List of Figures

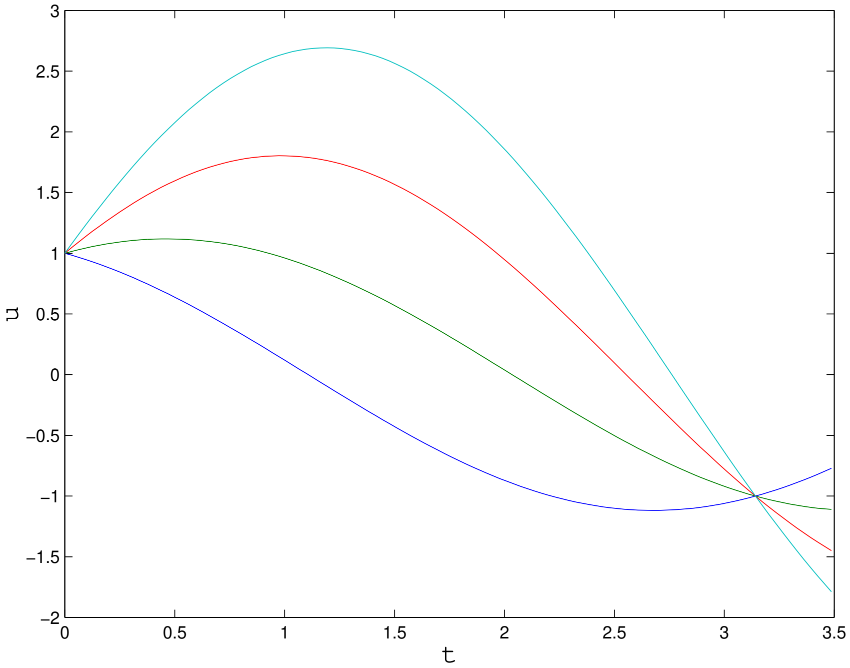


Figure 1.1: *u vs. t for $u(0) = 1$ and various values of $u'(0)$.*

can always be solved uniquely for $\beta = \tan^{-1}\frac{c_1}{c_2}$ and $\alpha = \frac{c_1}{\sin\beta}$ (or $\alpha = \frac{c_2}{\cos\beta}$; at least one of these is well defined). The IVP has a unique solution for any initial data $\mathbf{c} = (c_1, c_2)^T$. Such solution curves are plotted for $c_1 = 1$ and different values of c_2 in Figure 1.1.

- ***Boundary value problem*** (BVP): Given values $u(0) = c_1$ and $u(b) = c_2$, it appears from Figure 1.1 that for $b = 2$, say, if c_1 and c_2 are chosen carefully then there is a unique solution curve that passes through them, just like in the initial value case. However, consider the case where $b = \pi$: now different values of $u'(0)$ yield the same value $u(\pi) = -u(0)$ (see again Figure 1.1). So, if the given value of $u(b) = c_2 = -c_1$ then we have infinitely many solutions, whereas if $c_2 \neq -c_1$ then no solution exists.

This simple illustration already indicates some important general issues. For IVPs, one starts at the initial point with all the solution information and marches with it (in "time")—the process is *local*. For BVPs the entire solution information (for a second-order problem this consists of u and u') is not locally known anywhere, and the process of constructing a solution is *global* in t. Thus we may expect many more (and different) difficulties with the latter, and this is reflected in the numerical procedures discussed in this book.

1.1 IVPs

The general form of an IVP that we shall discuss is

$$\begin{aligned} \mathbf{y}' &= \mathbf{f}(t, \mathbf{y}), \qquad 0 \le t \le b, \\ \mathbf{y}(0) &= \mathbf{c} \ \text{ (given)}. \end{aligned} \tag{1.1}$$

Here $\mathbf{y}$ and $\mathbf{f}$ are vectors with m components, $\mathbf{y} = \mathbf{y}(t)$, and $\mathbf{f}$ is in general a nonlinear function of t and $\mathbf{y}$. When $\mathbf{f}$ does not depend explicitly on t, we speak of the *autonomous* case. When describing general numerical methods we shall often assume the autonomous case simply in order to carry less notation around. The simple example from the beginning of this chapter is in the form (1.1) with $m = 2$, $\mathbf{y} = (u, u')^T$, $\mathbf{f} = (u', -u)^T$.

In (1.1) we assume, for simplicity of notation, that the starting point for t is 0. An extension of everything which follows to an arbitrary interval of integration $[a, b]$ is obtained without difficulty.

Before proceeding further, we give three examples which are famous for being very simple on one hand and for representing important classes of applications on the other hand.

Example 1.1 (simple pendulum)
Consider a tiny ball of mass 1 attached to the end of a rigid, massless rod of length 1. At its other end the rod's position is fixed at the origin of a planar coordinate system (see Figure 1.2).

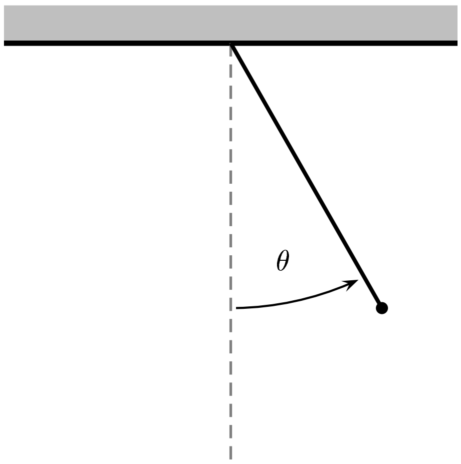


Figure 1.2: *Simple pendulum.*

Denoting by θ the angle between the pendulum and the y-axis, the friction-free motion is governed by the ODE (cf. Example 1.5 below)

$$\theta'' = -g \sin\theta, \tag{1.2}$$

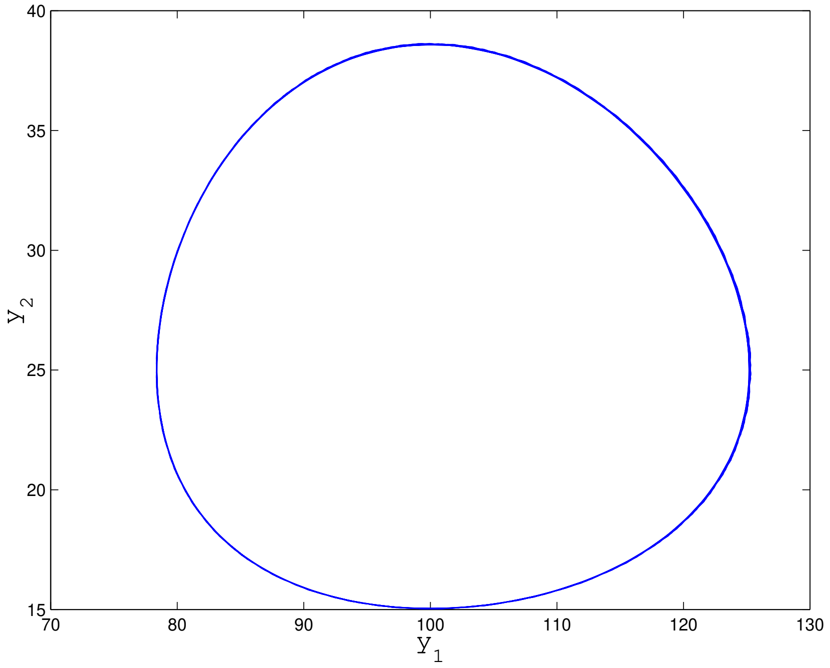


Figure 1.3: *Periodic solution forming a cycle in the $y_1 \times y_2$ plane.*

we obtain an ODE in the form (1.1) with $m = 2$ components, describing the time-evolution of these populations.

The qualitative question here is, starting from some initial values $\mathbf{y}(0)$ out of a set of reasonable possibilities, will these two populations survive or perish in the long run? As it turns out, this model possesses periodic solutions: starting, say, from $\mathbf{y}(0) = (80, 30)^T$, the solution reaches the same pair of values again after some time period T, i.e., $\mathbf{y}(T) = \mathbf{y}(0)$. Continuing to integrate past T yields a repetition of the same values, $\mathbf{y}(T + t) = \mathbf{y}(t)$. Thus, the solution forms a cycle in the phase plane (y_1, y_2) (see Figure 1.3). Starting from any point on this cycle, the solution stays on the cycle for all time. Other initial values not on this cycle yield other periodic solutions with a generally different period. So, under these circumstances the populations of the predator and prey neither explode nor vanish for all future times, although their number never becomes constant.[1] ♦

[1]In other examples, such as the Van der Pol equation (7.13), the solution forms an attracting *limit cycle*: starting from any point on the cycle the solution stays on it for all time, and starting from points near the solution, it tends in time towards the limit cycle.
The neutral stability of the cycle in our current example, in contrast, is one reason why this predator-prey model is discounted among mathematical biologists as being too simple.

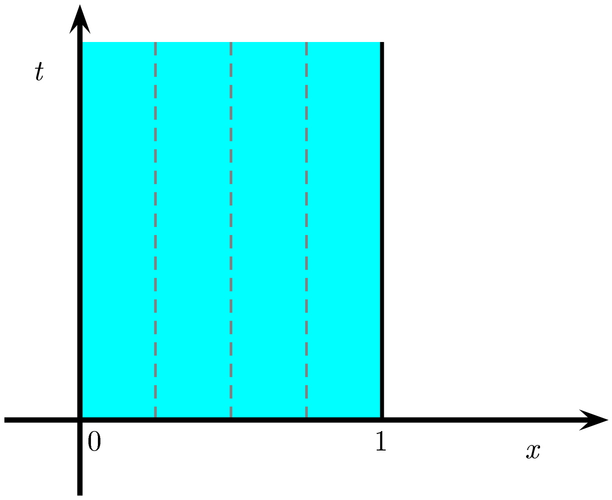


Figure 1.4: *Method of lines. The shaded strip is the domain on which the diffusion PDE is defined. The approximations $y_i(t)$ are defined along the dashed lines.*

Example 1.3 (a diffusion problem)
A typical diffusion problem in one space variable x and time t leads to the partial differential equation (PDE)

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(p\frac{\partial u}{\partial x}\right) + g(x,u),$$

for an unknown function $u(t,x)$ of two independent variables defined on a strip $0 \le x \le 1$, $t \ge 0$. For simplicity, assume that $p = 1$ and g is a known function. Typical side conditions which make this problem well posed are

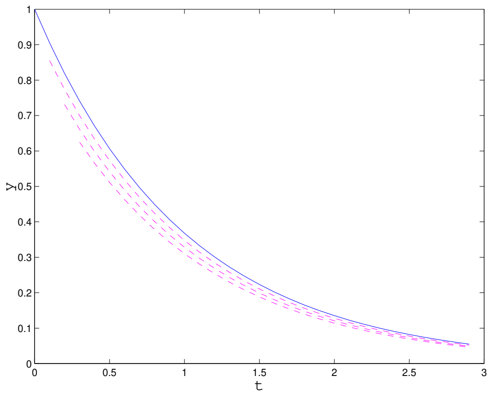
$$\begin{aligned} u(0,x) = q(x),\ 0 \le x \le 1 &\qquad \text{(initial conditions)},\\ u(t,0) = \alpha(t),\ u(t,1) = \beta(t),\ t \ge 0 &\qquad \text{(boundary conditions)}. \end{aligned}$$

To solve this problem numerically, consider discretizing in the space variable first. For simplicity assume a uniform mesh with spacing $\Delta x = 1/(m+1)$, and let $y_i(t)$ approximate $u(x_i,t)$, where $x_i = i\Delta x$, $i = 0,1,\ldots,m+1$. Then replacing $\frac{\partial^2 u}{\partial x^2}$ by a second-order central difference, we obtain

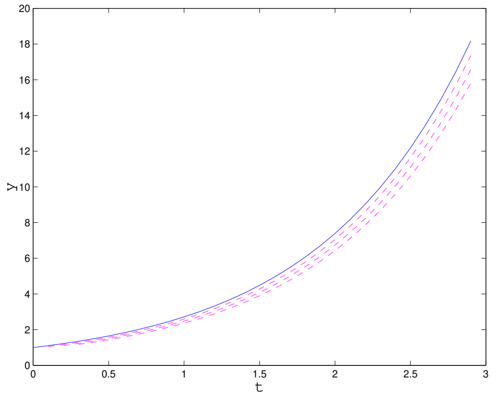
$$\frac{dy_i}{dt} = \frac{y_{i+1} - 2y_i + y_{i-1}}{\Delta x^2} + g(x_i,y_i), \qquad i = 1,\ldots,m,$$

with $y_0(t) = \alpha(t)$ and $y_{m+1}(t) = \beta(t)$ given. We have obtained an initial value ODE problem of the form (1.1) with the initial data $c_i = q(x_i)$.

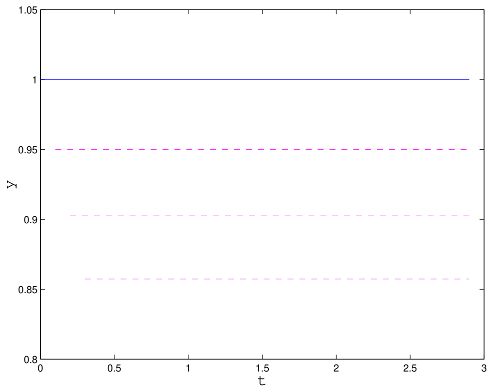
This technique of replacing spatial derivatives by finite difference approximations and solving an ODE problem in time is referred to as the *method of lines*. Figure 1.4 illustrates the origin of the name. Its more general form is discussed further in Example 1.7 below. ♦



(a) Error propagation for $y' = -y$



(b) Error propagation for $y' = y$



(c) Error propagation for $y' = 0$

Figure 2.1: *Errors due to perturbations for stable and unstable test equations. The original, unperturbed trajectories are in solid curves, the perturbed in dashed. Note that the y-scales in* (a) *and* (b) *are* not *the same.*

Then the change of variables

$$\mathbf{w} = T^{-1}\mathbf{y}$$

yields the ODE for $\mathbf{w}$:

$$\mathbf{w}' = \Lambda\mathbf{w}.$$

The system for $\mathbf{w}$ is decoupled: for each component w_i of $\mathbf{w}$ we have a test equation $w_i' = \lambda_i w_i$. Therefore, the stability for $\mathbf{w}$, hence also for $\mathbf{y}$, is determined by the eigenvalues: stability is obtained if $\mathcal{R}\mathrm{e}(\lambda_i) \leq 0$, for all $i = 1, \ldots, m$, and asymptotic stability holds if the inequalities are all strict.

In the more general case, A may not be similar to any diagonal matrix.

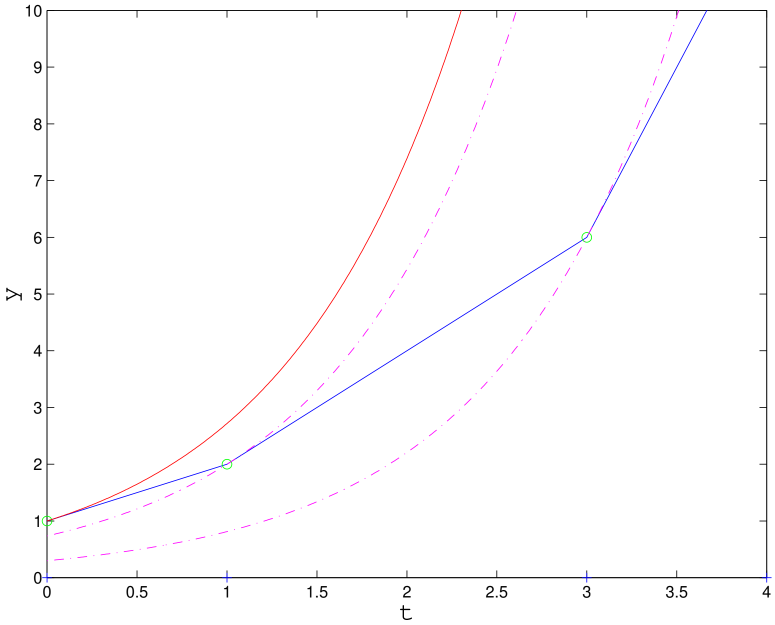


Figure 3.1: *The forward Euler method. The exact solution is the curved solid line. The numerical values are circled. The broken line interpolating them is tangential at the beginning of each step to the ODE trajectory passing through that point (dashed lines).*

This is a simple, *explicit* method; starting from $\mathbf{y}_0 = \mathbf{c}$ we apply (3.4) iteratively for $n = 1, 2, \ldots, N$. The effect of the approximation is depicted in Figure 3.1. The curved lines represent a family of solutions for the ODE with different initial values. At each step, the approximate solution $\mathbf{y}_{n-1}$ is on one of these curves at t_{n-1}. The forward Euler step amounts to taking a straight line in the tangential direction to the exact trajectory starting at $(t_{n-1}, \mathbf{y}_{n-1})$, continuing until the end of the step. (Recall Example 2.1.) One hopes that if h is small enough, then $\mathbf{y}_n$ is not too far from $\mathbf{y}(t_n)$. Let us assess this hope.

