approximating the correct value of the solution at $t = 1$.

You may well ask, what on earth is going on? The solution couldn't be simpler — why is it so difficult to compute it? To understand the basic issue, let us analyze how the Euler Method handles such simple differential equations. Consider the initial value problem

$$\frac{du}{dt} = \lambda\, u, \qquad u(0) = 1, \tag{10.66}$$

which has an exponential solution

$$u(t) = e^{\lambda t}. \tag{10.67}$$

As in Example 10.13, the Euler Method with step size h relies on the iterative scheme

$$u_{k+1} = (1 + \lambda\, h)\, u_k, \qquad u_0 = 1,$$

with solution

$$u_k = (1 + \lambda\, h)^k. \tag{10.68}$$

If $\lambda > 0$, the exact solution (10.67) is exponentially growing. Since $1 + \lambda\, h > 1$, the numerical iterates are also growing, albeit at a somewhat slower rate. In this case, there is no inherent surprise with the numerical approximation procedure — in the short run it gives fairly accurate results, but eventually trails behind the exponentially growing solution.

On the other hand, if $\lambda < 0$, then the exact solution is exponentially decaying and positive. But now, if $\lambda\, h < -2$, then $1 + \lambda\, h < -1$, and the iterates (10.68) grow exponentially fast in magnitude, with alternating signs. In this case, the numerical solution is nowhere close to the true solution; this explains the previously observed pathological behavior. If $-1 < 1 + \lambda h < 0$, the numerical solutions decay in magnitude, but continue to alternate between positive and negative values. Thus, to correctly model the qualitative features of the solution and obtain a numerically respectable approximation, we need to choose the step size h so as to ensure that $0 < 1 + \lambda h$, and hence $h < -1/\lambda$ when $\lambda < 0$. For the value $\lambda = -250$ in the example, then, we must choose $h < \frac{1}{250} = .004$ in order that the Euler Method give a reasonable numerical answer. A similar, but more complicated analysis applies to any of the Runge–Kutta schemes.

Thus, the numerical methods for ordinary differential equations exhibit a form of conditional stability. Paradoxically, the larger negative λ is — and hence the faster the solution tends to a trivial zero equilibrium — the *more* difficult and expensive the numerical integration. The system (10.65) is the simplest example of what is known as a *stiff differential equation*. In general, an equation or system is stiff if it has one or more very rapidly decaying solutions. In the case of autonomous (constant coefficient) linear systems $\dot{\mathbf{u}} = A\,\mathbf{u}$, stiffness occurs whenever the coefficient matrix A has an eigenvalue with a large negative real part: $\operatorname{Re}\lambda \ll 0$, resulting in a very rapidly decaying eigensolution. It only takes one such eigensolution to render the equation stiff, and ruin the numerical computation of even the well behaved solutions! Curiously, the component of the actual solution corresponding to such large negative eigenvalues is almost irrelevant, as it becomes almost instanteously tiny. However, the presence of such an eigenvalue continues to render the numerical solution to the system very difficult, even to the point of exhausting any available computing resources. Stiff equations require more sophisticated numerical procedures to integrate, and we refer the reader to [**21**, **28**] for details.