Chapter 7

Ordinary Differential Equations

In this chapter we will

- explore the difference between explicit and implicit numerical methods
- study the stability of numerical integrators
- learn how to solve algebraic equations using
  - Euler's method
  - Runge-Kutta methods

An equation that involves derivatives of the dependent variable is called a **differential equation**. Most problems in the physical sciences involve the solution of differential equations. In fact, the calculus of differential equations was invented by Newton and Leibnitz in order to describe the evolution of physical systems.

If a differential equation involves only one independent variable and, therefore, contains only full derivatives, it is called an **ordinary differential equation (ODE)** . If it involves more than one independent variables and, therefore, contains partial derivatives, it is a called a **partial differential equation (PDE)**. The order of a differential equation is the order of the highest derivative that appears in the equation.

There are many well known differential equations in Physics. For example, Newton's second law

$$\frac{d^2\vec{r}}{dt^2} = \frac{1}{m}\vec{F}(\vec{r}) \ , \tag{7.1}$$

which connects the acceleration $d^2\vec{r}/dt^2$ of a particle of mass m and the force $\vec{F}(\vec{r})$, is a second-order, ordinary differential equation. On the other hand, the one-dimensional Schrödinger's equation

$$-i\hbar\frac{\partial\Psi(x,t)}{\partial t} = -\frac{\hbar^2}{2m}\frac{\partial^2\Psi(x,t)}{\partial\chi^2} + V(x)\Psi(x,t) \ , \tag{7.2}$$

which describes the time evolution of a wavefunction $\Psi(x,t)$ of a particle in a potential $V(x)$, is a second-order partial differential equation. In this chapter, we will discuss methods for solving ordinary differential equations.

A differential equation of order N is typically equivalent to a system of N equations of first order. For example, we can rewrite Newton's second law (equation [7.1]) as the system of two first-order equations

$$\begin{aligned} \frac{d\vec{r}}{dt} &= \vec{v} \\ \frac{d\vec{v}}{dt} &= \frac{1}{m}\vec{F}(\vec{r}) \ , \end{aligned} \tag{7.3}$$

by introducing the particle velocity $\vec{v}$ as an additional dependent variable. Because of such decompositions, we will develop here algorithms that solve a system of N, first-order, ordinary differential equations.

The solution to a differential equation depends not only on the equation itself, but on the boundary conditions as well. In fact, it is the boundary conditions that pick one, many, or even none of all the possible solutions to a differential equation. Furthermore, the number, location, and character of the boundary conditions affect considerably the choice of numerical method used. In a typical problem, for each differential equation of order N, there are N boundary conditions required, each of which has to be of order at most $N-1$. In a so-called **initial value problem**, all boundary conditions are given at the same place in the domain of solution, i.e., for the same value of the independent parameter. In a **boundary-value problem**, different boundary conditions are given at different places in the domain of solution.

As an example, Newton's second law (equation [7.1]) is a second-order differential equation. It, therefore, requires two boundary conditions. Each of the boundary conditions can be at most of order one, i.e., it can be either a position or a velocity at a given time. In an initial-value problem, for example, the position and velocity of a particle are given at time t_0 and then the differential equation is solved forward in time. In a boundary value problem, the initial and final position of a particle may be given. In physical situations, initial value problems almost always have unique solutions. On the other hand, boundary value problems may have none, one, or even many solutions. In this chapter, we will discuss only methods for solving a system of first order, ordinary differential equations for initial-value problems.

7.1 Explicit and Implicit Methods

Solving ordinary differential equations is conceptually identical to integrating a function. In fact, for an equation of the form

$$\frac{df}{dt} = g(t) \ , \tag{7.4}$$

in which the right-hand side depends only on the independent variable and not on the function itself, the solution is simply

$$f(t) = f(t_0) + \int_{t_0}^{t} g(t)dt \ , \tag{7.5}$$

where $f(t_0)$ is the boundary condition. In general, however, the right-hand side of a differential equation will depend on both the dependent and the independent variables, i.e.,

$$\frac{df}{dt} = g[t, f(t)] \ . \tag{7.6}$$

In this general case, separation of variables is impossible and ordinary integration methods are not useful in solving differential equations.