3.2 Convergence, Accuracy, Consistency, and 0-Stability

We now rewrite Euler's method (3.4) in a form compatible with the approximated ODE,

$$\frac{\mathbf{y}_n - \mathbf{y}_{n-1}}{h_n} - \mathbf{f}(t_{n-1}, \mathbf{y}_{n-1}) = \mathbf{0}.$$

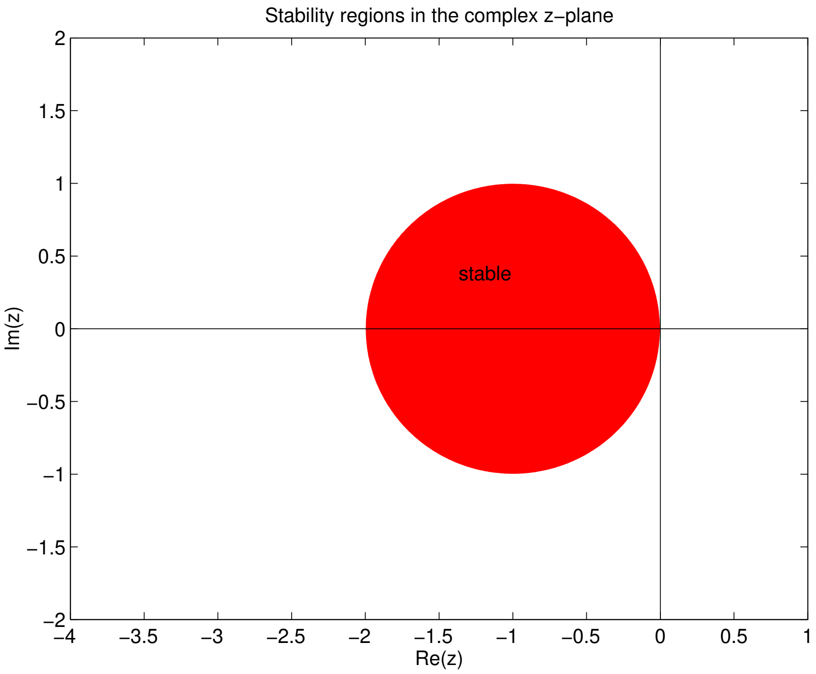


Figure 3.2: *Absolute stability region for the forward Euler method.*

For a given numerical method, the *region of absolute stability* is that region of the complex z-plane such that applying the method for the test equation (3.15), with $z = h\lambda$ from within this region, yields an approximate solution satisfying the absolute stability requirement (3.16).

For the forward Euler method we obtain the condition

$$|1 + h\lambda| \leq 1 \tag{3.17}$$

which yields the region of absolute stability depicted in Figure 3.2. For instance, if λ is negative, then h must be restricted to satisfy

$$h \leq \frac{2}{-\lambda}.$$

For Example 3.1 this gives $h < .2$. In this case the restriction is not practically unbearable. To see the effect of violating the absolute stability restriction, we plot in Figure 3.3 the approximate solutions obtained with uniform step sizes $h = .19$ and $h = .21$. For $h = .19$ the solution profile looks like the exact one ($y = 1/t$). The other, oscillatory profile is obtained for $h = .21$, which is outside the absolute stability region. When computing with $h = .4$ using the same forward Euler method and floating point arithmetic with a 14-hexadecimal-digit mantissa (this is the standard "double precision" in IEEE FORTRAN, for example), the computed solution oscillates and then blows up (i.e., overflow is detected) before reaching $t = 25$.

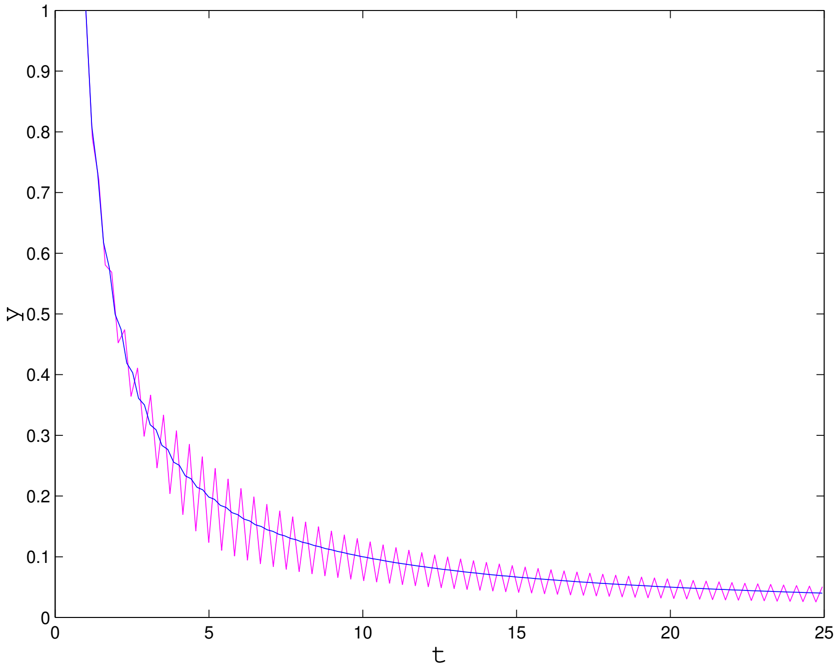


Figure 3.3: *Approximate solutions for Example* 3.1 *using the forward Euler method, with* $h = .19$ *and* $h = .21$. *The oscillatory profile corresponds to* $h = .21$*; for* $h = .19$ *the qualitative behavior of the exact solution is obtained.*

It is important to understand that the absolute stability restriction is indeed a stability, not accuracy, requirement. Consider the initial value $c = 10^{-15}$ for the test equation with $\mathcal{R}\mathrm{e}(\lambda) < 0$, so the exact solution is approximated very well by the constant 0. Such an initial value corresponds to an unavoidable perturbation in the numerical method, due to roundoff errors. Now, the forward Euler solution corresponding to this initial perturbation of 0 remains very close to 0 for all $t_n > 0$, like the exact solution, when using any h from the absolute stability region, but it blows up as n increases if h is from outside that region, i.e., if $|1 + h\lambda| > 1$.

The concept of absolute stability was defined with respect to a very simple test equation (3.15), an ODE whose numerical solution is not a computationally challenging problem in itself. Nonetheless, it turns out that absolute stability gives useful information, at least qualitatively, in more general situations, where complicated systems of nonlinear ODEs are integrated numerically.

Note: Those readers who are prepared to trust us on the above statement may wish to skip the rest of this section, at least on first reading (especially if your linear algebra needs some dusting).

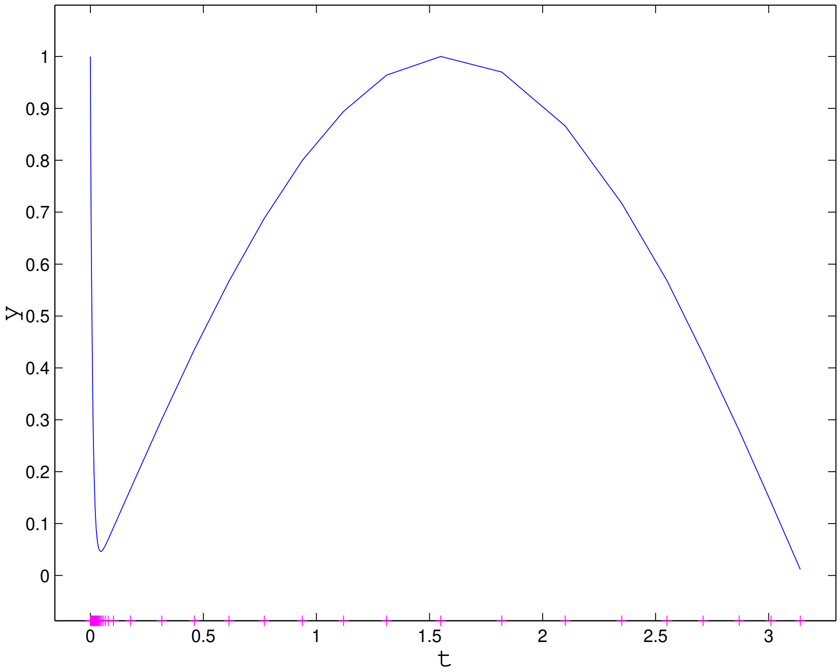


Figure 3.4: *Approximate solution and plausible mesh, Example* 3.2.

3.4 Stiffness: Backward Euler

Ideally, the choice of step size h_n should be dictated by approximation accuracy requirements. But we just saw that when using the forward Euler method (and, as it turns out, many other methods too), h_n must be chosen sufficiently small to obey an additional, absolute stability restriction, as well. Loosely speaking, the IVP is referred to as being *stiff* if this absolute stability requirement dictates a much smaller step size than is needed to satisfy approximation requirements alone. In this case other methods, which do not have such a restrictive absolute stability requirement, should be considered.

To illustrate this, consider a simple example.

Example 3.2
The scalar problem

$$y' = -100(y - \sin t), \quad t \geq 0, \quad y(0) = 1$$

has a solution which starts at the given initial value and varies rapidly. But after a short while, say, for $t \geq 0.03$, $y(t)$ varies much more slowly, satisfying $y(t) \approx \sin t$; see Figure 3.4. For the initial small interval of rapid change (commonly referred to as an initial *layer* or transient), we expect to use small step sizes, so that $100h_n \leq 1$, say. This is within the absolute stability

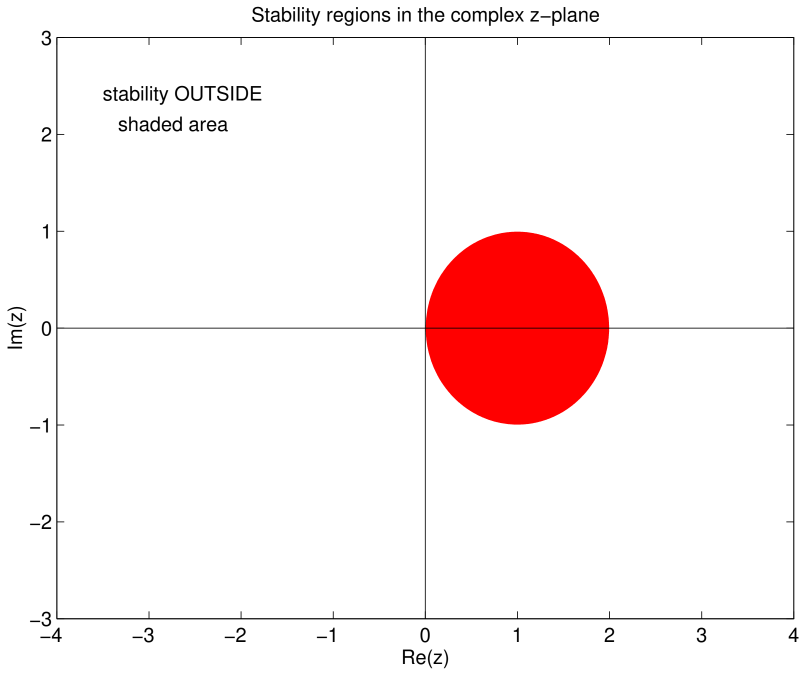


Figure 3.5: *Absolute stability region for the backward Euler method.*

The region of absolute stability of the backward Euler method is depicted in Figure 3.5. It contains, in addition to the entire left half-plane of $z = h\lambda$, a major part of the right half-plane as well. (The latter is a mixed blessing, though, as will be discussed later on.) For a given stiff problem, the backward Euler method needs fewer steps than the forward Euler method. In general, however, each backward Euler step may be more expensive in terms of computing time. Still, there are many applications where the overall computational expense using the implicit method is much less than with the explicit Euler method.

3.4.2 Solving Nonlinear Equations

For an implicit method like backward Euler, a nonlinear system of equations must be solved in each time step. For backward Euler, this nonlinear system is

$$\mathbf{g}(\mathbf{y}_n) = \mathbf{y}_n - \mathbf{y}_{n-1} - h\mathbf{f}(t_n, \mathbf{y}_n) = \mathbf{0}$$

(where $h = h_n$ for notational simplicity). There are a number of ways to solve this nonlinear system. We mention two basic ones.

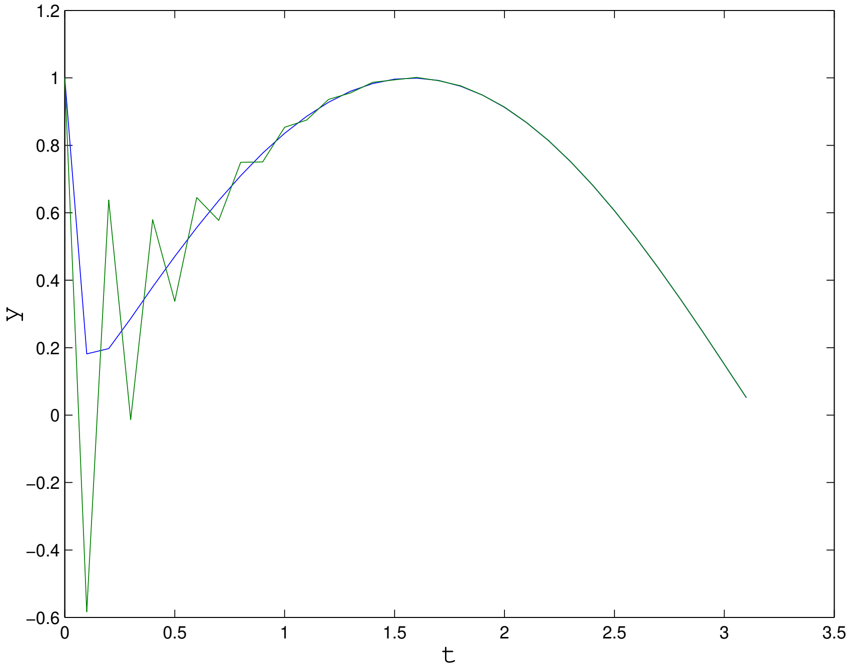


Figure 3.6: *Approximate solution on a coarse uniform mesh for Example* 3.2*, using backward Euler (the smoother curve) and trapezoidal methods.*

This is a stronger requirement than absolute stability in the very stiff limit, and it does not relate to what happens elsewhere in the $h\lambda$-plane. The backward Euler method has stiff decay, because when applied to (3.28) it yields

$$y_n - g(t_n) = (1 - h_n\lambda)^{-1}(y_{n-1} - g(t_n)).$$

The forward Euler method, of course, does not have stiff decay.

The practical advantage of methods with stiff decay lies in their ability to skip fine-level (i.e., rapidly varying) solution details and still maintain a decent description of the solution on a coarse level in the very stiff (not the highly oscillatory!) case. For instance, using backward Euler with a fixed step $h = .1$ to integrate the problem of Example 3.2, the initial layer is poorly approximated, and still the solution is qualitatively recovered where it varies slowly; see Figure 3.6. Herein lies a great potential for efficient use, as well as a great danger of misuse, of such discretization methods.

