Numerical solution of ordinary differential equations

Ernst Hairer and Christian Lubich

Université de Genève and Universität Tübingen

1 Introduction: Euler methods

Ordinary differential equations are ubiquitous in science and engineering: in geometry and mechanics from the first examples onwards (Newton, Leibniz, Euler, Lagrange), in chemical reaction kinetics, molecular dynamics, electronic circuits, population dynamics, and many more application areas. They also arise, after semi-discretization in space, in the numerical treatment of time-dependent partial differential equations, which are even more impressively omnipresent in our technologically developed and financially controlled world.

The standard initial value problem is to determine a vector-valued function $y : [t_0, T] \to \mathbb{R}^d$ with a given initial value $y(t_0) = y_0 \in \mathbb{R}^d$ such that the derivative $y'(t)$ depends on the current solution value $y(t)$ at every $t \in [t_0, T]$ in a prescribed way:

$$y'(t) = f(t, y(t)) \quad \text{for} \quad t_0 \le t \le T, \quad y(t_0) = y_0.$$

Here, the given function f is defined on an open subset of $\mathbb{R} \times \mathbb{R}^d$ containing (t_0, y_0) and takes values in $\mathbb{R}^d$. If f is continuously differentiable then there exists a unique solution at least locally on some open interval containing t_0. In many applications, t represents time, and it will be convenient to refer to t as time in what follows.

In spite of ingenious efforts of mathematicians throughout the 18th and 19th century, in most cases the solution of a differential equation cannot be given in closed form by functions that can be evaluated directly on a computer. This even applies to linear differential equations $y' = Ay$ with a square matrix A, for which $y(t) = e^{(t-t_0)A} y_0$, as computing the matrix exponential is a notoriously tricky problem. One therefore must rely on numerical methods that are able to approximate the solution of a differential equation to any desired accuracy.

1.1 The explicit Euler method

The ancestor of all the advanced numerical methods in use today was formed by Leonhard Euler in 1768. On writing down the first terms in the Taylor expansion of the solution at t_0 and using the prescribed initial value and the differential equation at $t = t_0$, it is noted that $y(t_0 + h) = y(t_0) + hy'(t_0) + \cdots = y_0 + hf(t_0, y_0) + \cdots$. Choosing a small step size $h > 0$ and neglecting the higher-order terms represented by the dots, an approximation y_1 to $y(t_1)$ at the later time $t_1 = t_0 + h$ is obtained by setting

$$y_1 = y_0 + hf(t_0, y_0).$$

The next idea is to take y_1 as the starting value for a further step, which then yields an approximation to the solution at $t_2 = t_1 + h$ as $y_2 = y_1 + hf(t_1, y_1)$. Continuing in this way, in the $(n+1)$st step we take $y_n \approx y(t_n)$ as the starting value for computing an approximation at $t_{n+1} = t_n + h$ as

$$y_{n+1} = y_n + hf(t_n, y_n),$$

and after a sufficient number of steps we reach the final time T. The computational cost of the method lies in the evaluations of the function f. The step size need not be the same in each step and could be replaced by h_n in the formula, so that $t_{n+1} = t_n + h_n$.

It is immediate that the quality of the approximation y_n depends on two aspects: the error made by truncating the Taylor expansion and the error introduced by continuing from approximate solution values. These two aspects are captured in the notions of *consistency* and *stability*, respectively, and are fundamental to all numerical methods for ordinary differential equations.

1.2 The implicit Euler method and stiff differential equations

A minor-looking change in the method, already considered by Euler in 1768, makes a big difference: taking as the argument of f the new value instead of the previous one yields

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1}),$$

from which y_{n+1} is now determined implicitly. In general, the new solution approximation needs to

be computed iteratively, typically by a modified Newton method such as $y_{n+1}^{(k+1)} = y_{n+1}^{(k)} + \Delta y_{n+1}^{(k)}$, where the increment is computed by solving a linear system of equations

$$(I - hJ_n)\Delta y_{n+1}^{(k)} = -r_{n+1}^{(k)}$$

with an approximation J_n to the Jacobian matrix $\partial_y f(t_n, y_n)$ and the residual $r_{n+1}^{(k)} = y_{n+1}^{(k)} - y_n - hf(t_{n+1}, y_{n+1}^{(k)})$. The computational cost per step has increased dramatically: whereas the explicit Euler method requires a single function evaluation we now need to compute the Jacobian and then solve a linear system and evaluate f on each Newton iteration.