In a manner similar to all other discrete numerical methods, solving a continuous differential equation will involve taking a sequence of very small steps in the independent variable and calculating approximately the evolution of the dependent variable at the end of each step. For a well behaving function $f(t)$ that obeys a differential equation of the form (7.6), we can calculate its evolution after a time step Δt by a Taylor expansions, i.e.,

$$f(t+\Delta t) = f(t) + \left.\frac{df}{dt}\right|_t \Delta t + \frac{1}{2}\left.\frac{d^2 f}{dt^2}\right|_t (\Delta t)^2 + \frac{1}{3!}\left.\frac{d^3 f}{dt^3}\right|_t (\Delta t)^3 + ... \qquad (7.7)$$

Ignoring terms of second or higher order in Δt and using the definition of $g[t, f(t)]$ we obtain the alebgraic equation

$$\boxed{\frac{f(t+\Delta t) - f(t)}{\Delta t} = g[t, f(t)]} \qquad (7.8)$$

Euler's Method

that allows us to evolve the function $f(t)$ by a time increment Δt. This algebraic equation is called a **difference equation** and is mathematically equivalent to the original differential equation when we take the limit $\Delta t \to 0$. The particular approximation of the differential operator we derived above is called **Euler's method** and is equivalent to the rectangle rule of integration. The error we introduce at each timestep with Euler's method scales as $(\Delta t)^2$.

Euler's method is an **explicit**, first order method for solving differential equations. It is an explicit method because calculating the value of the function $f(t+\Delta t)$ at the end of each time step Δt depends only explicitly on the function and its derivative evaluated at the beginning of the timestep, i.e., it depends only on $f(t)$ and on $g[t, f(t)]$ (see eq. [7.8]). Moreover, it is a first order method, because it is accurate up to the first order in Δt; the error introduced at each time step scales as the second power of the Δt.

The above method can be improved by following a procedure reminiscent of the trapezoid rule of integration. We can obtain another approximation to the value of the function $f(t+\Delta t)$ at the end of each timestep by Taylor expanding the function backwards from $t + \Delta t$ to t, i.e.,

$$f(t) = f(t+\Delta t) - \left.\frac{df}{dt}\right|_{t+\Delta t} \Delta t + \frac{1}{2}\left.\frac{d^2 f}{dt^2}\right|_{t+\Delta t} (\Delta t)^2 - \frac{1}{3!}\left.\frac{d^3 f}{dt^3}\right|_{t+\Delta t} (\Delta t)^3 ... \qquad (7.9)$$

Subtracting equation (7.9) from equation (7.7) and rearranging some terms, we obtain

$$f(t+\Delta t) = f(t) + \frac{1}{2}\left\{g[t, f(t)] + g[t+\Delta t, f(t+\Delta t)]\right\} \Delta t \; . \qquad (7.10)$$

As in the case of trapezoid integration, the leading term that we neglected in equation (7.10) is of third order. This is an **implicit**, second order method of solving a differential equation. Is is implicit because the value of the function $f(t+\Delta t)$ at the end of each time step is calculated by approximation (7.10) that depends implicitly on the value of the function at the end of the timestep. It is also a second order method, because the error introduced at each time step scales as the third power of Δt, as in the case of the trapezoid rule.

7.2 Accuracy of ODE solvers

The previous discussion suggests that the accuracy of a numerical method for solving differential equations increases as the timestep used decreases. However, for

very small values of the timestep, the dependent variable does not change significantly between timesteps and hence round-off errors affect the numerical solution. Indeed, for every case there is an optimal timestep at which the fractional error between the numerical and the analytic solution is minimized. In this section, we study the accuracy of differential equation solvers and its dependence on the various parameters of the problem.