3.6 Symmetry: Trapezoidal Method

The forward Euler method was derived using a Taylor expansion centered at t_{n-1}. The backward Euler method was likewise derived, centered at t_n instead. Both methods are first-order accurate, which is often insufficient

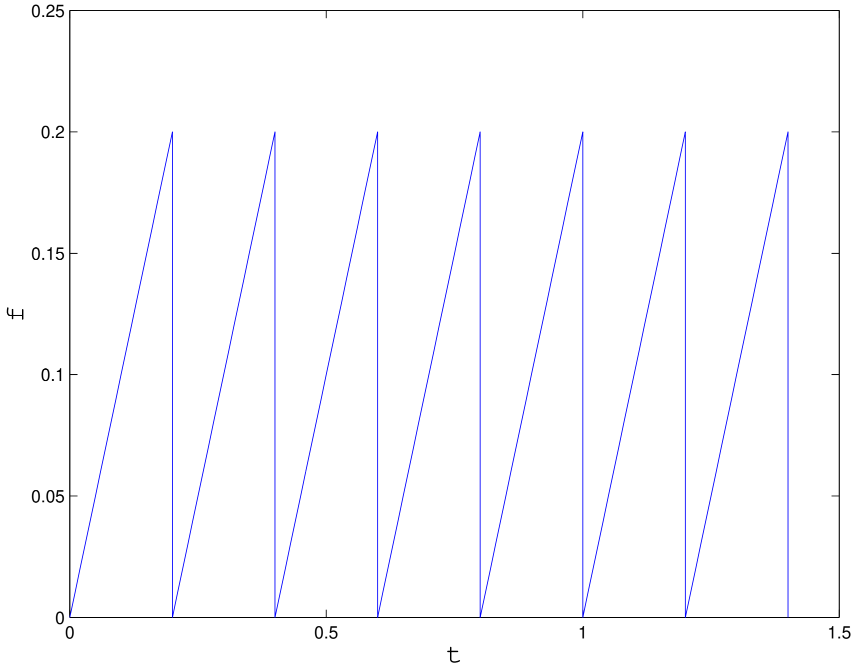


Figure 3.7: *Sawtooth function for* $\tau = 0.2$.

where $\tau > 0$ is a parameter. The ODE $y' = f$ is therefore a quadrature problem for a sawtooth function; see Figure 3.7. With $y(0) = 0$, the solution is

$$y(t) = j\tau^2/2 + (t - j\tau)^2/2, \qquad j\tau \le t < (j+1)\tau, \quad j = 0, 1, \ldots, J.$$

We also calculate that away from the discontinuity points $j\tau$,

$$y''(t) = 1, \qquad y'''(t) = 0.$$

For the trapezoidal method, the local truncation error therefore vanishes on a step $[t_{n-1}, t_n]$ that does not contain a point of discontinuity. For this special case, the trapezoidal method using a constant step size h reproduces $y(t_n)$ exactly if $\tau = lh$ for some positive integer l. If, on the other hand, $\tau = (l+r)h$ for some fraction r, then an $O(h\tau)$ error results, up to J times, for a combined $O(h)$ error. A worse, $O(1)$ error, may result if the "teeth are sharper," e.g., $f(t) = t/\tau - j$, $j\tau \le t < (j+1)\tau$, and $\tau = O(h)$. ♦

If $\mathbf{f}(t, \mathbf{y})$ has a discontinuity, then it is important to locate it and place a mesh point as close to it as possible. Unlike in the simple Example 3.4, however, the precise location of such a discontinuity may not be known. In many examples, the discontinuity is defined by a *switching function*, e.g.,

$$\mathbf{f}(t, \mathbf{y}) = \begin{cases} \mathbf{f}_I(t, \mathbf{y}(t)) & \text{if } g(t, \mathbf{y}(t)) < 0, \\ \mathbf{f}_{II}(t, \mathbf{y}(t)) & \text{if } g(t, \mathbf{y}(t)) > 0. \end{cases}$$

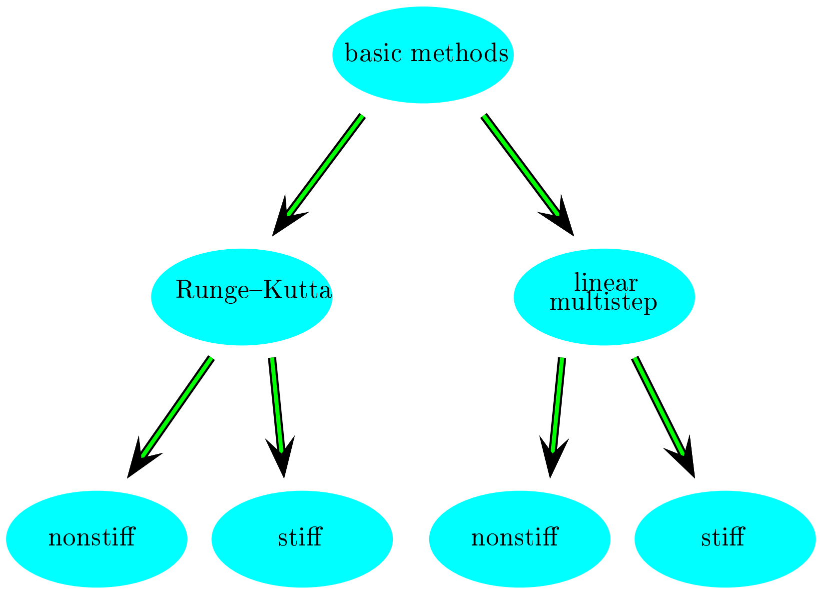


Figure 4.1: *Classes of higher-order methods.*

with $f(t, y)$ and its derivatives evaluated at (t_{n-1}, y_{n-1}),

$$\begin{aligned} y'_{n-1} &= f, \\ y''_{n-1} &= f_t + f_y f, \\ y'''_{n-1} &= f_{tt} + 2f_{ty} f + f_y f_t + f_{yy} f^2 + f_y^2 f, \end{aligned} \tag{4.1}$$

etc. The local truncation error is $h^p y^{(p+1)}(t_n)/(p+1)! + O(h^{p+1})$. For a system of differential equations, the derivatives are defined similarly.

A problem with this method is that it requires analytic expressions for derivatives which in a practical application can be quite complicated. On the other hand, advances in compiler technology have enabled much more robust programs for symbolic and automatic differentiation in recent years, which may make this method more attractive for some applications.

We thus seek one-step methods that achieve a higher accuracy order without forming the symbolic derivatives of the Taylor series method. This leads to *Runge–Kutta methods*, to which the rest of this chapter is devoted.

4.1 The First Runge–Kutta Methods

We stay with a scalar ODE for some of this exposition. The extension to ODE systems is straightforward, and will be picked up later.

Many Runge–Kutta methods are based on quadrature schemes. In fact, the reader may want to quickly review basic quadrature rules at this point.

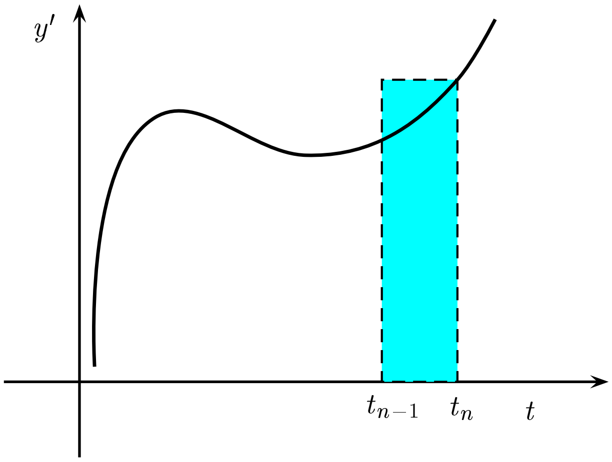


Figure 4.2: *Approximate area under curve.*

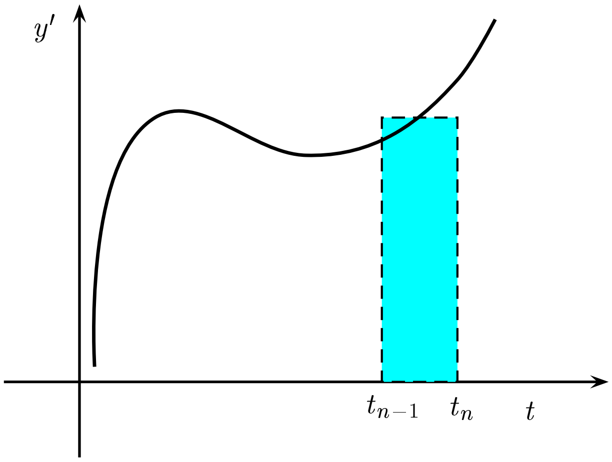


Figure 4.3: *Midpoint quadrature.*

function evaluations of f within the interval $[t_{n-1}, t_n]$. Note that this method is not linear in f anymore (substitute (4.3a) into (4.3b) to see this). At first glance, it might seem that the order would be limited to 1, because the first stage (4.3a) uses the forward Euler method, which is first order. However, note that the term involving $\hat{y}_{n-1/2}$ enters into (4.3b) multiplied by h, and therefore its error becomes less important.

Indeed, the local truncation error of (4.3) is given by

$$\begin{aligned} d_n &= \frac{y(t_n) - y(t_{n-1})}{h} - f\left(t_{n-1/2}, y(t_{n-1}) + \frac{h}{2} f(t_{n-1}, y(t_{n-1}))\right) \\ &= y' + \frac{h}{2} y'' + \frac{h^2}{6} y''' \\ &- \left(f + \frac{h}{2}(f_t + f_y f) + \frac{h^2}{8}(f_{tt} + 2 f_{ty} f + f_{yy} f^2)\right) + O(h^3), \end{aligned} \tag{4.4}$$

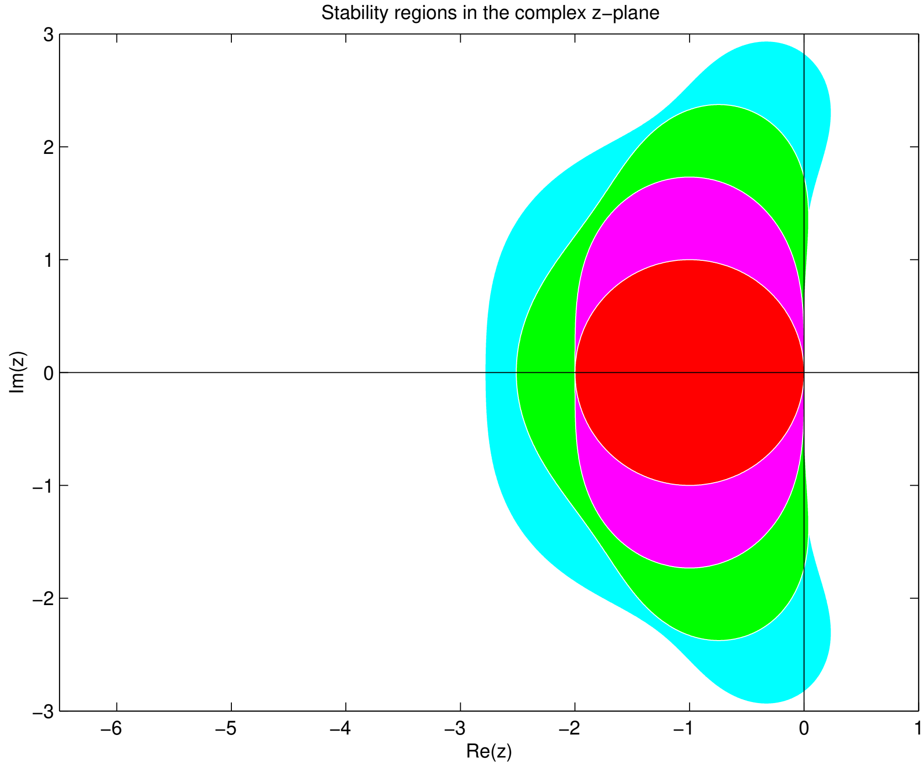


Figure 4.4: *Stability regions for p-stage explicit Runge–Kutta methods of order p, $p = 1, 2, 3, 4$. The inner circle corresponds to forward Euler, $p = 1$. The larger p is, the larger the stability region. Note the "ear lobes" of the fourth-order method protruding into the right half-plane.*

For explicit Runge–Kutta methods $R(z)$ is a polynomial in $z = h\lambda$ given, e.g., by the expression whose magnitude appears in (4.19). Thus, to find the boundary of the region of absolute stability, we find the roots $z(\theta)$ of

$$R(z) = e^{i\theta}$$

for a sequence of θ values. Starting with $\theta = 0$, for which $z = 0$, we repeatedly increase θ by a small increment, each time applying a root finder to find the corresponding z, starting from z of the previous θ as a first guess,[11] until the stability boundary curve returns to the origin.

It is also possible to compute the region of absolute stability via a brute force approach. To do this, we first form a grid over a large part of the complex plane including the origin. Then at each mesh point z_{ij}, if $|R(z_{ij})| < 1$, we mark z_{ij} as being inside the stability region.

Finally, we note that no explicit Runge–Kutta method can have an unbounded region of absolute stability. This is because all Runge–Kutta meth-

[11] This is an elementary example of a *continuation* method.

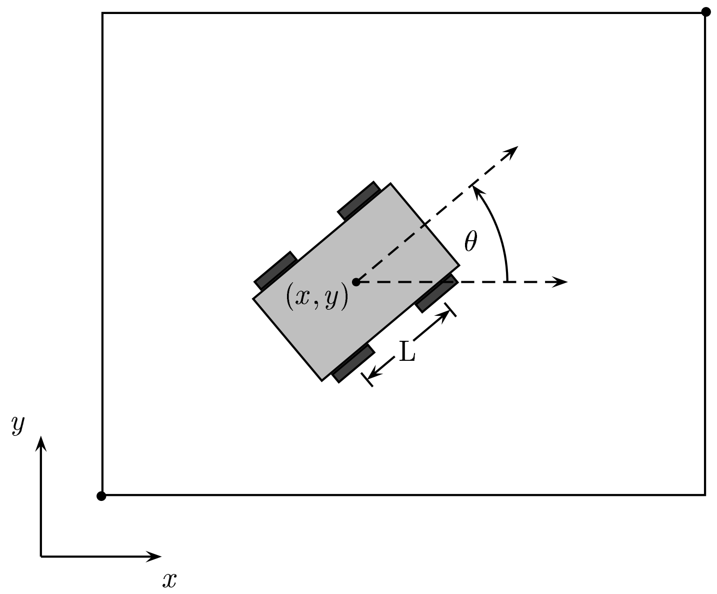


Figure 4.5: *Schematic of a toy car (not to scale).*

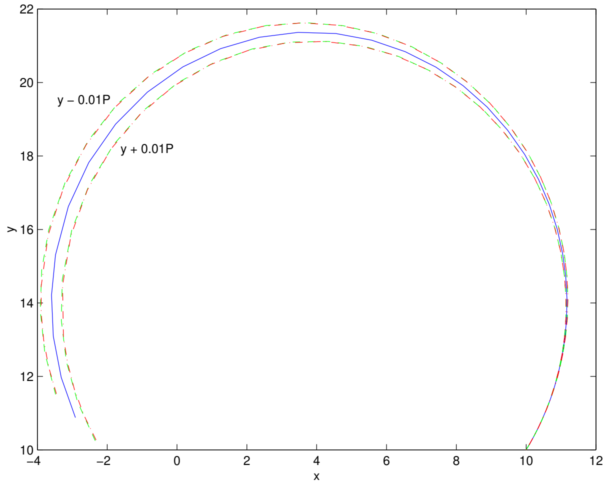
Example 4.7
A simplified description of the motion of a car in an arena is given by the equations corresponding to the ODE in (4.20),

$$x' = v\cos\theta, \tag{4.24a}$$
$$y' = v\sin\theta, \tag{4.24b}$$
$$\theta' = v\frac{\tan\psi}{L}, \tag{4.24c}$$
$$v' = a - \gamma v, \tag{4.24d}$$

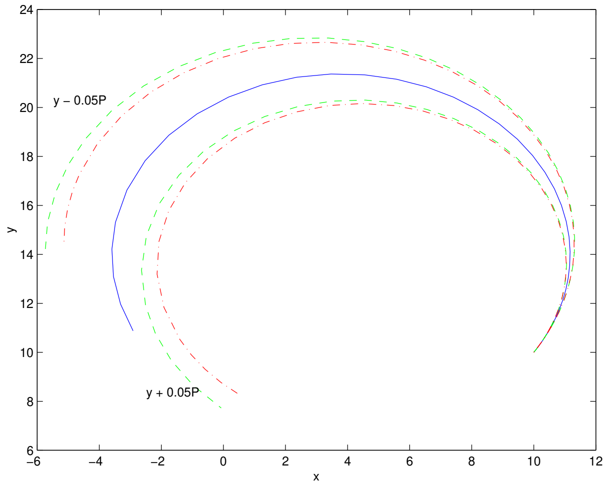
where x and y are the Cartesian coordinates of the car's center, θ is the angle the car makes with the x-axis (see Figure 4.5), and v is the velocity with which the car is moving. Denote $\mathbf{y} = (x, y, \theta, v)^T$.

The damping (friction) factor γ and the car's length L are given: we take $\gamma = 1/6$ and $L = 11.5$ cm (it's a toy car, representative of a mobile robot). The acceleration a and the steering angle ψ that the front wheels make with the car's body are two functions which one normally controls in order to drive a car. Here we take them, for simplicity, as constant parameters: we set $a = 100$ cm s^{-2}, $\psi = 1$. We are interested in the sensitivity of the car's position (x, y) with respect to a constant change in ψ.

Since we are checking sensitivity with respect to only one parameter, $P(t)$ is a vector of length 4. The differential equations (4.23) for $P(t)$ are



(a) $\phi = 0.01$



(b) $\phi = 0.05$

Figure 4.6: *Toy car routes under constant steering: unperturbed (solid line), steering perturbed by* $\pm\phi$ *(dash-dot lines), and corresponding trajectories computed by the linear sensitivity analysis (dashed lines).*

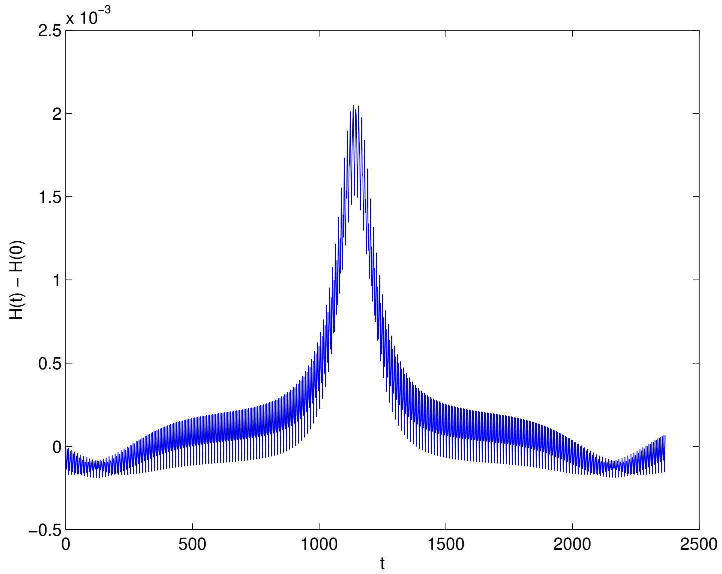


Figure 4.7: *Energy error for the Morse potential using leapfrog with* $h =$ 2.3684.