Why it may nevertheless be preferable to perform the computation using the implicit rather than the explicit Euler method is evident for the scalar linear example, made famous by Germund Dahlquist in 1963,

$$y' = \lambda y,$$

where the coefficient λ is large and negative (or complex with large negative real part). Here the exact solution $y(t) = e^{(t-t_0)\lambda} y_0$ decays to zero as time increases, and so does the numerical solution given by the implicit Euler method *for every step size* $h > 0$:

$$y_n^{\text{impl}} = (1 - h\lambda)^{-n} y_0.$$

In contrast, the explicit Euler method yields

$$y_n^{\text{expl}} = (1 + h\lambda)^n y_0,$$

which decays to zero for growing n only when h is so small that $|1 + h\lambda| < 1$. This imposes a severe step size restriction when λ is a negative number of large absolute value (just think of $\lambda = -10^{10}$). For larger step sizes the numerical solution suffers an instability that is manifested in wild oscillations of increasing amplitude. The problem is that the explicit Euler method and the differential equation have completely different stability behaviors unless the step size is chosen extremely small (see Figure 1).

Such behavior is not restricted to the simple scalar example considered above, but extends to linear systems of differential equations in which

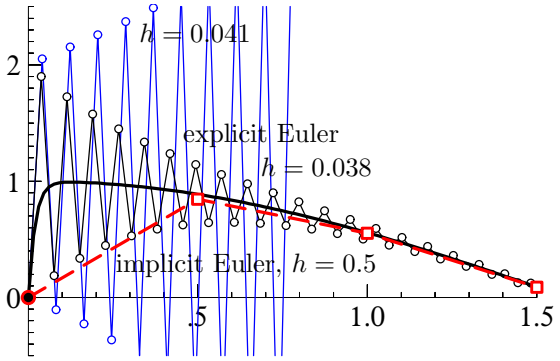

Figure 1: Exact solution (solid), implicit Euler solution (dashed), and two explicit Euler solutions (oscillating) for the problem $y' = -50(y - \cos t)$, $y(0) = 0$.

the matrix has some eigenvalues with large negative real part, and to classes of nonlinear differential equations with a Jacobian matrix $\partial_y f$ having this property. The explicit and implicit Euler methods also give rise to very different behaviors for nonlinear differential equations in which the function $f(t, y)$ has a large (local) Lipschitz constant L with respect to y, while for some inner product the inequality

$$\langle f(t, y) - f(t, z), y - z \rangle \le \ell \|y - z\|^2$$

holds for all t and y, z with a moderate constant $\ell \ll L$ (called a one-sided Lipschitz constant).

Differential equations for which the numerical solution using the implicit Euler method is more efficient than that using the explicit Euler method are called *stiff* differential equations. They include important applications in the description of processes with multiple time scales (e.g., fast and slow chemical reactions) and in spatial semi-discretizations of time-dependent partial differential equations. For example, for the heat equation, stable numerical solutions are obtained with the explicit Euler method only when temporal step sizes are bounded by the square of the spatial grid size, whereas the implicit Euler method is unconditionally stable.

1.3 The symplectic Euler method and Hamiltonian systems

An important class of differential equations where neither the explicit nor the implicit Euler method is appropriate is *Hamiltonian differential equations*

$$p' = -\nabla_q H(p,q), \quad q' = +\nabla_p H(p,q),$$

which are fundamental to many branches of physics. Here, the real-valued Hamilton function H, defined on a domain of $\mathbb{R}^{d+d}$, represents the total energy and $q(t) \in \mathbb{R}^d$ and $p(t) \in \mathbb{R}^d$ represent the positions and momenta, respectively, of a conservative system at time t. The total energy is conserved:

$$H(p(t), q(t)) = \text{Const.}$$

along any solution $(p(t), q(t))$ of the Hamiltonian system. It turns out that a partitioned method obtained by applying the explicit Euler method to the position variables and the implicit Euler method to the momentum variables (or *vice versa*) behaves much better than either Euler method applied to the system as a whole. The *symplectic Euler method* reads

$$\begin{aligned} p_{n+1} &= p_n - h\nabla_q H(p_{n+1}, q_n) \\ q_{n+1} &= q_n + h\nabla_p H(p_{n+1}, q_n). \end{aligned}$$

For a separable Hamiltonian $H(p,q) = T(p) + V(q)$ the method is explicit.

Figure 2 illustrates the qualitative behavior of the three Euler methods applied to the differential equations of the mathematical pendulum,

$$p' = -\sin q, \quad q' = p,$$

which are Hamiltonian with $H(p,q) = \frac{1}{2}p^2 - \cos q$. The energy of the implicit Euler solution decreases, while that of the explicit Euler solution increases. The symplectic Euler method nearly conserves the energy over extremely long times.