For large values of the timestep, the systematic error of a numerical scheme always dominates. If we use a scheme of n-th order, then the error we introduce at every timestep between the numerical approximation (e.g., eq. [7.8] or eq. [7.10]), is dominated by the term of order $n+1$ of the Taylor expansion around the initial value, i.e.,

$$f - f_{\rm num} = \frac{1}{(n+1)!} \left. \frac{d^{n+1} f}{dt^{n+1}} \right|_{\rm t} (\Delta t)^{n+1} + ... \, , \tag{7.11}$$

where we used $f_{\rm num}$ and f to denote the numerical and true solutions, respectively. The compound error after evolving the equation for a time t is approximately equal to the product of the error per timestep (eq. [7.11]) times the total number of timesteps taken $N = t/\Delta t$. As a result, after many timesteps, the compound error is

$$f - f_{\rm num} \simeq \frac{1}{(n+1)!} \left. \frac{d^{n+1} f}{dt^{n+1}} \right|_{\rm t} (\Delta t)^{n} t \, , \tag{7.12}$$

Clearly, the compound error depends on the problem under study (through the derivative term), on the order of the numerical scheme employed, on the size of the timestep taken, and on the total time over which the differential equation is evolved.

For small values of the timestep, it is the round-off error that dominates the overall accuracy of the numerical solution. If we denote by ϵ the accuracy with which a real number is stored in the computer, then the round-off error after each timestep is about $\epsilon/2$. The compound error after taking $N = t/\Delta t$ timesteps is approximately proportional to $\sqrt{N}$. Therefore, for small values of the timestep, the compound error is

$$f - f_{\rm num} \simeq \frac{\epsilon}{2} \left(\frac{t}{\Delta t} \right)^{1/2} \, , \tag{7.13}$$

At this limit, the error depends on the machine accuracy, ϵ, the timestep, and the total time over which the differential equation is evolved.

The systematic error introduced by the numerical approximation of the algorithm decreases with decreasing size of the timestep (equation 7.24). On the other hand, the effect of the round-off error increases with decreasing size of the timestep (equation 7.25). As a result, there is an optimal size of the timestep at which the combined error is minimal. Equating relations (7.24) and (7.25) we obtain

$$(\Delta t)_{\rm opt} \simeq \left[\frac{\epsilon (n+1)!}{2 t^{1/2}} \left(\left. \frac{d^{n+1} f}{dt^{n+1}} \right|_{\rm t} \right)^{-1} \right]^{1/(n+1/2)} \tag{7.14}$$

for the optimal size of the timestep. If we have expressed the differential equation in a dimensionless form in terms of the natural units of the problem, then both the function f and its derivatives will be of order unity. Therefore, an approximate value for the optimal timestep after one characteristic timescale in the problem is

$$(\Delta t)_{\rm opt} \simeq \left[\frac{\epsilon (n+1)!}{2} \right]^{1/(n+1/2)} \, , \tag{7.15}$$