To evaluate $\mathbf{q}_n$ at any mesh point, the expression

$$\mathbf{q}_n = \frac{1}{2}(\mathbf{q}_{n-1/2} + \mathbf{q}_{n+1/2})$$

can be used.

Show that this method is explicit and second-order accurate.

(c) Integrate the Morse problem defined in the previous exercise using 1000 uniform steps h. Apply three methods: forward Euler, symplectic Euler, and leapfrog. Try the values $h = 2$, $h = 2.3684$, and $h = 2.3685$ and plot in each case the discrepancy in the Hamiltonian (which equals 0 for the exact solution). The plot for $h = 2.3684$ is given in Figure 4.7.

What are your observations? [The surprising increase in leapfrog accuracy from $h = 2.3684$ to $h = 2.3685$ relates to a phenomenon called *resonance instability*.]

[Both the symplectic Euler and the leapfrog method are *symplectic*—like the exact ODE they conserve certain volume projections for Hamiltonian systems (Section 2.5). We refer to [82, 50, 93] for much more on symplectic methods.]

4.12. The following classical example from astronomy gives a strong motivation to integrate initial value ODEs with error control.

Consider two bodies of masses $\mu = 0.012277471$ and $\hat{\mu} = 1 - \mu$ (earth and sun) in a planar motion, and a third body of negligible mass

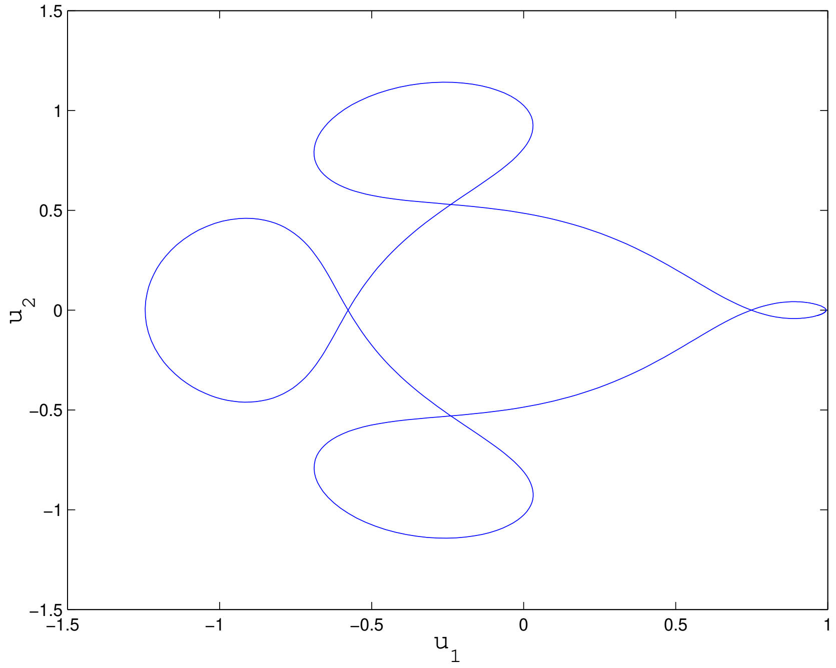


Figure 4.8: *Astronomical orbit using a Runge–Kutta* 4(5) *embedded pair method.*

(moon) moving in the same plane. The motion is governed by the equations

$$\begin{aligned}
u_1'' &= u_1 + 2u_2' - \hat{\mu}\frac{u_1+\mu}{D_1} - \mu\frac{u_1-\hat{\mu}}{D_2},\\
u_2'' &= u_2 - 2u_1' - \hat{\mu}\frac{u_2}{D_1} - \mu\frac{u_2}{D_2},\\
D_1 &= ((u_1+\mu)^2 + u_2^2)^{3/2},\\
D_2 &= ((u_1-\hat{\mu})^2 + u_2^2)^{3/2}.
\end{aligned}$$

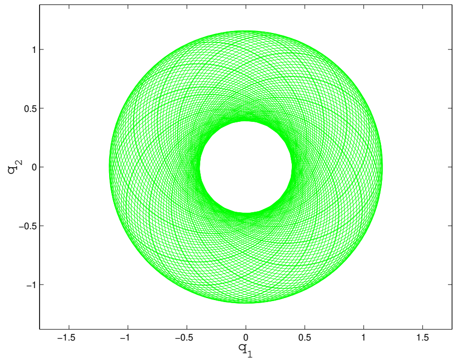
Starting with the initial conditions

$$u_1(0) = 0.994,\ u_2(0) = 0,\ u_1'(0) = 0,$$
$$u_2'(0) = -2.00158510637908252240537862224,$$

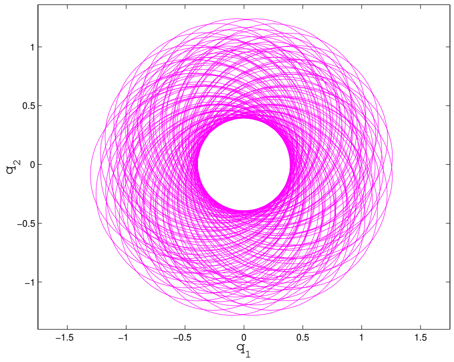
the solution is periodic with period < 17.1. Note that $D_1 = 0$ at $(-\mu, 0)$ and $D_2 = 0$ at $(\hat{\mu}, 0)$, so we need to be careful when the orbit passes near these singularity points.

The orbit is depicted in Figure 4.8. It was obtained using a 4(5) embedded pair with a local error tolerance 1.e − 6. This necessitated 204 time steps.

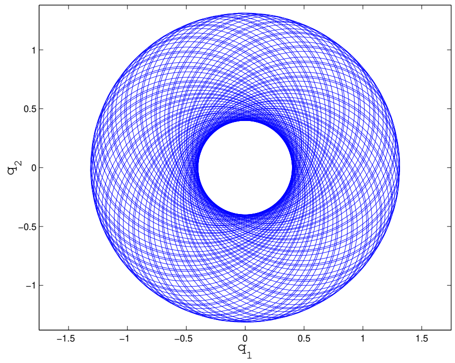
Using the classical Runge–Kutta method of order 4, integrate this problem on $[0, 17.1]$ with a *uniform* step size, using 100, 1000, 10,000,



(a) Implicit midpoint, 5000 uniform steps



(b) RK4, 5000 uniform steps



(c) Exact solution for T=500

Figure 4.9: *Modified Kepler problem: Approximate and exact solutions.*

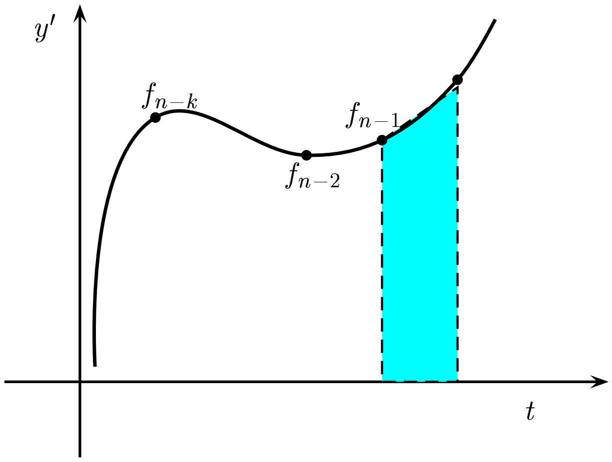


Figure 5.1: *Adams–Bashforth methods.*

general form (5.1) we therefore set, *for all Adams methods*, $\alpha_0 = 1$, $\alpha_1 = -1$, and $\alpha_j = 0$, $j > 1$.

The k-step **explicit Adams method** is obtained by interpolating f through the previous points $t = t_{n-1}, t_{n-2}, \ldots, t_{n-k}$; see Figure 5.1.

The explicit Adams methods[16] are the most popular among explicit multistep methods. A simple exercise in polynomial interpolation yields the formula

$$y_n = y_{n-1} + h \sum_{j=1}^{k} \beta_j f_{n-j},$$

where[17]

$$\begin{aligned} \beta_j &= (-1)^{j-1} \sum_{i=j-1}^{k-1} \binom{i}{j-1} \gamma_i, \\ \gamma_i &= (-1)^i \int_0^1 \binom{-s}{i} ds\,. \end{aligned}$$

This formula is a k-step method because it uses information at the k points $t_{n-1}, t_{n-2}, \ldots, t_{n-k}$. It is sometimes also called a $(k+1)$-value method, because the total information per step, which determines storage requirements, also involves y_{n-1}.

[16] Called Adams–Bashforth after J. C. Adams, who invented them to solve a problem of capillary action in collaboration with F. Bashforth. They were published in 1883.

[17] Recall

$$\binom{s}{i} = \frac{s(s-1)\cdots(s-i+1)}{i!}, \qquad \binom{s}{0} = 1.$$

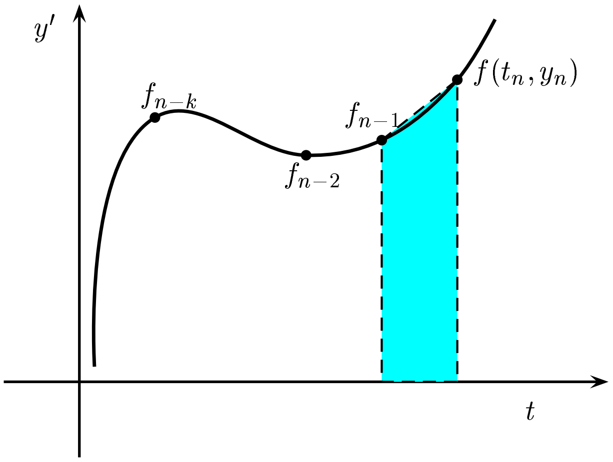


Figure 5.2: *Adams–Moulton methods.*

polynomial interpolation). An exception is the case for $k = 1$, where f_{n-1} is not used, yielding $p = k = 1$. A straightforward use of interpolation yields the coefficients summarized in Table 5.2.

p	k	$j \rightarrow$	0	1	2	3	4	5	C_{p+1}
1	1	β_j	1						$-\frac{1}{2}$
2	1	$2\beta_j$	1	1					$-\frac{1}{12}$
3	2	$12\beta_j$	5	8	-1				$-\frac{1}{24}$
4	3	$24\beta_j$	9	19	-5	1			$-\frac{19}{720}$
5	4	$720\beta_j$	251	646	-264	106	-19		$-\frac{3}{160}$
6	5	$1440\beta_j$	475	1427	-798	482	-173	27	$-\frac{863}{60480}$

Table 5.2: *Coefficients of Adams–Moulton methods up to order* 6.

Example 5.2
Here are some examples of Adams–Moulton methods:

- $k = 1$ with $\beta_1 = 0$ gives the backward Euler method;
- $k = 1$ with $\beta_1 \neq 0$ gives the implicit trapezoidal method;
- $k = 2$ gives $y_n = y_{n-1} + \frac{h}{12}[5f_n + 8f_{n-1} - f_{n-2}]$. ♦

The Adams–Moulton methods have smaller error constants than the Adams–Bashforth methods of the same order, and use one fewer step for

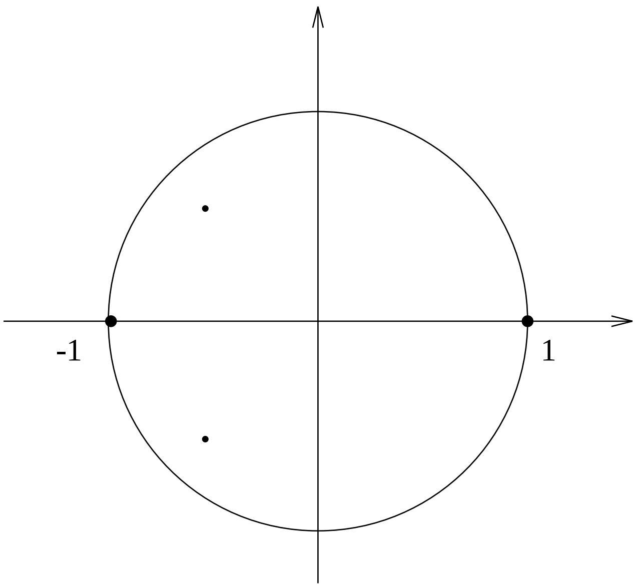


Figure 5.3: *Zeros of $\rho(\xi)$ for a 0-stable method.*

Theorem 5.1

- *The linear multistep method is 0-stable iff all roots ξ_i of the characteristic polynomial $\rho(\xi)$ satisfy*

$$|\xi_i| \leq 1,$$

and if $|\xi_i| = 1$, then ξ_i is a simple root, $1 \leq i \leq k$.

- *If the root condition is satisfied, the method is accurate to order p, and the initial values are accurate to order p, then the method is convergent to order p.*

Note that the root condition guaranteeing 0-stability relates to the characteristic polynomial $\rho(\xi)$ alone; see Figure 5.3. Also, for any consistent method the polynomial ρ has the root 1. One-step methods have no other roots, which again highlights the fact that they are automatically 0-stable.

Example 5.6
Instability is a disaster. Here is an example of an unstable method,

$$y_n = -4y_{n-1} + 5y_{n-2} + 4hf_{n-1} + 2hf_{n-2}\,.$$

In terms of the local truncation error, this is the most accurate explicit two-step method. However, $\rho(\xi) = \xi^2 + 4\xi - 5 = (\xi - 1)(\xi + 5)$. The extraneous root is $\xi_2 = -5$ and the root condition is violated.

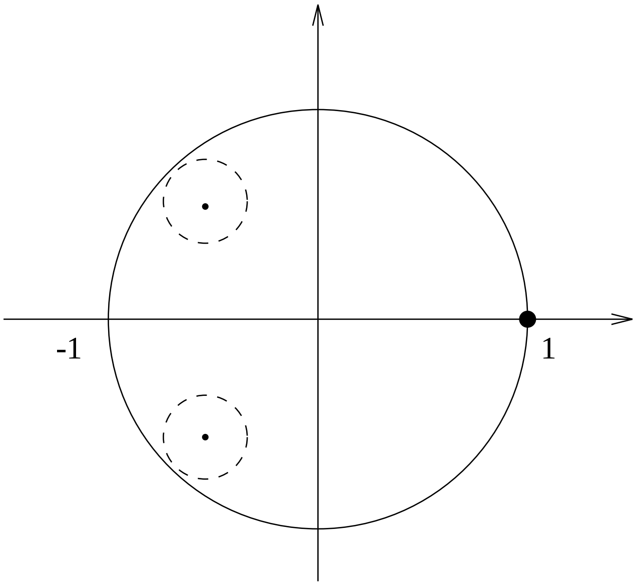


Figure 5.4: *Zeros of $\rho(\xi)$ for a strongly stable method. It is possible to draw a circle contained in the unit circle about each extraneous root.*

Consider solving $y' = 0$ with initial values $y_0 = 0$, $y_1 = \epsilon$. Then

$$\begin{aligned} y_2 &= -4y_1 = -4\epsilon, \\ y_3 &= -4y_2 + 5y_1 = 21\epsilon, \\ y_4 &= -4y_3 + 5y_2 = -104\epsilon, \\ &\vdots \end{aligned}$$

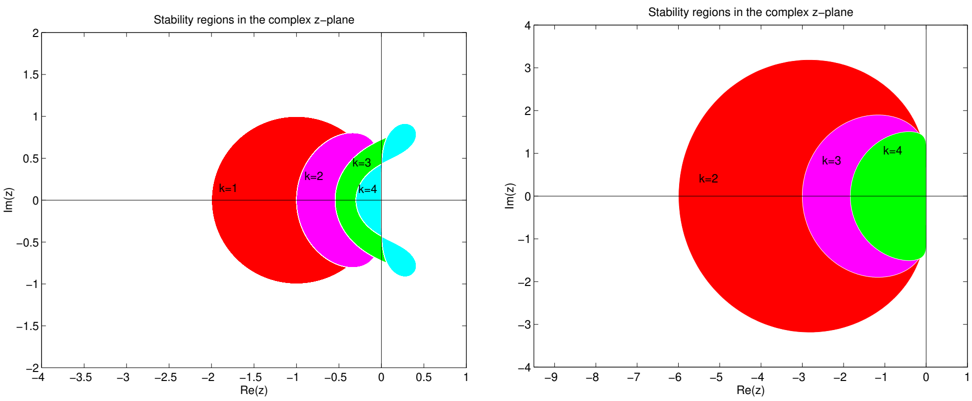
There is no hope for convergence here. ♦

Consider again the test equation, and its discretization (5.9). If $\mathcal{R}\mathrm{e}(\lambda) < 0$ then the exact solution decays and we must prevent any growth in the approximate solution. This is not possible for all such λ if there are extraneous roots of the polynomial ρ with magnitude 1. For $h > 0$ sufficiently small the difference equation (5.9) must be asymptotically stable in this case; see Figure 5.4. We define a linear multistep method to be

- *strongly stable* if all roots of $\rho(\xi) = 0$ are inside the unit circle except for the root $\xi = 1$;
- *weakly stable* if it is 0-stable but not strongly stable.