2 Basic notions

In this section we describe some of the mechanisms that lead to the different behaviors of the various methods.

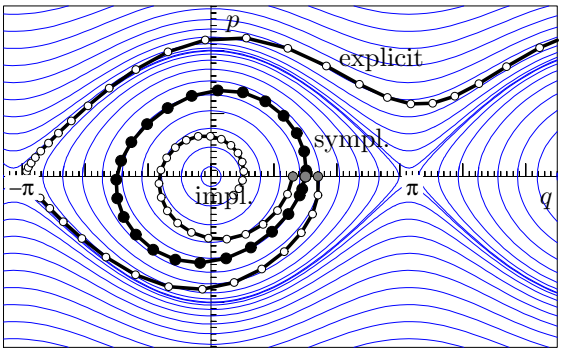

Figure 2: Pendulum equation: Euler polygons with step size $h = 0.3$; initial value $p(0) = 0$ and $q(0) = 1.7$ for explicit Euler, $q(0) = 1.5$ for symplectic Euler, and $q(0) = 1.3$ for implicit Euler. The solid lines are solution curves for the differential equations.

2.1 Local error

For the explicit Euler method, the error after one step of the method starting from the exact solution, called the *local error*, is given as

$$d_{n+1} = \big(y(t_n) + hf(t_n, y(t_n))\big) - y(t_n + h).$$

By estimating the remainder term in the Taylor expansion of $y(t_n+h)$ at t_n, we can bound d_{n+1} by

$$\|d_{n+1}\| \le Ch^2 \quad \text{with} \quad C = \tfrac{1}{2} \max_{t_0 \le t \le T} \|y''(t)\|,$$

provided that the solution is twice continuously differentiable, which is the case if f is continuously differentiable.

2.2 Error propagation

Since the method advances in each step with the computed values y_n instead of the exact solution values $y(t_n)$, it is important to know how errors, once introduced, are propagated by the method. Consider explicit Euler steps starting from different starting values,

$$\begin{aligned} u_{n+1} &= u_n + hf(t_n, u_n), \\ v_{n+1} &= v_n + hf(t_n, v_n). \end{aligned}$$

When f is (locally) Lipschitz continuous with Lipschitz constant L, the difference is controlled by the *stability estimate*

$$\|u_{n+1} - v_{n+1}\| \le (1 + hL)\|u_n - v_n\|.$$

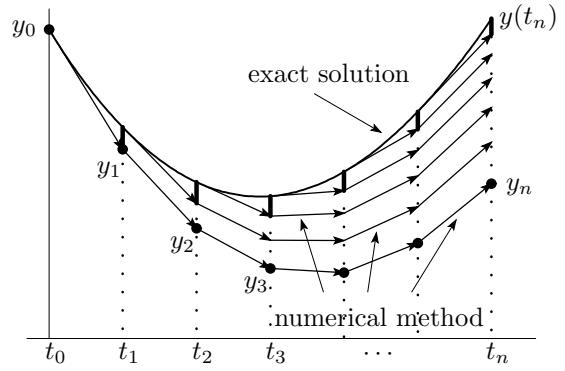

Figure 3: Lady Windermere's fan.

2.3 Lady Windermere's fan

The above estimates can be combined to study the error accumulation, as illustrated by the fan of Figure 3 (named by Gerhard Wanner in the 1980s after a play by Oscar Wilde). Each arrow from left to right represents a step of the numerical method, with different starting values. The fat vertical bars represent the local errors, whose propagation by the numerical method is controlled using repeatedly the stability estimate from step to step: the *global error*

$$e_n = y_n - y(t_n)$$

is the sum of the propagated local errors (represented as the distances between two adjacent arrow heads ending at t_n in Figure 3). The contribution of the first local error d_1 to the global error e_n is bounded by $(1+hL)^{n-1}\|d_1\|$, as is seen by applying the stability estimate $n-1$ times following the numerical solutions starting from y_1 and $y(t_1)$. The contribution of the second local error d_2 is bounded by $(1+hL)^{n-2}\|d_2\|$, and so on. Since the local errors are bounded by Ch^2, the global error is thus bounded by

$$\begin{aligned}\|e_n\| &\leq \sum_{j=0}^{n-1}(1+hL)^j Ch^2 \\ &= \frac{(1+hL)^n - 1}{1+hL-1}\,Ch^2 \leq \frac{e^{nhL}-1}{L}\,Ch.\end{aligned}$$

With $M = (e^{(T-t_0)L} - 1)C/L$, the global error satisfies

$$\|e_n\| \leq Mh \quad \text{for} \quad t_n \leq T.$$

The numerical method thus converges to the exact solution as $h \to 0$ with nh fixed, but only at first order, that is, with an error bound proportional to h. We will later turn to higher-order numerical methods, with an error bound proportional to h^p with $p > 1$.