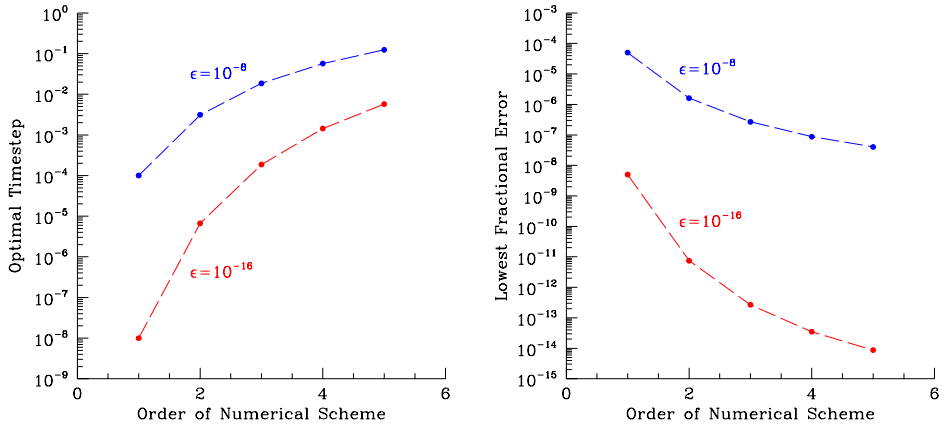

Figure 7.1: The optimal timestep *(left)* and the lowest possible fractional error *(right)* of a numerical solution of a differential equation at $t = 1$, as a function of the order of the scheme used and the machine accuracy. In this plot we have assumed that the differential equation has been expressed in terms of the natural units of the problem.

which depends strongly on the machine accuracy, ϵ, and the order, n, of the numerical scheme used (see Fig. 7.1).

This result might appear counter intuitive in two ways. First, the optimal timestep in order for round-off errors to be avoided is significantly larger than the machine accuracy. For a first-order scheme it is $\sim \epsilon^{2/3}$ and it asymptotes towards unity for higher orders. This is, of course, a consequence of the fact that the round-off error competes with the systematic error of the numerical scheme and the latter scales as $(\Delta t)^{n+1}$. Second, and more important, is the fact that the optimal timestep increases rapidly with the order of the numerical scheme. Indeed, we can achieve a more accurate solution to a problem by increasing the order of the numerical scheme, if and only if we also increase the size of the timestep. If we were to always use the optimal timestep for a first-order scheme to solve numerically a differential equation, the accuracy of the solution would always be the same, independent of the order of the numerical scheme we employed.

We can, in fact, estimate the lowest possible error in our numerical solution, by evaluating the sum of equations (7.24) and (7.25) at the optimal timestep (7.15). The result for the fractional error is (see also the right panel of Fig. 7.1)

$$\begin{aligned} \frac{f - f_{\rm num}}{f} &\simeq \left(\frac{\epsilon}{2}\right)^{n/(n+1)} \left(\frac{1}{(n+1)!f} \left.\frac{d^{n+1}f}{dt^{n+1}}\right|_t \right)^{1/(n+1)} t \\ &\simeq \left(\frac{\epsilon}{2}\right)^{n/(n+1)} \left[\frac{1}{(n+1)!}\right]^{1/(n+1)} t \end{aligned} \tag{7.16}$$

It is clear from this last expression that the error we introduce to a numerical solution of an ordinary differential equation depends on a number of factors. First, it depends on the particular equation being solved; the accuracy is always better for solutions that depend very weakly on the independent variable. Second, it depends on the order of the method used. In general, the higher the order of the scheme, the more accurate the result. This is not always true, however, and especially if we do not use the optimal timestep for every scheme. Third, the accuracy of a numerical solution depends on the precision of the floating point arithmetic used, since the latter sets the roundoff error introduced at every timestep. Finally, the accuracy of the final result depends on the total number of timesteps taken.

We will illustrate the effect of all these factors on the accuracy of a numerical

solution of an ordinary differential equation with the following application.

7.2.1 Application: Carbon dating

The decay of the isotope ^{14}C of carbon is used in dating carbonaceous materials that are younger than about 60,000 years of age. The method is based on the fact that ^{14}C is unstable and beta decays into ^{14}N, an electron, and an antineutrino, according to the reaction

$$^{14}C \rightarrow^{15} N + e^- + \bar{\nu}_e \ . \tag{7.17}$$

The halftime of this reaction is equal to $t_{1/2} = 5730 \pm 40$ years The equation that describes the decay of ^{14}C since the time of disintegration of the material, at which point replenishment of ^{14}C stopped, is

$$\frac{df_{\rm C}}{dt} = -\frac{\ln 2}{t_{1/2}} f_{\rm C} \ , \tag{7.18}$$

where $f_{\rm C}$ is the concentration of ^{14}C. The solution to this differential equation gives the concentration of ^{14}C as a function of the initial concentration f_0 and time t since disintegration. Measuring the current concentration of ^{14}C and inverting this relation leads to an absolute and accurate dating of the material.