Example 5.7
Weak stability can be a disaster for some problems, too. Consider Milne's method,

$$y_n = y_{n-2} + \frac{1}{3}h(f_n + 4f_{n-1} + f_{n-2})$$



(a) Adams–Bashforth $k = 1, 2, 3, 4$ (b) Adams–Moulton $k = 2, 3, 4$

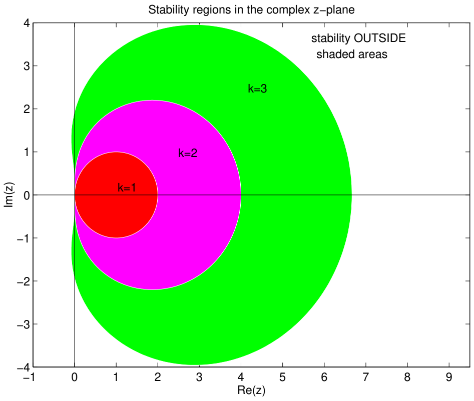
Figure 5.5: *Absolute stability regions of Adams methods.*

- an explicit linear multistep method cannot be A-stable;
- the order of an A-stable linear multistep method cannot exceed 2;
- the second-order A-stable implicit linear multistep method with smallest error constant ($C_3 = \frac{1}{12}$) is the trapezoidal method.

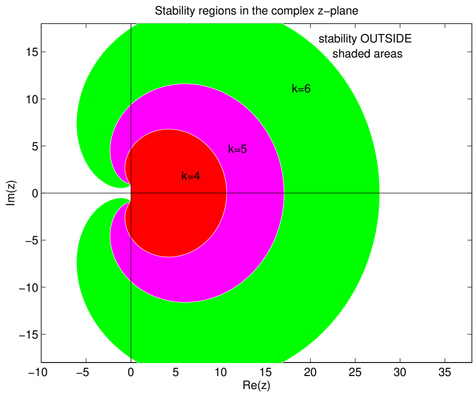
The utility of the trapezoidal method has already been discussed in Section 3.6.[19]

If we want to use linear multistep methods for stiff problems, the A-stability requirement must be relaxed. Moreover, the discussion in Chapter 3 already reveals that in the very stiff limit $h\mathcal{R}\mathrm{e}(\lambda) \to -\infty$, the A-stability bound may not be sufficient and the concept of stiff decay is more useful. The BDF methods introduced in Section 5.1 trade in chunks of absolute stability regions near the imaginary axis for stiff decay. The size of these chunks increases with the number of steps k, until the methods become unstable for $k > 6$; see Figure 5.6.

[19]These results were given by Dahlquist in the 1960s. They had a major impact on research in this area in the 1960s and 1970s. Today it is still easy to appreciate their mathematical beauty and the sophistication that went into the proofs, even though a glance at the methods used in successful implementations makes it clear that A-stability is *not* the property that separates the winners from the also-rans.



(a) BDF $k = 1, 2, 3$



(b) BDF $k = 4, 5, 6$

Figure 5.6: *BDF absolute stability regions. The stability regions are outside the shaded area for each method.*

5.4 Implementation of Implicit Linear Multistep Methods

When using an implicit, k-step linear multistep method, i.e., $\beta_0 \neq 0$ in the formula

$$\sum_{j=0}^{k} \alpha_j \mathbf{y}_{n-j} = h \sum_{j=0}^{k} \beta_j \mathbf{f}_{n-j},$$

a system of m nonlinear equations for $\mathbf{y}_n$ has to be solved at each step (recall Section 3.4). We can solve this system by some variant of functional iteration (for nonstiff systems), or by a modified Newton iteration (for stiff systems). For any of these iterative methods we must "guess," or *predict* a starting iterate $\mathbf{y}_n^0$, usually by evaluating an interpolant passing through past values of $\mathbf{y}$ and/or $\mathbf{f}$ at t_n, or via an explicit multistep method.

5.4.1 Functional Iteration

The simplest way to solve the nonlinear algebraic system for $\mathbf{y}_n$ is via functional iteration. The iteration is given by[20]

$$\mathbf{y}_n^{\nu+1} = h\beta_0 \mathbf{f}(t_n, \mathbf{y}_n^{\nu}) - \sum_{j=1}^{k} \alpha_j \mathbf{y}_{n-j} + h \sum_{j=1}^{k} \beta_j \mathbf{f}_{n-j}, \quad s = 0, 1, \ldots . \quad (5.12)$$

This is a fixed point iteration. It converges to the fixed point $\mathbf{y}_n$ if it is a contraction, i.e., if $\|h\beta_0 \frac{\partial \mathbf{f}}{\partial \mathbf{y}}\| \leq r < 1$. Hence it is appropriate only for

[20]Do not confuse the notation $\mathbf{y}_n^{\nu+1}$ for the $(\nu+1)$st iterate of $\mathbf{y}_n$ with the notation for the $(\nu+1)$st power. Which of these is the correct interpretation should be clear from the context. We have reserved the superscript ν for an iteration counter in this chapter.

where σ, r, b are positive parameters. Following Lorenz we set $\sigma = 10$, $b = 8/3$, $r = 28$ and integrate starting from $\mathbf{y}(0) = (0, 1, 0)^T$. Plotting y_3 vs. y_1 we obtain the famous "butterfly" depicted in Figure 5.7.

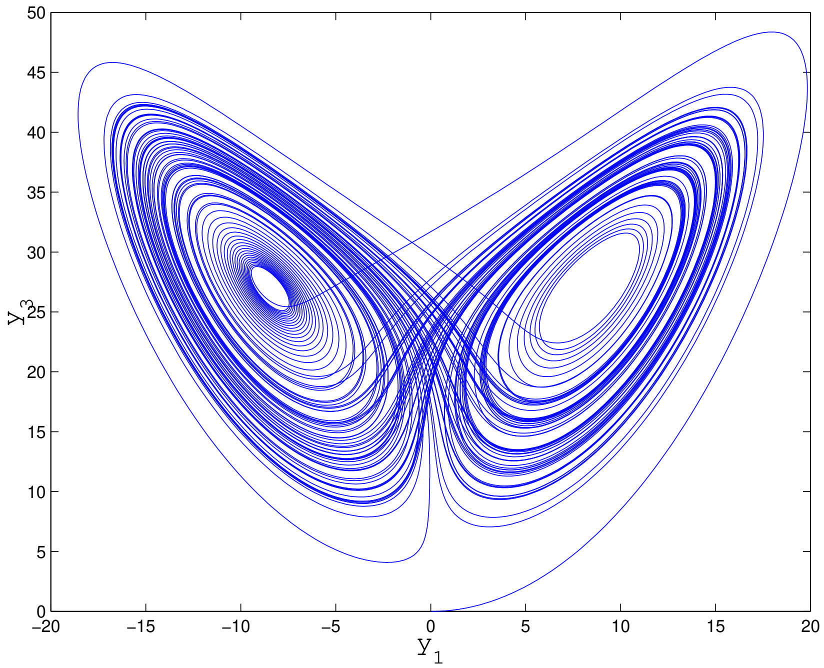


Figure 5.7: *Lorenz "butterfly" in the $y_1 \times y_3$ plane.*

(a) Using a software package of your choice, integrate these equations for $0 \le t \le 100$ with an error tolerance 1.e $-$ 6, and plot y_3 vs. y_1, as well as y_2 as a function of t. What do you observe?

(b) Plot the resulting trajectory in the three-dimensional phase space (i.e., the three y-coordinates; if in MATLAB, type `help plot3`). Observe the *strange attractor* that the trajectory appears to settle into.

(c) Chaotic solutions are famous for their highly sensitive dependence on initial data. This leads to *unpredictability* of the solution (and the physical phenomena it represents). When solving numerically we also expect large errors to result from the numerical discretization. Recompute your trajectory with the same initial data using the same package, changing only the error tolerance to 1.e $-$ 7. Compare the values of $\mathbf{y}(100)$ for the two computations, as well as the plots in phase plane. Discuss.

5.6. (a) Show that the only k-step method of order k which has the stiff decay property is the k-step BDF method.

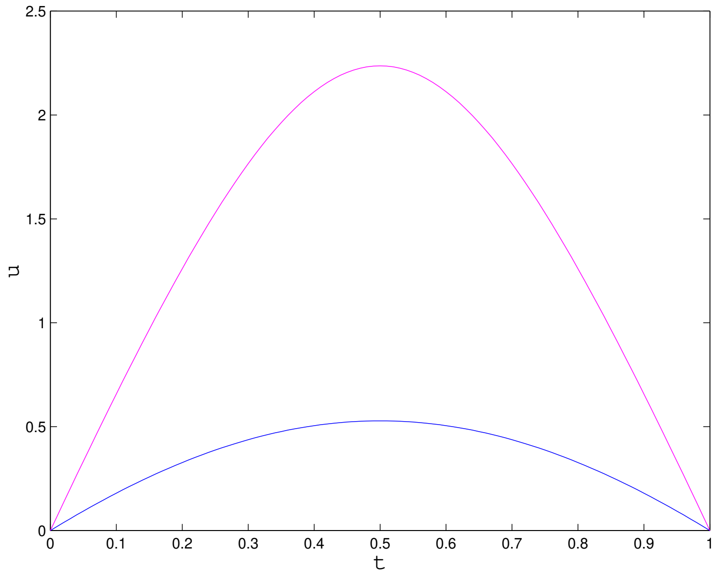


Figure 6.1: *Two solutions $u(t)$ for the BVP of Example* 6.2.

has two solutions of the form

$$u(t) = -2\ln\left\{\frac{\cosh[(t-1/2)\theta/2]}{\cosh(\theta/4)}\right\},$$

where θ satisfies

$$\theta = \sqrt{2e}\cosh(\theta/4)\,.$$

This nonlinear algebraic equation has two solutions for θ (Exercise 6.1). The corresponding two solutions of the BVP are plotted in Figure 6.1. ♦

The possibility of having more than one solution does not in itself prevent us from expecting to be able to find them. The question of existence of unique solutions for a nonlinear BVP must be considered in a local sense. The important question is whether a BVP solution is *isolated*, i.e., if there is a neighborhood about it in which it is the only solution. For this purpose we look at the *variational problem* for the BVP (6.1)–(6.2): assuming for simplicity of notation that $\mathbf{g}$ is linear, i.e., that the boundary conditions are in the form (6.4), the variational problem corresponding to linearizing the problem about an exact solution $\mathbf{y}(t)$ is

$$\begin{aligned} \mathbf{z}' &= A(t,\mathbf{y}(t))\mathbf{z}, \\ B_0\mathbf{z}(0) + B_b\mathbf{z}(b) &= \mathbf{0}, \end{aligned} \tag{6.5}$$

where $A = \frac{\partial \mathbf{f}}{\partial \mathbf{y}}$ is the Jacobian matrix. Now, if the variational problem has the unique solution $\mathbf{z} = \mathbf{0}$ then the solution $\mathbf{y}(t)$ of the given nonlinear problem is isolated, or locally unique. We will show this claim following (6.14)

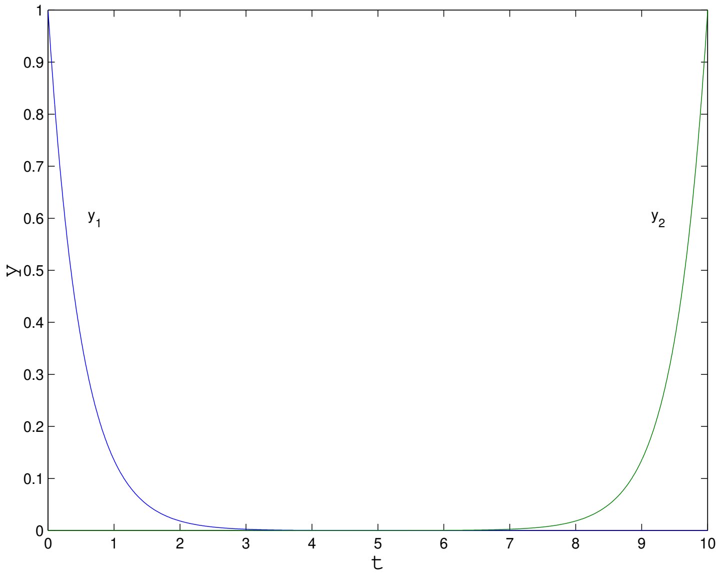


Figure 6.2: *The function* $y_1(t)$ *and its mirror image* $y_2(t) = y_1(b-t)$, *for* $\lambda = -2$, $b = 10$.

where $G(t,s)$ is the *Green's function*

$$G(t,s) = \begin{cases} \Phi(t)B_0\Phi(0)\Phi^{-1}(s), & s \le t\ , \\ -\Phi(t)B_b\Phi(b)\Phi^{-1}(s), & s > t. \end{cases} \tag{6.12}$$

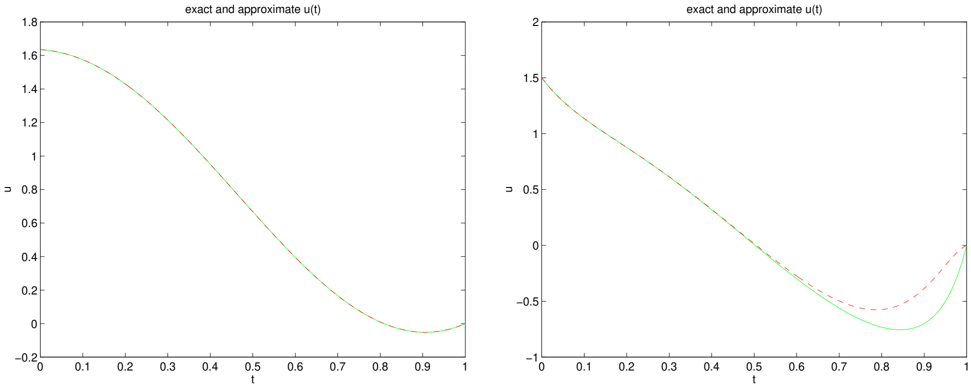
Green's function may be loosely viewed as the inverse of the differential operator (or as the solution operator).

6.2 Stability of BVPs

To understand the fundamental issues in stability for BVPs, the reader must be familiar with the rudiments of stability of IVPs. Therefore, please make sure that you are familiar with the contents of Chapter 2.

Consider the test equation $y_1' = \lambda y_1$ for $0 < t < b$ and regard b as very large. The IVP (e.g., $y_1(0) = 1$) is stable if $\mathcal{R}\mathrm{e}(\lambda) \le 0$. Now apply a variable transformation $\tau = b - t$. The same problem in τ then reads $\frac{dy_2}{d\tau} = -\lambda y_2$, with $y_2(b) = 1$; i.e., this is a terminal value problem and we are integrating from b to 0 (see Figure 6.2). Of course, reversing the direction of time does not affect the stability, which has to do with the effect of small changes in the data on the solution, so this terminal value problem is stable as well (for $\mathcal{R}\mathrm{e}(-\lambda) \ge 0$). Putting the two together, we obtain that the following BVP is stable:

$$\mathbf{y}' = A\mathbf{y}, \quad A = \begin{pmatrix} \lambda & 0 \\ 0 & -\lambda \end{pmatrix},$$



(a) $\lambda = 1$ (b) $\lambda = 10$

Figure 7.1: *Exact (solid line) and shooting (dashed line) solutions for Example* 7.2.

has the exact solution

$$\mathbf{y}(t) = (u(t), u'(t), u''(t))^T, \quad u(t) = \frac{e^{\lambda(t-1)} + e^{2\lambda(t-1)} + e^{-\lambda t}}{2 + e^{-\lambda}} + \cos \pi t$$

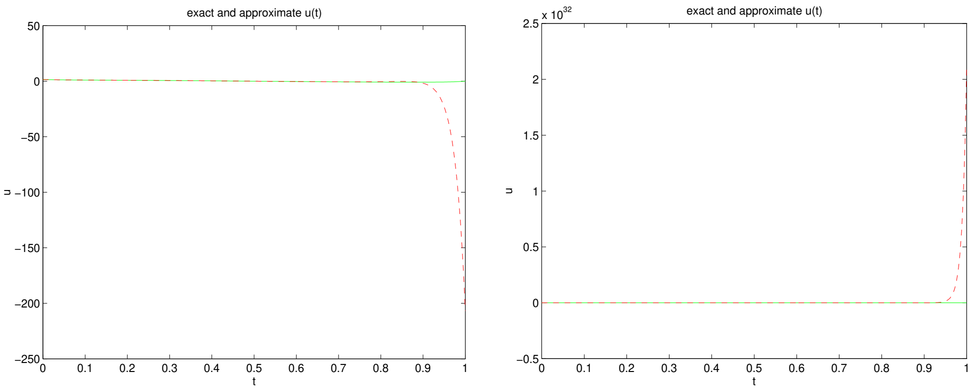
(you can evaluate the expressions and values for $\mathbf{q}(t) = \mathbf{y}'(t) - A(t)\mathbf{y}(t)$ and the boundary values $\mathbf{b}$ from this exact solution). The problem is in the form (7.7) with

$$B_0 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \qquad B_b = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

For $\lambda = 20$, say, the BVP is stable but the IVP is not.