We will use this problem, for which an analytic solution exists, in order to study the various factors that affect the accuracy of a numerical solution to a differential equation.

Converting Equations to Dimensionless Form.— In this problem, there is one dependent variable, $f_{\rm C}$, which is dimensionless. On the other hand, the independent variable t as well as the single parameter, $t_{1/2}$, have units of time. We define a new dimensionless independent variable

$$\tau \equiv \frac{\ln 2}{t_{1/2}} t \tag{7.19}$$

and a normalized dependent variable

$$f \equiv \frac{f_{\rm C}}{f_0} \ , \tag{7.20}$$

so that we can rewrite the differential equation in dimensionless form as

$$\frac{df}{d\tau} = -f \ , \tag{7.21}$$

with an initial condition $f(\tau = 0) = 1$.

Analytic Solution.— We can solve easily the differential equation (7.21) using the method of separation of variables, i.e.,

$$\begin{aligned} \int_1^f \frac{df}{f} &= -\int_0^\tau \tau \Rightarrow \\ f(\tau) &= e^{-\tau} \ . \end{aligned} \tag{7.22}$$

In dimensional form, the solution is

$$f_{\rm C}(t) = f_0 e^{-\ln 2 t/t_{1/2}} \ . \tag{7.23}$$

This is the analytic solution against which we will verify our numerical scheme.

Numerical Solution.— We solve the differential equation (7.21) both using the first-order explicit Euler's method and the second-order implicit method discussed

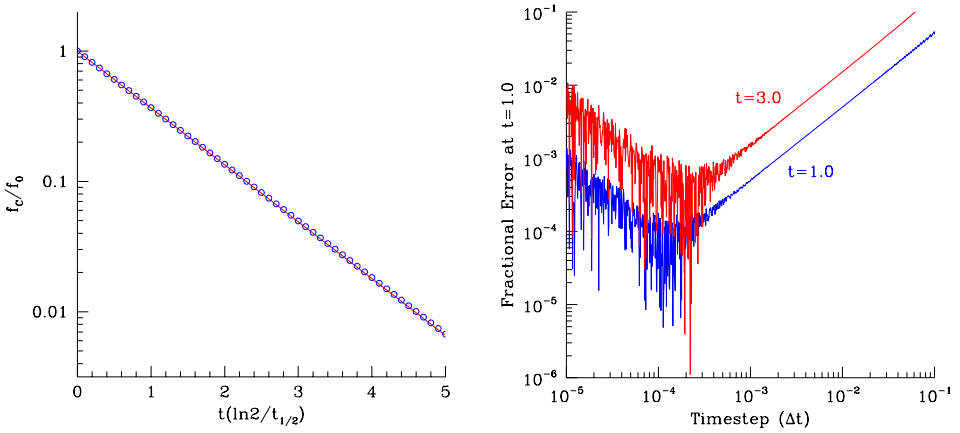

Figure 7.2: *(Left)* The analytic solution (red line) and the numerical solution (blue circles) evaluated every tenth timestep, for the carbon dating problem discussed in §7.2. For this calculation, the timestep is set to $\Delta t = 0.01$. *(Right)* The fractional error between the analytic and numerical solution for the same problem as a function of the timestep Δt, evaluated at two different times during the integration.

earlier. We will not discuss in detail the implementation of either method here, because they are both surpassed in quality by the superior Runge-Kutta methods that we will study later in this chapter.

Accuracy of the Numerical Solutions.—The left panel of figure 7.2 compares the analytic (solid line) to the numerical (open circles) solution of the differential equation obtained using Euler's method with a timestep $\Delta\tau = 0.01$. It appears, at least visually, that the numerical solution agrees very well with the analytic solution. This is demonstrated quantitatively in the right panel of figure 7.2 that shows the fractional error at time $\tau = 1$ (blue line) and at time $\tau = 3$ (red line) for the same calculation, as a function of the timestep Δt.