In Figures 7.1 and 7.2 we display the exact and approximate solutions (solid and dashed lines, respectively) for various values of λ ranging from a harmless 1 to a tough 50. We use the classical Runge–Kutta method of order 4 with a fixed step size $h = .004$ and a 14-hexadecimal-digit floating point arithmetic.

Note that the disastrous effect observed is due to the propagation of errors in the obtained initial values $\mathbf{c}$ by unstable modes (recall Example 2.1). The error in $\mathbf{c}$ is unavoidable and is due to *roundoff*, not truncation errors. We have chosen the discretization step size so small, in fact, that for the case $\lambda = 1$ the errors in the initial values vector are all below 10^{-9}, as is the maximum error in u in the ensuing integration for the approximate solution of the BVP. For $\lambda = 10$, already an $O(1)$ error is observed (the maximum error in u is .34). This case may be regarded as particularly worrisome, because the wrong solution obtained for a moderate value of λ may also look plausible. For $\lambda = 20$, this maximum error is 206.8 (although the error



(a) $\lambda = 20$ (b) $\lambda = 50$

Figure 7.2: *Exact (solid line) and shooting (dashed line) solutions for Example* 7.2.

in the initial conditions is only less than 10^{-4}); for $\lambda = 50$, the overall error in u is 2.1e + 32, and in the initial conditions it is about 10^8.

The instability is already extreme for $\lambda = 20$, a value for which the BVP is not very stiff. ♦

Another potential difficulty with the simple shooting method arises for nonlinear problems. The method assumes that the IVPs encountered will have solutions, even for inaccurate initial values, that reach all the way to $t = b$. For nonlinear problems, however, there is no guarantee that this would be the case. Initial value solutions with incorrect initial values are typically guaranteed to exist *locally* in t, but not necessarily globally. For another potential difficulty with the nonlinear iteration, see Exercise 7.7.

7.2 Multiple Shooting

Both disadvantages of the simple shooting method become worse for larger intervals of integration of the IVPs. In fact, a rough bound on the propagation error, which is approximately achieved in Example 7.2, is e^{Lb}, where $L = \max_t \|A(t)\|$. The basic idea of multiple shooting is then to restrict the size of intervals over which IVPs are integrated. Defining a mesh

$$0 = t_0 < t_1 < \cdots < t_{N-1} < t_N = b,$$

we consider approximating the solution of the ODE system $\mathbf{y}' = \mathbf{f}(t, \mathbf{y})$ by constructing an approximate solution on each subinterval $[t_{n-1}, t_n]$ and patching these approximate solutions together to form a global one (see Figure 7.3). Thus, let $\mathbf{y}_n(t; \mathbf{c}_{n-1})$ be the solution of the IVP

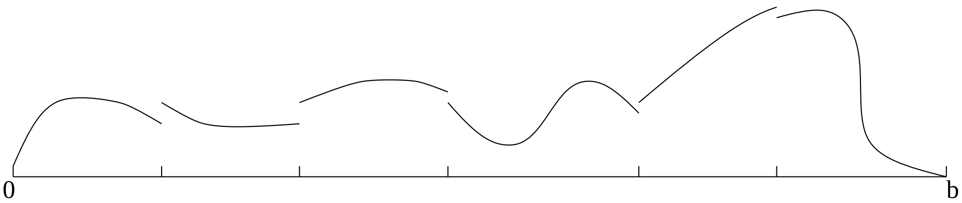


Figure 7.3: *Multiple shooting.*

$$\mathbf{y}_n' = \mathbf{f}(t, \mathbf{y}_n), \qquad t_{n-1} < t < t_n, \tag{7.4a}$$
$$\mathbf{y}_n(t_{n-1}) = \mathbf{c}_{n-1} \tag{7.4b}$$

for $1 \le n \le N$.[30] Assuming for the moment that the IVPs (7.4) are solved exactly, we then have that the exact solution of the problem (7.1)–(7.2) satisfies

$$\mathbf{y}(t) = \mathbf{y}_n(t; \mathbf{c}_{n-1}), \qquad t_{n-1} \le t \le t_n, \ 1 \le n \le N$$

if

$$\mathbf{y}_n(t_n; \mathbf{c}_{n-1}) = \mathbf{c}_n, \qquad 1 \le n \le N-1, \tag{7.5a}$$
$$\mathbf{g}(\mathbf{c}_0, \mathbf{y}_N(b; \mathbf{c}_{N-1})) = \mathbf{0}. \tag{7.5b}$$

The conditions (7.5a) are *patching* conditions which ensure that $\mathbf{y}(t)$ patched from the different pieces $\mathbf{y}_n(t; \mathbf{c}_{n-1})$ is continuous on the entire interval $[0, b]$, and (7.5b) is just the resulting expression for the boundary conditions (7.2).

The conditions (7.5) give Nm algebraic equations for the Nm coefficients

$$\mathbf{c} = (\mathbf{c}_0^T, \mathbf{c}_1^T, \dots, \mathbf{c}_{N-1}^T)^T.$$

We write these equations, as before, as

$$\mathbf{h}(\mathbf{c}) = \mathbf{0}. \tag{7.6}$$

Applying Newton's method to solve the nonlinear equations (7.6) results at each iteration ν in a system of linear equations which can be viewed as arising from the same multiple shooting method for the linearized BVP

$$\mathbf{A}(\mathbf{c}^{\nu+1} - \mathbf{c}^\nu) = -\mathbf{h}(\mathbf{c}^\nu),$$

where $\mathbf{A} = \frac{\partial \mathbf{h}}{\partial \mathbf{c}}(\mathbf{c}^\nu)$ has a sparse block structure, as in (7.10) below. An advantage of Newton's method here (not shared by quasi-Newton methods) is that this sparse block structure remains intact during the iteration process.

[30] It is important not to confuse this notation with what is used in the finite difference chapters 3, 4, 5, and 8 for a slightly different purpose. Here $\mathbf{y}_n$ is meant to be the *exact* solution on a subinterval $[t_{n-1}, t_n]$, provided we can find the right $\mathbf{c}_{n-1}$.

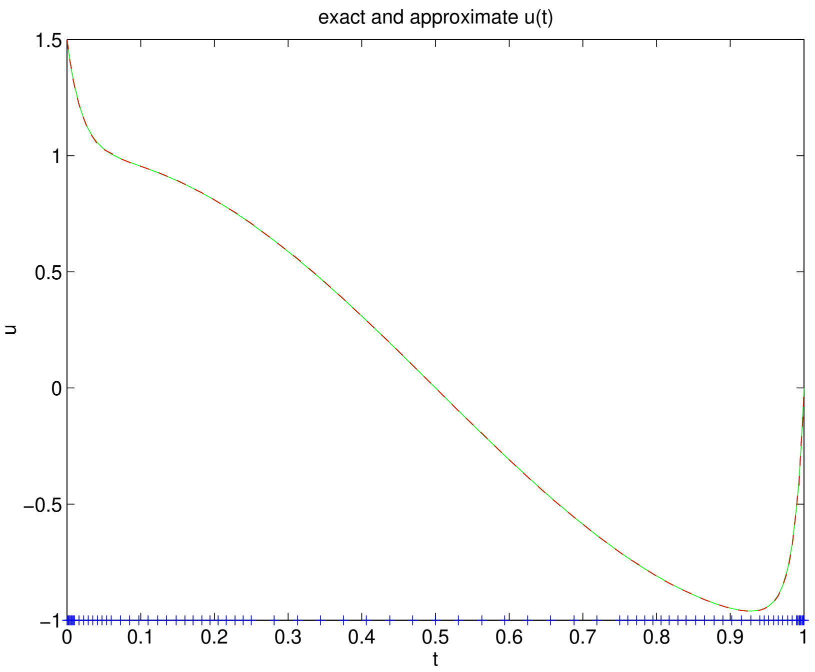


Figure 8.1: *Example* 8.1: *Exact and approximate solutions (indistinguishable) for* $\lambda = 50$, *using the indicated mesh.*

measured only at mesh points away from the layer.

Even with these simple nonuniform meshes, a significant improvement in the quality of the solution is obtained. The exact and approximate solutions for $\lambda = 50$ are plotted in Figure 8.1, together with the mesh that was used to generate these curves. This mesh corresponds to the last entry of Table 8.2 ($N = 80$). The approximate solution is in agreement with the exact one, as far as the eye can tell. It turns out that for this type of problem it is possible to construct more sophisticated meshes on which we obtain good, accurate solutions for any $\lambda \geq 1$ with N independent of λ. This is in contrast to multiple shooting techniques, where N grows linearly with λ. ♦

For solving the many nonlinear algebraic equations we again consider Newton's method, because it is basic, it is fast when it works well, and it retains the sparsity structure of the Jacobian, which is important for such a large system. As it turns out, Newton's method applied to the midpoint equations (8.3)–(8.4) is equivalent to the method of quasi-linearization coupled with the midpoint discretization for linear problems. The latter approach has the attraction of being more modular, so we describe it next.

8.1.1 Solving Nonlinear Problems: Quasi-Linearization

Newton's method for algebraic equations is obtained by expanding in Taylor series and truncating the nonlinear terms at each iteration. The quasi-

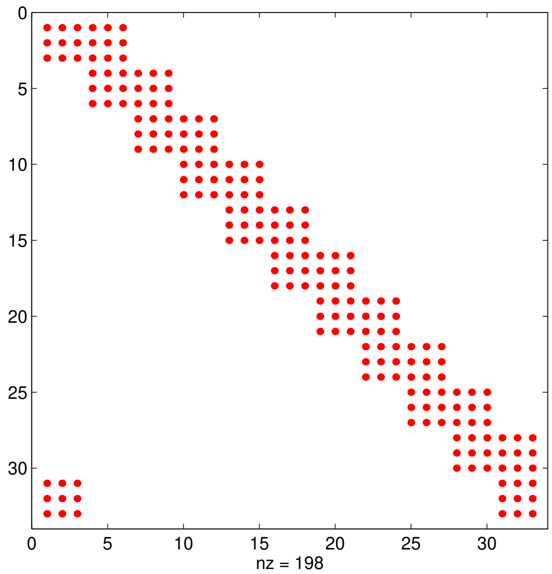


Figure 8.2: *Zero-structure of the matrix* $\mathbf{A}$, $m = 3, N = 10$. *The matrix size is* $m(N+1) = 33$.

permuted accordingly as well. This also establishes a "time" direction—the lower is the row in the permuted $\mathbf{A}$, the larger is t to which it refers. In Figure 8.3 we depict the permuted $\mathbf{A}$ corresponding to Figure 8.2, where two boundary conditions are prescribed at $t = 0$ and one at $t = b$.

A number of methods which require $O(Nm^3)$ flops (instead of the usual $O(N^3m^3)$) to solve the linear system (8.14) in the case of separated boundary conditions have been proposed in the literature. Here we describe the simplest and crudest of these and only comment on other methods.

Once permuted, the matrix $\mathbf{A}$ can be considered as banded, with $m+k-1$ diagonals below the main diagonal and $2m - k - 1$ diagonals above the main diagonal possibly having nonzero entries. Outside this total of $3m - 1$ diagonals, all entries of $\mathbf{A}$ are 0. Gaussian elimination with partial pivoting extends to the banded case in a straightforward fashion. Simply, two of the three nested loops defining the elimination process are shortened in order to not eliminate elements known to be 0 at the start. The fill-in is only within the banded structure, with the addition of a few diagonals due to the partial pivoting. It is not difficult to write a program to carry out this algorithm. Also, there exists standard software to do this, e.g., in LINPACK or LAPACK.

If you look at the band containing all nonzero elements in the matrix depicted in Figure 8.3, you will notice that there are triangles of zeros within the band for each interior mesh point. These zeros are not taken advantage of in the band method just described. Other, more sophisticated methods for solving (8.14) attempt to avoid, or at least minimize, fill-in of these triangles, thereby achieving an additional savings of up to 50% in both storage and computational efficiency.

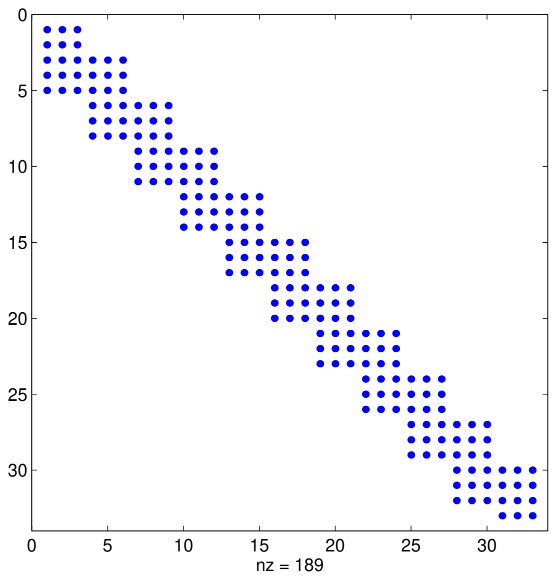


Figure 8.3: *Zero-structure of the permuted matrix* $\mathbf{A}$ *with separated boundary conditions,* $m = 3, k = 2, N = 10$.

The fact that all the nonzeros of $\mathbf{A}$ densely populate a narrow band as in Figure 8.3 is typical for boundary value ODEs, where neighboring elements (i.e., subintervals sharing unknowns) can be ordered consecutively. For boundary value partial differential equations (PDEs), on the other hand, the band is necessarily much wider and the matrix is sparse inside the band as well. Variants of Gaussian elimination become less effective then, and iterative methods like preconditioned conjugate gradients and multigrid take center stage.

8.3 Higher-Order Methods

The midpoint and trapezoidal methods may be considered as basic methods. The only problem in using them as they are for many applications is that they are only second-order accurate. There are two types of higher-order methods extending the basic methods: higher-order Runge–Kutta and acceleration techniques. The overview picture is given in Figure 8.4.

8.3.1 Collocation

One class of extensions to higher-order methods is simply higher-order implicit Runge–Kutta methods. Continuing to prefer symmetric methods, this leads to collocation methods at Gauss or Lobatto points. We have already considered the basic properties of these methods in Chapter 4, and everything else pertaining to BVPs extends in a very similar way to the treatment in the previous two sections. We summarize without repeating the proof:

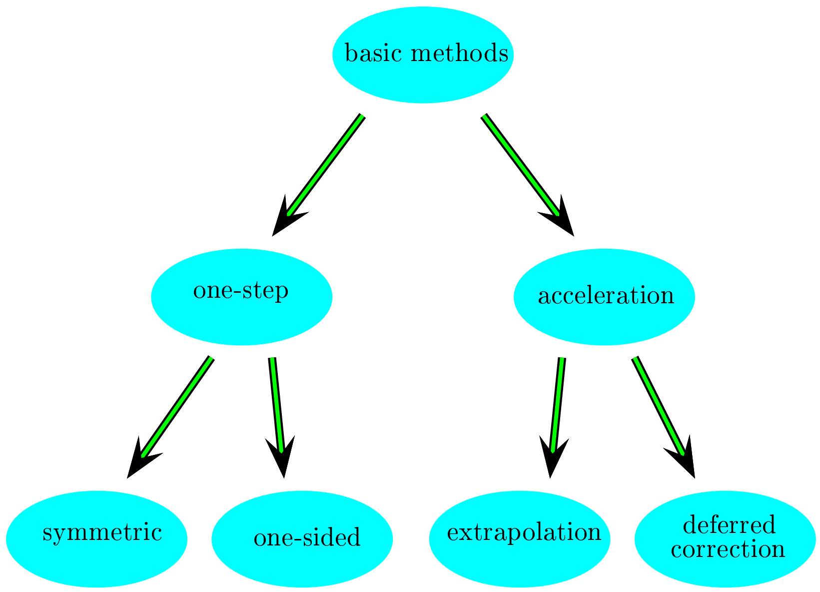


Figure 8.4: *Classes of higher-order methods.*

- Collocation at s Gaussian points is 0-stable with a numerical stability constant satisfying (8.13). It converges with the error bound

$$|\mathbf{e}_n| \le K \max_j |\mathbf{d}_j| = O(h^{2s}), \qquad 0 \le n \le N. \qquad (8.15)$$

Example 8.3
We repeat the computations of Example 8.1 using collocation at three Gaussian points per mesh element. The results are recorded in Tables 8.3 and 8.4.

The errors for $\lambda = 1$ in Table 8.3 reflect the fact that this method is of order 6 and has a nice error constant to boot. For $\lambda = 50$ the errors are fairly good even on the uniform mesh, although they are better on the nonuniform meshes (whose construction is discussed in Example 8.1). For $\lambda = 500$ a nonuniform mesh is certainly needed; see Table 8.4. In fact, a better layer mesh can become useful as well in order to retrieve the full convergence order (which turns out to be 4 for $h|\lambda| \gg 1$) that this method has outside the layer regions. ♦

Applying quasi-linearization to nonlinear problems and considering collocation for the resulting linear ones, we obtain a linear system of equations for the mesh unknowns as well as the internal stages (see (4.8) or (4.10)).