For large values of the timestep, the systematic error decreases linearly with decreasing timestep but increases linearly with the total time over which the differential equation was evolved. This is in agreement with our previous estimate (7.24), which applied to the current problem gives

$$\frac{f - f_{\rm num}}{f} \simeq \frac{1}{2}(\Delta\tau)\tau \ , \qquad \text{for large } \Delta t \ . \tag{7.24}$$

On the other hand, for small values of the timestep, the error increases again with decreasing timestep as well as with increasing total evolution time. Indeed, according to equation (7.25) the fractional round-off error is approximately equal to

$$\frac{f - f_{\rm num}}{f} \simeq \frac{\epsilon}{2}\left(\frac{\tau}{\Delta\tau}\right) e^{\tau} \ , \qquad \text{for small } \Delta t \ . \tag{7.25}$$

The optimal timestep is $(\Delta\tau)_{\rm opt} \simeq 10^{-4}$ and the lowest possible fractional error in the solution is $\simeq 10^{-4}t$.

We can improve the accuracy of the solution either by increasing the order of the numerical scheme or by increasing the accuracy of the floating point arithmetic used. In either case, however, we achieve an improved accuracy if and only if we also change the size of the timestep appropriately. This is shown in Figure 7.3. In the left panel, the accuracy of the first-order Eulerian scheme is compared to that of the second-order implicit scheme. In the right panel, the accuracy of the numerical solution is shown, when single- and double-precision floating point arithmetic is used. Increasing the order of the numerical scheme requires an increase in the size

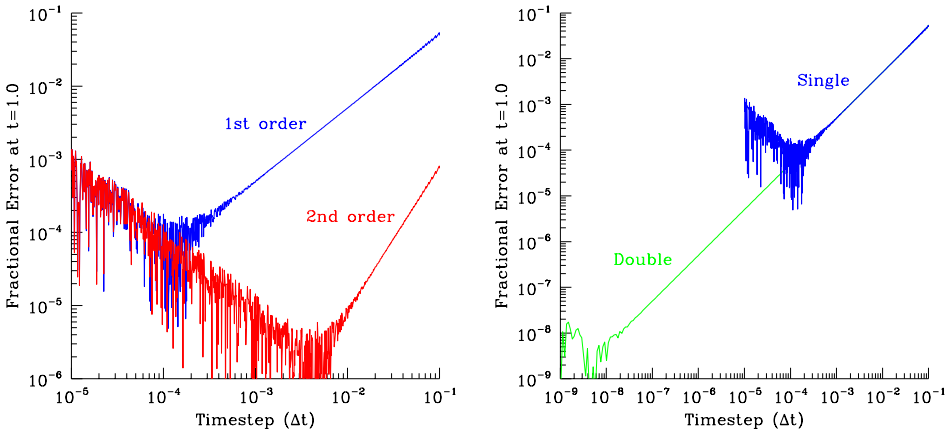

Figure 7.3: The dependence on the size of the timestep of the fractional error after one e-folding time (i.e., at $\tau = 1$) for the problem discussed in §7.2. The left panel compares a first-order Euler scheme with a second-order implicit scheme. The right panel compares two first-order Euler schemes in which floating point arithmetic of single and double precission was used. In both cases, an improvement in the accuracy of the numerical solution is achieved if and only if the size of the timestep is also changed accordingly.

of the timestep, while increasing the accuracy of floating point arithmetic requires a decrease in the size of the timestep.

7.3 Stability of ODE solvers

In the previous section we showed how considerations of the accuracy of a numerical scheme lead to a minimum value of the timestep, below which round-off errors dominate the systematic uncertainty of the method. In this section, we will discuss how the stability of an explicit numerical scheme sets a maximum value of the timestep, above which the method diverges and leads to unphysical results.