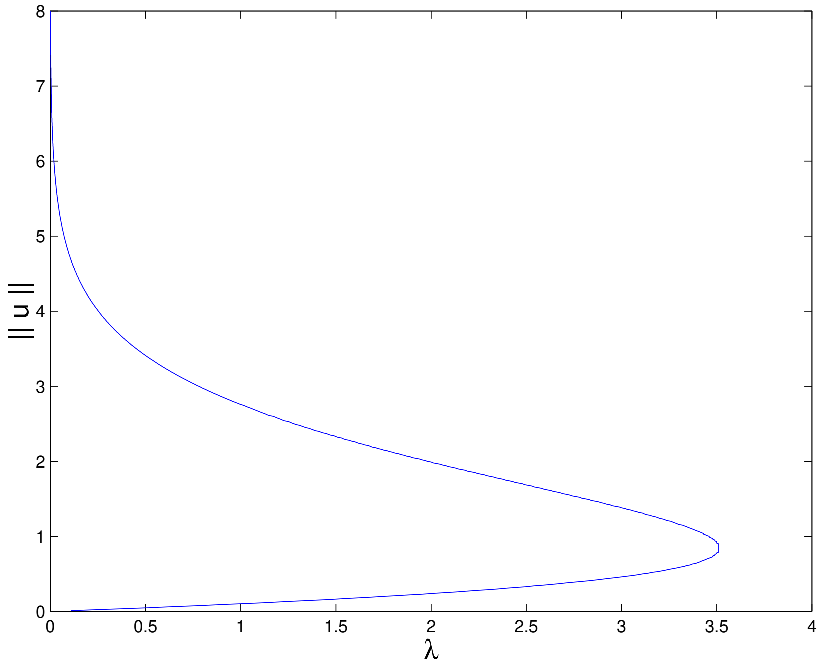


Figure 8.5: *Bifurcation diagram for Example* 8.5: $\|u\|_2$ *vs.* λ*.*

proceeds from $t = 0$ to $t = b$. To capitalize on this, IVP solvers often abandon controlling the global error $\mathbf{e}_n$ (although this is usually what the user may want) and control the local truncation error $\mathbf{d}_n$ instead.

In the BVP case there is no compelling reason, in general, not to estimate the global error $\mathbf{e}_n$. Such an estimate is compared against user-specified tolerances or used to select a new mesh. The process in overview for a given BVP is to discretize and solve it on a sequence of meshes, where the error in the solution on the current mesh is estimated and this information is used to decide what the next mesh should be, in case there is deemed to be a need for a next mesh. The first mesh is a guess.

The error estimation can be achieved using a process similar to the one described for extrapolation methods in Section 8.3.2. For instance, given a midpoint or a trapezoidal solution $\{\mathbf{y}_n\}$ on a mesh π and another one $\{\tilde{\mathbf{y}}_j\}$ on a mesh obtained by subdividing each element of π into two halves, we have

$$\tilde{\mathbf{y}}_{2n} - \mathbf{y}_n = \frac{3}{4}\mathbf{c}h^2 + O(h^4)\,.$$

So

$$\mathbf{e}_n = \mathbf{y}(t_n) - \mathbf{y}_n \approx \frac{4}{3}(\tilde{\mathbf{y}}_{2n} - \mathbf{y}_n)$$

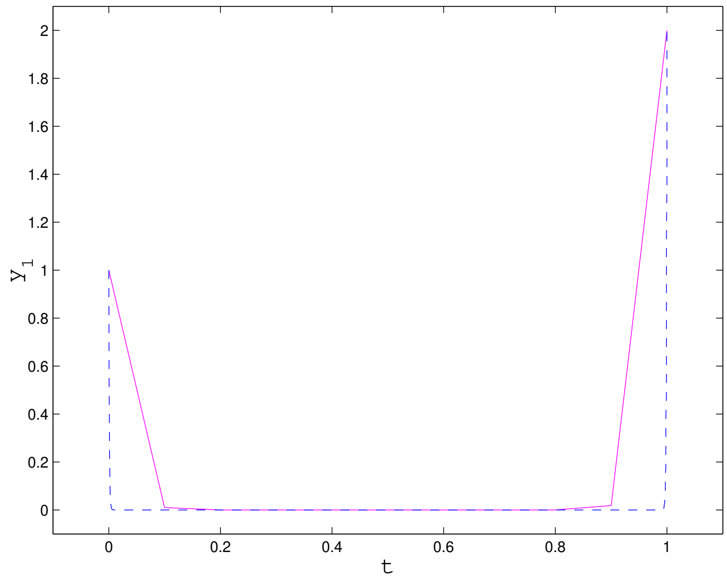


Figure 8.6: *Solution for Example* 8.6 *with* $\lambda = -1000$ *using an upwind discretization with a uniform step size* $h = 0.1$ *(solid line). The "exact" solution is also displayed (dashed line).*

Symmetric Discretization

For symmetric difference methods no explicit decoupling transformation is usually needed. But, as mentioned earlier, the lack of stiff decay can be evident in computations. For the test equation

$$y' = \lambda y$$

the midpoint method yields, to recall,

$$y_n = \frac{2 + h\lambda}{2 - h\lambda} y_{n-1}.$$

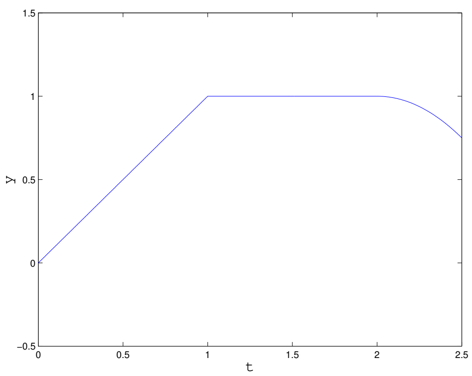
So, although $|y_n| \leq |y_{n-1}|$ precisely whenever the exact solution satisfies $|y(t_n)| \leq |y(t_{n-1})|$, as $h|\lambda| \to \infty$ we get

$$y_n \approx -y_{n-1} \, .$$

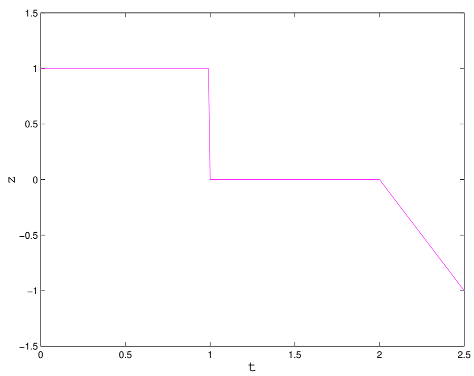
Thus, if $y(0) = 1$ and $h\mathcal{R}\mathrm{e}(\lambda) \ll -1$, then the exact solution satisfies $y(h) \approx 0$, yet for the numerical solution,

$$y_n \approx (-1)^n, \qquad n = 1, \ldots, N \, .$$

This not only necessitates covering layer regions (where the solution varies rapidly) with dense meshes, as we have already seen in Examples 3.2 and 8.1, it also means that in the very stiff case local errors propagate through smooth



(a) $y(t)$



(b) $z(t) = y'(t)$

Figure 9.1: *A function and its less smooth derivative.*

Thus, integration is a *smoothing* process, while differentiation is an *antismoothing* process. The differentiation process is in a sense unstable,[38] although it is often very simple to carry out analytically.

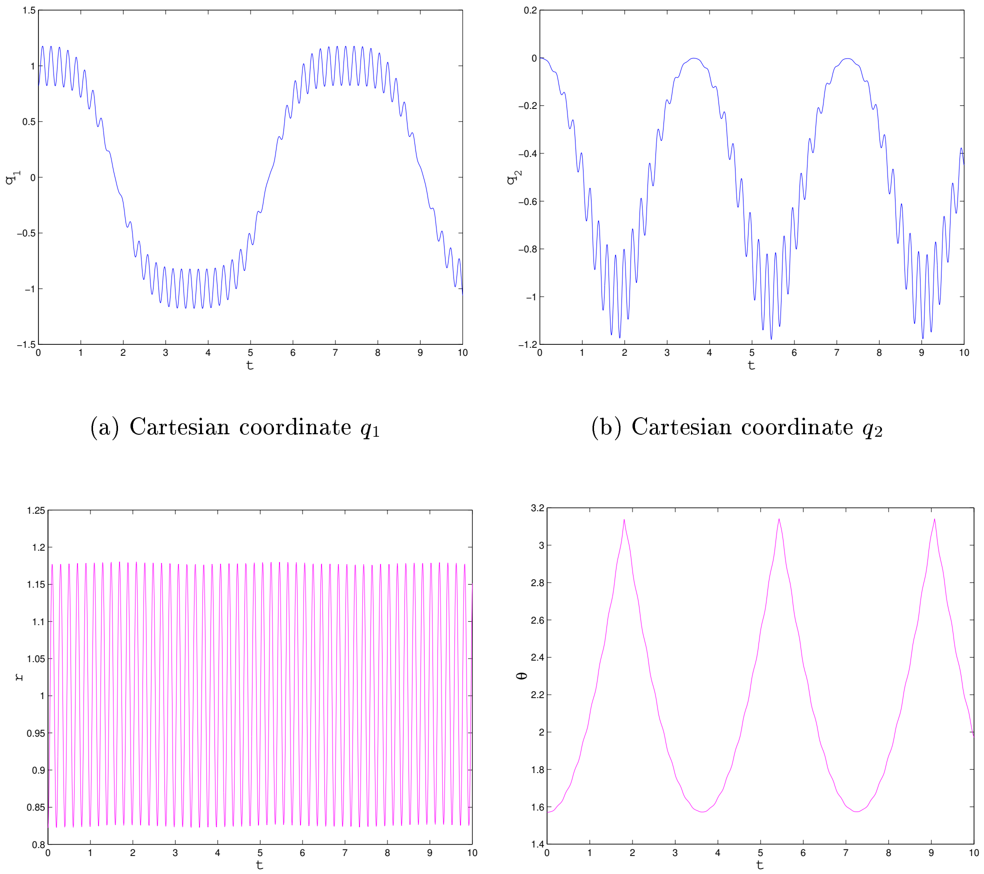
A differential equation involves integration, hence smoothing: the solution $\mathbf{y}(t)$ of the linear system $\mathbf{y}' = A\mathbf{y} + \mathbf{q}(t)$ is one derivative smoother than $\mathbf{q}(t)$. A DAE, on the other hand, involves both differentiations and integrations. The class of DAEs contains all ODEs, as well as the problems in Example 9.1 below. But it also contains problems where both differentiations and integrations are intertwined in a complex manner, and that's when the fun really starts: simple differentiations may no longer be possible, but their effect complicates the numerical integration process, potentially well beyond what we have seen so far in this book.

9.1 Index and Mathematical Structure

Since a DAE involves a mixture of differentiations and integrations, one may hope that applying analytical differentiations to a given system and eliminating as needed, repeatedly if necessary, will yield an explicit ODE system for all the unknowns. This turns out to be true unless the problem is singular. The number of differentiations needed for this transformation is called the *index* of the DAE. Thus, ODEs have index 0. We will refine this definition later, but first let us consider some simple examples.

Example 9.1
Let $q(t)$ be a given, smooth function, and consider the following problems for $\mathbf{y}(t)$.

[38] If we add to $y(t)$ a small perturbation $\epsilon \cos \omega t$, where $|\epsilon| \ll 1$ and $\omega > |\epsilon^{-1}|$, then $z(t)$ is perturbed by a large amount $|\omega\epsilon|$.



(a) Cartesian coordinate q_1

(b) Cartesian coordinate q_2

(c) Radius r

(d) Angle θ

Figure 9.2: *Stiff spring pendulum,* $\varepsilon = 10^{-3}$, *initial conditions* $\mathbf{q}(0) = (1 - \varepsilon^{1/4}, 0)^T, \mathbf{v}(0) = \mathbf{0}$.

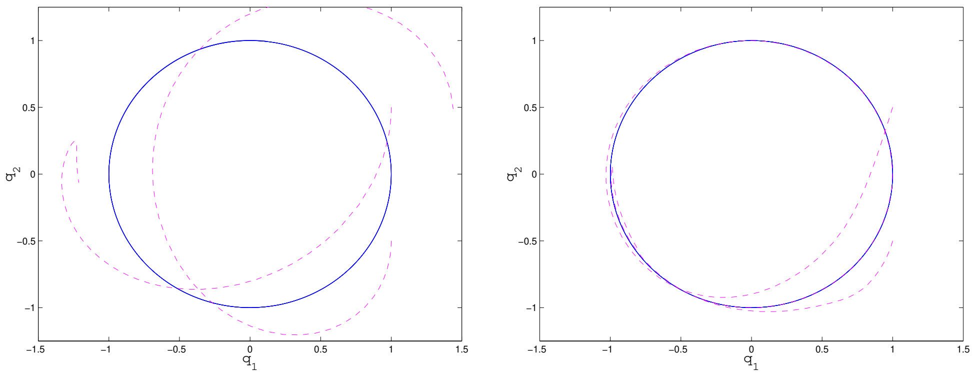
Provided that the initial conditions yield

$$r(t) = 1 + O(\varepsilon),$$

we have $\lambda(t) = O(1)$ to balance the constraint equation in the index-1 formulation. The passage to the limit $\varepsilon \to 0$ is simple then, and we obtain the DAE

$$\begin{aligned} \mathbf{q}'' &= -\mathbf{q}\lambda - \begin{pmatrix} 0 \\ g \end{pmatrix}, \\ 0 &= r - 1, \end{aligned}$$

which gives the equations for the simple pendulum of Examples 1.5 and



(a) Unstabilized pendulum equations

(b) Stabilized pendulum equations, $\gamma = 10$

Figure 9.3: *Perturbed (dashed lines) and unperturbed (solid line) solutions for Example* 9.9.

The parameters γ_1 and γ_2 are chosen such that the polynomial equation

$$x^2 + \gamma_1 x + \gamma_2 = 0$$

has two negative roots; thus the ODE (9.43) for $\mathbf{g}$ is stable. This stabilizes the invariant set $\mathcal{M}$. The system (9.30a), (9.30b), (9.43) is a semi-explicit index-1 DAE. It can be made into an ODE upon elimination of $\boldsymbol{\lambda}$ and be subsequently solved numerically by standard methods. But the choice of the parameters has proved tricky in practice. Exercise 9.11 and Section 10.2 elaborate more on this.

9.2.3 State Space Formulation

The differentiations of the constraints of the given high-index DAE (9.30) yield an ODE (9.34) with an inflated dimension, as we have seen. Even though the number of degrees of freedom of the system is $m - 2l$, we have in (9.34) m ODEs, and in (9.35) an additional $2l$ algebraic equations. Rather than stabilizing the invariant, another approach is to use these algebraic equations to define a reduced set of unknowns, obtaining an ODE system of the minimal size $m - 2l$. The main difficulty with this idea arises in the presence of highly nonlinear terms.

Suppose that R is a rectangular, constant matrix such that, together with the constraint Jacobian G, we obtain a nonsingular matrix with a bounded inverse:

$$\left\| \begin{pmatrix} R \\ G \end{pmatrix}^{-1} \right\| \le K\,. \tag{9.44}$$

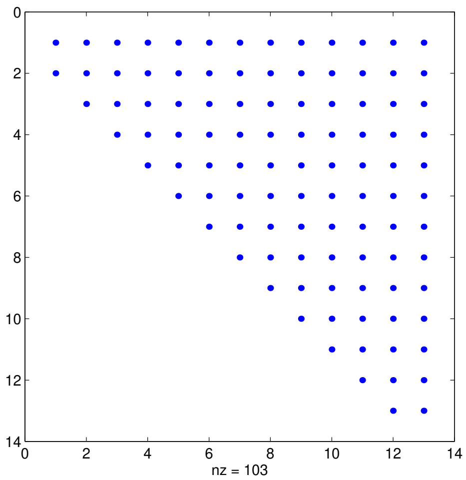


Figure 9.4: *A matrix in Hessenberg form.*

(a) Write the ODE in first-order form for the variables $y_1 = u$ and $y_2 = \varepsilon u' - au$.

(b) Letting $\varepsilon \to 0$, show that the limit system is an index-1 DAE.

(c) Show that only one of the boundary conditions in (9.47) is needed to determine the reduced solution (i.e., the solution of the DAE). Which one?

9.3. Consider the DAE [49]

$$\begin{aligned} y_1' &= y_3 y_2' - y_2 y_3', \\ 0 &= y_2, \\ 0 &= y_3 \end{aligned}$$

with $y_1(0) = 1$.

(a) Show that this DAE has index 1.

(b) Show that if we add to the right-hand side the perturbation

$$\boldsymbol{\delta}(t) = (0, \epsilon \sin \omega t, \epsilon \cos \omega t)^T,$$

which is bounded in norm by a small ϵ, the perturbed solution $\hat{\mathbf{y}}(t)$ satisfies $\hat{y}_1' = \epsilon^2 \omega$, which is unbounded as $\omega \to \infty$. The stability bound is seen to depend on $\boldsymbol{\delta}'$, as is typical for index-2 rather than index-1 problems.