We will use again, as an example, the numerical solution to the differential equation for the carbon dating problem (eq. 7.21)

$$\frac{df}{d\tau} = -f \; . \tag{7.26}$$

Using Euler's scheme (eq. [7.8]) to evolve this equation by $\Delta\tau$ away from some initial condition f_0, we obtain

$$f(\Delta\tau) = f_0 + \left.\frac{df}{d\tau}\right|_{\tau=0} \Delta\tau \tag{7.27}$$

$$= (1 - \Delta\tau) f_0 \; . \tag{7.28}$$

After having taken N timesteps, the numerical solution becomes

$$f(N \cdot \Delta\tau) = (1 - \Delta\tau)^N f_0 \; . \tag{7.29}$$

Clearly, the numerical solution has the correct asymptotic behavior

$$\lim_{N\to\infty} f(N \cdot \Delta\tau) = 0 \; , \tag{7.30}$$

if and only if $1 - \delta\tau > -1$, i.e., if and only if

$$\Delta\tau < 2 \; . \tag{7.31}$$

For larger values of the timestep, the numerical solution is unstable and diverges rapidly. This is an unphysical result and is simply an artifact of the explicit Euler differencing scheme. In fact, most explicit numerical schemes are unstable for large values of the timestep.

We can follow the same procedure in order to study the stability properties of the implicit scheme of equation (7.10). For the carbon dating problem,

$$f(\Delta\tau) = f_0 + \frac{1}{2}\left[-f(0) - f(\delta\tau)\right]\Delta\tau \ , \tag{7.32}$$

which is equivalent to

$$f(\delta\tau) = \frac{2-\Delta\tau}{2+\Delta\tau} f_0 \ . \tag{7.33}$$

After having taken N timesteps, the numerical solution becomes

$$f(N\cdot\Delta\tau) = \left(\frac{2-\Delta\tau}{2+\Delta\tau}\right)^N f_0 \ , \tag{7.34}$$

which is unconditionally stable. This is a characteristic of most implicit schemes.

Even though the Euler scheme was unstable for large sizes of the timestep, the upper limit on its size that we calculated with condition (7.31) does not introduce any problems in solving the carbon dating problem. Indeed, accuracy considerations alone will force us to use a timestep much smaller than this upper limit (see Fig. 7.2). There are situations, however, in which the stability of an explicit numerical scheme is not just an intellectual curiosity but a serious limiting factor on the applicability of the scheme. The most common situation in computational physics is the study of systems in which phenomena occur over a wide range of timescales, which we will discuss below.

We will start by considering a physical system that is described by a system of first order differential equations

$$\frac{df_i}{dt} = -\sum_{j=1}^{N} A_{ij} f_j \ , \qquad\qquad j = 1, ..., N \ . \tag{7.35}$$

We can also write this system in matrix notation as

$$\frac{d}{dt}\mathbf{F} = -\mathbf{A}\cdot\mathbf{F} \ , \tag{7.36}$$

where $\mathbf{A}$ is a $N \times N$ positive definite matrix. Applying the Euler scheme gives

$$\mathbf{F}(N\cdot\Delta t) = (\mathbf{1}\text{-}\mathbf{A}\Delta t)^N \cdot \mathbf{F}_0 \ , \tag{7.37}$$

where $\mathbf{1}$ is the unit matrix. The numerical solution tends to zero at late times if and only if the largest eigenvalue of the matrix $\mathbf{C}\equiv(\mathbf{1}\text{-}\mathbf{A})\Delta t$ is less than unity. This is equivalent to the timestep being smaller than

$$\Delta t < \frac{2}{\lambda_{\max}} \ , \tag{7.38}$$

where $\lambda_{\max}$ is the largest eigenvalue of the matrix $\mathbf{A}$.

The inverse of each eigenvalue of the matrix $\mathbf{A}$ is typically the characteristic timescale over which each physical phenomenon described by the system of equations (7.35) evolves. As a result, the Euler scheme is stable if and only if the timestep is shorter than twice the shortest timescale, $\tau_{\min}$, in the physical problem. On the other hand, accuracy considerations require the timestep to be larger than