(c) Show that if we add a similar perturbation to the linearization around the solution $\mathbf{y}(t)$ for $\mathbf{z} = (z_1, z_2, z_3)^T$:

$$\begin{aligned} z_1' &= z_3 y_2' + y_3 z_2' - z_2 y_3' - y_2 z_3', \\ 0 &= z_2, \\ 0 &= z_3, \end{aligned}$$

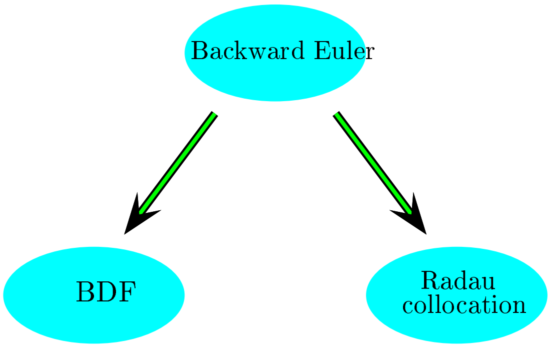


Figure 10.1: *Methods for the direct discretization of DAEs in general form.*

The rest of this section is therefore devoted to the direct application of ODE methods to low-index DAEs. All the winning methods have stiff decay, but this property alone is not sufficient. For initial value DAEs which are cumbersome to transform, and especially for DAEs whose underlying ODE is stiff, the backward differentiation formulae (BDF) and Radau collocation methods discussed in this section are the overall methods of choice. We thus begin with the simplest method of this kind, the backward Euler method, and then consider its extension to higher order via BDF or Radau methods; see Figure 10.1.

10.1.1 A Simple Method: Backward Euler

Consider the general DAE

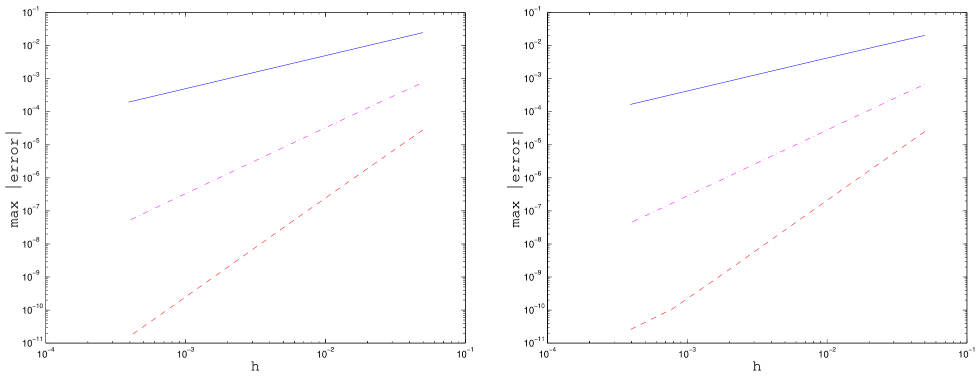
$$\mathbf{0} = \mathbf{F}(t, \mathbf{y}, \mathbf{y}')\,.$$

The idea of a direct discretization is simple: approximate $\mathbf{y}$ and $\mathbf{y}'$ by a discretization formula like multistep or Runge–Kutta. Applying the backward Euler method to this DAE, we obtain

$$\mathbf{0} = \mathbf{F}\left(t_n, \mathbf{y}_n, \frac{\mathbf{y}_n - \mathbf{y}_{n-1}}{h_n}\right). \tag{10.5}$$

This gives, in general, a system of m nonlinear equations for $\mathbf{y}_n$ at each time step n.

Unfortunately, this simple method does not always work. In the worst case, there are simple higher-index DAE systems with well-defined solutions for which the backward Euler method, and in fact all other multistep and Runge–Kutta methods, are unstable or not even applicable.



(a) Errors in $x_1(t)$ (b) Errors in $z(t)$

Figure 10.2: *Maximum errors for the first three BDF methods for Example* 10.2.

from $\frac{1}{20}$ to $\frac{1}{2560}$. We use a log-log scale, so the slopes of the curves indicate the orders of the methods. The results clearly indicate that the convergence order of the k-step BDF method is indeed k and that in absolute value the errors are pleasantly small. ♦

10.1.3 Radau Collocation and Implicit Runge–Kutta Methods

Runge–Kutta Methods

The s-stage implicit Runge–Kutta method applied to the general nonlinear DAE of the form (10.1) is defined by

$$\mathbf{0} = \mathbf{F}(t_i, \mathbf{Y}_i, \mathbf{K}_i), \tag{10.13a}$$

$$t_i = t_{n-1} + c_i h, \quad i = 1, 2, \ldots, s, \tag{10.13b}$$

$$\mathbf{Y}_i = \mathbf{y}_{n-1} + h \sum_{j=1}^{s} a_{ij} \mathbf{K}_j, \tag{10.13c}$$

and

$$\mathbf{y}_n = \mathbf{y}_{n-1} + h \sum_{i=1}^{s} b_i \mathbf{K}_i \,. \tag{10.14}$$

We assume here that the coefficient matrix $A = (a_{ij})$ is nonsingular.

For the semi-explicit problem (10.3) the formula (10.13) for the internal

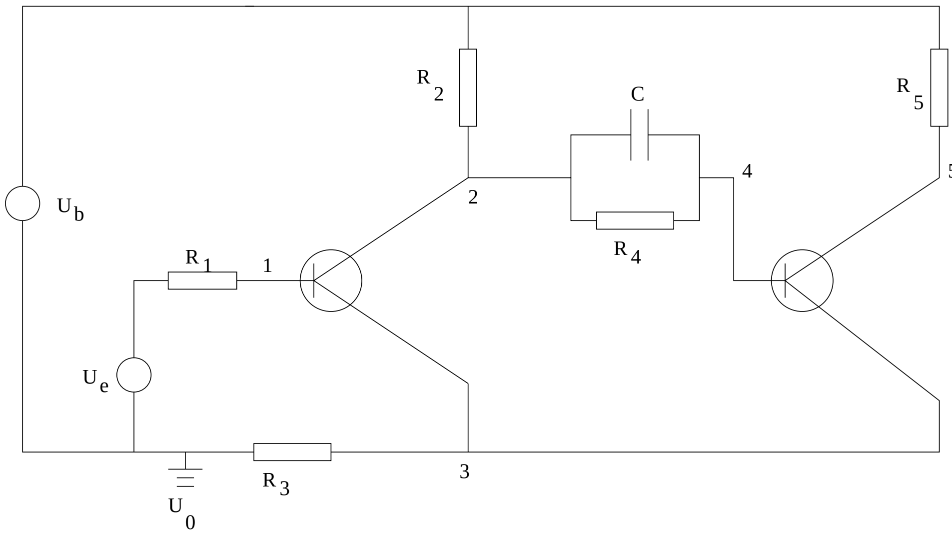


Figure 10.3: *A simple electric circuit.*

Applying Kirchoff's current law at the five nodes in sequence, we get

$$\begin{aligned}
0 &= (U_1 - U_e)/R_1 - (\alpha - 1) f(U_1 - U_3), \\
C(U_2' - U_4') &= (U_B - U_2)/R_2 + (U_4 - U_2)/R_4 - \alpha f(U_1 - U_3), \\
0 &= (U_3 - U_0)/R_3 - f(U_1 - U_3) - f(U_4 - U_3), \\
C(U_4' - U_2') &= (\alpha - 1) f(U_4 - U_3) - (U_4 - U_2)/R_4, \\
0 &= (U_4 - U_b)/R_5 + \alpha f(U_4 - U_3).
\end{aligned}$$

We use the values $U_0 = 0$ (ground voltage), $U_b = 5$, $U_e = 5\sin(2000\pi t)$, $R_1 = 200, R_2 = 1600, R_3 = 100, R_4 = 3200, R_5 = 1600$, $C = 40.\mathrm{e} - 6$. (The potentials are in Volts, the resistances are in Ohms, t is in seconds.)

This is a simple index-1 DAE which, however, has scaling and sensitivity difficulties due to the exponential in the definition of f. We can obviously make it semi-explicit for the differential variable $U_2 - U_4$, but we leave the system in the fully implicit form and apply the collocation code RADAU5. Recall that this code is based on collocation at three Radau points. It applies to ODEs and DAEs of the form

$$M\mathbf{y}' = \tilde{\mathbf{f}}(t, \mathbf{y})$$

(see Section 10.3.2 and Exercise 10.15), and here we have such a form with the constant matrix

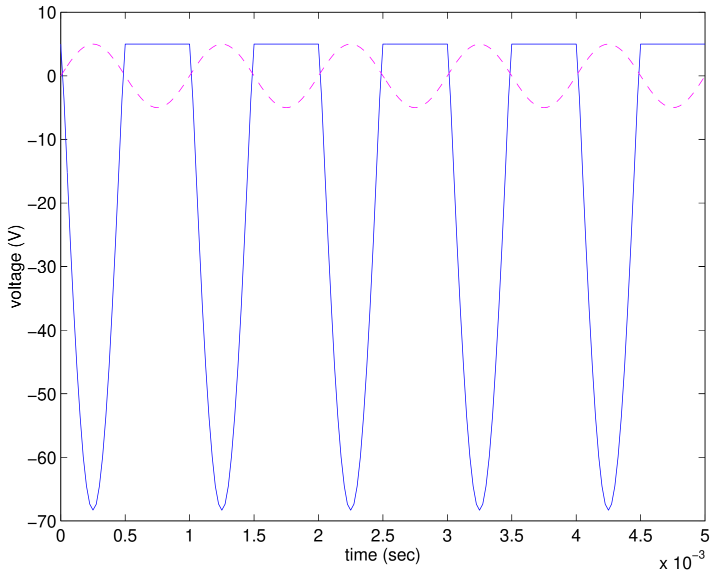


Figure 10.4: *Results for a simple electric circuit: $U_2(t)$ (solid line) and the input $U_e(t)$ (dashed line).*

$$M = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & C & 0 & -C & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & C & 0 & -C & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

For consistent initial conditions, only $U_2(0) - U_4(0)$ are free. The rest are determined by the four algebraic equations. (How four? Three are apparent; the fourth is obtained upon adding up the two equations containing derivatives, which cancels out the derivative term.) A consistent initial vector is given by

$$\mathbf{y}(0) = (0, U_b, 0, 0, U_b)^T.$$

A plot of the resulting U_2 as a function of time, as well as the input signal U_e, is given in Figure 10.4. It is seen that $U_2(t)$ has periods where it becomes large and negative. The solution is not very smooth. The code used 655 steps, of which 348 were accepted, for this simulation. The right-hand side function was evaluated almost 7000 times, but the Jacobian, only 342 times. ♦

10.1.4 Practical Difficulties

Even though there are order and convergence results for the backward Euler method (as well as for BDF and collocation at Radau points) applied to fully

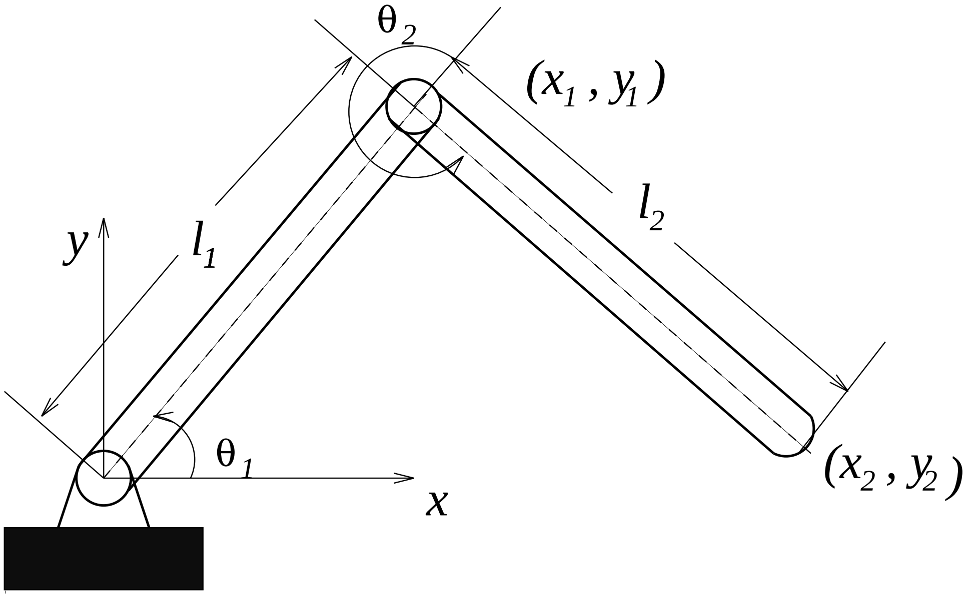


Figure 10.5: *Two-link planar robotic system.*

$\mathbf{q} = (\theta_1, \theta_2)^T$ and obtain

$$M = \begin{pmatrix} m_1 l_1^2/3 + m_2(l_1^2 + l_2^2/3 + l_1 l_2 c_2) & m_2(l_2^2/3 + l_1 l_2 c_2/2) \\ m_2(l_2^2/3 + l_1 l_2 c_2/2) & m_2 l_2^2/3 \end{pmatrix},$$

$$\mathbf{f} = \begin{pmatrix} -m_1 g l_1 c_1/2 - m_2 g(l_1 c_1 + l_2 c_{12}/2) \\ -m_2 g l_2 c_{12}/2 \end{pmatrix} + \begin{pmatrix} m_2 l_1 l_2 s_2/2(2\theta_1'\theta_2' + (\theta_2')^2) \\ -m_2 l_1 l_2 s_2 (\theta_1')^2/2 \end{pmatrix}.$$

In the following simulation we use the data

$$m_1 = m_2 = 36kg, \quad l_1 = l_2 = 1m, \quad g = 9.81m/s^2,$$
$$\theta_1(0) = 70^o, \quad \theta_2(0) = -140^o, \quad \theta_1'(0) = \theta_2'(0) = 0.$$

So far we do not have constraints $\mathbf{g}$. Indeed, for a double pendulum the equations of motion form an implicit ODE (or an index-0 DAE), because the topology of this simple mechanical system has no closed loops and we are using relative (minimal) coordinates to describe the system. But now we prescribe some path constraint on the position of (x_2, y_2), and this yields, in turn, a constraint force $G^T\lambda$ as well. We choose the constraint

$$y_2(t) = \sin^2(t/2)$$

(for y_2 expressed in terms of $\mathbf{q}$ as described above). The obtained constrained path for (x_2, y_2) is depicted in Figure 10.6. In this case the constraint forces become large at a few distinct times.

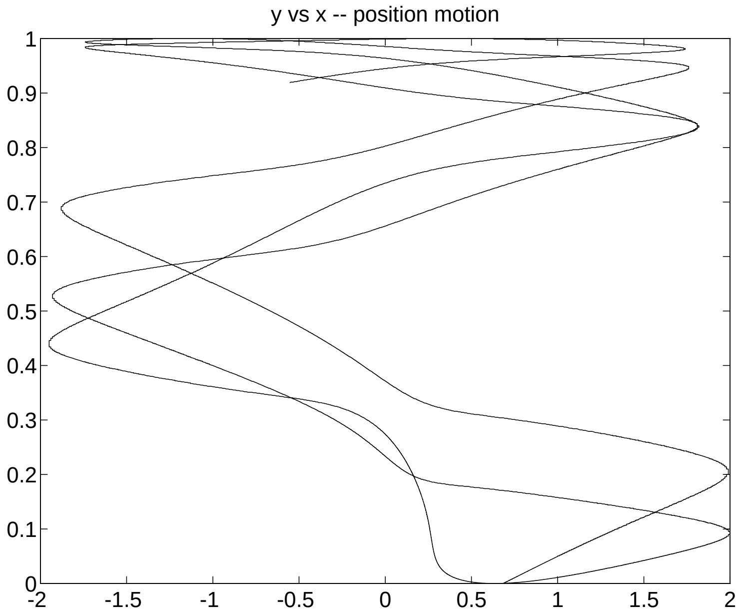


Figure 10.6: *Constraint path for* (x_2, y_2).

In Table 10.2 we record the measured drifts, i.e., the error in the path constraint ("drift-position") and in its derivative ("drift-velocity"), based on runs up to $b = 10s$ using an explicit Runge–Kutta scheme of order 2 with a constant step size h. We record results using Baumgarte's technique (9.43), denoting it "Baum(γ_1, γ_2)," and also using various choices of F for poststabilization: "S-pos" stands for stabilizing only with respect to the position constraints $\mathbf{g}(t, \mathbf{q}) = \mathbf{0}$; "S-vel" stands for stabilizing only with respect to the velocity constraints $\mathbf{g}' = \mathbf{0}$; "S-both" stands for using F of (10.28) once; "S-both^2" is the choice recommended in Algorithm 10.1; and finally "S-full" uses $F = H^T(HH^T)^{-1}$.

Note that without stabilization the computation blows up for $h = 0.01$. The Baumgarte stabilization is not as effective as the S-stabilizations, especially for the case $h = .01$. Other parameters (γ_1, γ_2) tried do not yield significantly better results. The choice of Algorithm 10.1 shows drift-convergence order $6 = 2(p + 1)$ and, given that it is much cheaper than S-full and not much more expensive than the other choices for F, we conclude that S-both^2 gives the most bang for the buck here. ♦

10.3 Software, Notes, and References

10.3.1 Notes

Of course, scientists encountered the need to numerically solve mathematical models involving differential equations with constraints and implicit differen-