2.4 Stiff differential equations

The above error bound becomes meaningless for stiff problems, where L is large. The implicit Euler method admits an analogous error analysis in which only the one-sided Lipschitz constant ℓ appears in the stability estimate: provided that $h\ell < 1$,

$$\|u_{n+1} - v_{n+1}\| \leq \frac{1}{1-h\ell}\,\|u_n - v_n\|$$

holds for the results of two Euler steps starting from u_n and v_n. For stiff problems with $\ell \ll L$ this is much more favorable than the stability estimate of the explicit Euler method in terms of L. It leads to an error bound $\|e_n\| \leq mh$ in which m is essentially of the same form as M above, but with the Lipschitz constant L replaced by the one-sided Lipschitz constant ℓ.

The above arguments explain the convergence behavior of the explicit and implicit Euler methods and their fundamentally different behavior for large classes of stiff differential equations. They do not explain the favorable behavior of the symplectic Euler method for Hamiltonian systems. This requires another concept, backward analysis, which is treated next.

2.5 Backward analysis

Much insight into numerical methods is obtained by interpreting the numerical result after a step as the (almost) exact solution of a *modified differential equation*. Properties of the numerical method can then be inferred from properties of a differential equation. For each of the Euler methods applied to $y' = f(y)$ an asymptotic expansion

$$\widetilde{f}(\widetilde{y}) = f(\widetilde{y}) + hf_2(\widetilde{y}) + h^2 f_3(\widetilde{y}) + \cdots$$

can be uniquely constructed recursively such that, up to arbitrarily high powers of h,

$$y_1 = \widetilde{y}(t_1),$$

where $\widetilde{y}(t)$ is the solution of the modified differential equation $\widetilde{y}' = \widetilde{f}(\widetilde{y})$ with initial value y_0. The remarkable feature is that when the symplectic Euler method is applied to a Hamiltonian system, then the modified differential equation is again Hamiltonian. The modified Hamilton function has an asymptotic expansion

$$\widetilde{H} = H + hH_2 + h^2 H_3 + \cdots .$$

The symplectic Euler method therefore conserves the modified energy $\widetilde{H}$ (up to arbitrarily high powers of h), which is close to the original energy H. This conservation of the modified energy prevents the linearly growing drift in the energy that is present along numerical solutions of the explicit and implicit Euler methods. For these two methods the modified differential equation is no longer Hamiltonian.

3 Nonstiff problems

3.1 Higher-order methods

A method is said to have *order* p if the local error (recall that this is the error after one step of the method starting from the exact solution) is bounded by Ch^{p+1}, where h is the step size and C depends only on bounds of derivatives of the solution $y(t)$ and of the function f. Like for the Euler method in Section 2.1, the order is determined by comparing the Taylor expansions of the exact and the numerical solution, which for a method of order p should agree up to and including the h^p term.

A drawback of the Euler methods is that they are only of order 1. There are different ways to increase the order: using additional, auxiliary function evaluations in passing from y_n to y_{n+1} (one-step methods); using previously computed solution values $y_{n-1}, y_{n-2}, \ldots$ and/or their function values (multistep methods); or using both (general linear methods). For nonstiff initial value problems the most widely used methods are explicit Runge–Kutta methods of orders up to 8, in the class of one-step methods, and Adams-type multistep methods up to order 12. For very stringent accuracy requirements of 10 or 100 digits, high-order extrapolation methods or high-order Taylor series expansions of the solution (when higher derivatives of f are available with automatic differentiation software) are sometimes used.

3.2 Explicit Runge–Kutta methods

Two ideas underlie Runge–Kutta methods: first, the integral in

$$y(t_0 + h) = y(t_0) + h \int_0^1 y'(t_0 + \theta h)\, d\theta,$$

with $y'(t) = f(t, y(t))$, is approximated by a quadrature formula with weights b_i and nodes c_i,

$$y_1 = y_0 + h \sum_{i=1}^{s} b_i Y_i', \qquad Y_i' = f(t_0 + c_i h, Y_i).$$

Second, the internal stage values $Y_i \approx y(t_0 + c_i h)$ are determined by another quadrature formula for the integral from 0 to c_i:

$$Y_i = y_0 + h \sum_{j=1}^{s} a_{ij} Y_j', \quad i = 1, \ldots, s,$$

with the *same* function values Y_j' as for y_1. If the coefficients satisfy $a_{ij} = 0$ for $j \geq i$, then the above sum actually extends only from $j = 1$ to $i - 1$, and hence $Y_1, Y_1', Y_2, Y_2', \ldots, Y_s, Y_s'$ can be computed explicitly one after the other. The methods are named after Carl Runge, who in 1895 proposed two- and three-stage methods of this type, and after Wilhelm Kutta, who in 1901 proposed what is now known as the *classical Runge–Kutta method* of order 4, which extends the Simpson quadrature rule from integrals to differential equations:

$$\begin{array}{c|c} c_i & a_{ij} \\ \hline & b_j \end{array} \equiv \begin{array}{c|cccc} 0 & & & & \\ 1/2 & 1/2 & & & \\ 1/2 & 0 & 1/2 & & \\ 1 & 0 & 0 & 1 & \\ \hline & 1/6 & 2/6 & 2/6 & 1/6 \end{array}$$

Using Lady Windermere's fan as in Section 2, one finds that the global error $y_n - y(t_n)$ of a pth-order Runge–Kutta method over a bounded time interval is $\mathcal{O}(h^p)$.

The order conditions of general Runge–Kutta methods were elegantly derived by John Butcher

in 1964 using a tree model for the derivatives of f and their concatenations by the chain rule, as they appear in the Taylor expansions of the exact and the numerical solutions. This enabled the construction of even higher-order methods, among which excellently constructed methods of orders 5 and 8 by Dormand and Prince (from 1980) have found widespread use. These methods are equipped with lower-order error indicators from embedded formulas that use the same function evaluations. These error indicators are used for an adaptive selection of the step size that is intended to keep the local error close to a given error tolerance in each time step.

3.3 Extrapolation methods

A systematic, if sub-optimal construction of explicit Runge–Kutta methods of arbitrarily high order is provided by *Richardson extrapolation* of the results of the explicit Euler method obtained with different step sizes. This technique makes use of an asymptotic expansion of the error,

$$y(t,h) - y(t) = e_1(t)h + e_2(t)h^2 + \cdots ,$$

where $y(t,h)$ is the explicit Euler approximation at t obtained with step size h. At $t = t_0 + H$, the error expansion coefficients up to order p can be eliminated by evaluating at $h = 0$ (*extrapolating*) the interpolation polynomial through the Euler values $y(t_0 + H, h_j)$ for $j = 1, \ldots, p$ corresponding to different step sizes $h_j = H/j$. This gives a method of order p, which formally falls into the general class of Runge–Kutta methods. Instead of using the explicit Euler method as basic method, it is preferable to take *Gragg's method* (from 1964) which uses the explicit midpoint rule

$$y_{n+1} = y_{n-1} + 2hf(t_n, y_n), \quad n \geq 1,$$

and an explicit Euler starting step to compute y_1. This method has an error expansion in powers of h^2 (instead of h above) at even n, and with the elimination of each error coefficient one therefore gains a power of h^2. Extrapolation methods have built-in error indicators that can be used for order and step-size control.

3.4 Adams methods

The methods introduced by Astronomer Royal John Couch Adams in 1855 were the first of high order that use only function evaluations, and in their current variable-order, variable step-size implementations they are among the most efficient methods for general nonstiff initial value problems.

When k function values $f_{n-j} = f(t_{n-j}, y_{n-j})$ $(j = 0, \ldots, k-1)$ have already been computed, the integrand in

$$y(t_{n+1}) = y(t_n) + h \int_0^1 f(t_n + \theta h, y(t_n + \theta h))\, d\theta$$

is approximated by the interpolation polynomial $P(t)$ through (t_{n-j}, f_{n-j}), $j = 0, \ldots, k-1$, yielding the *explicit Adams method* of order k,

$$y_{n+1} = y_n + h \int_0^1 P(t_n + \theta h)\, d\theta,$$

which, upon inserting the Newton interpolation formula, becomes (for constant step size h)

$$y_{n+1} = y_n + hf_n + h \sum_{i=1}^{k-1} \gamma_i \nabla^i f_n,$$

with the backward differences $\nabla f_n = f_n - f_{n-1}$, $\nabla^i f_n = \nabla^{i-1} f_n - \nabla^{i-1} f_{n-1}$, and with coefficients $(\gamma_i) = (\frac{1}{2}, \frac{5}{12}, \frac{3}{8}, \frac{251}{720}, \ldots)$. The method thus corrects the explicit Euler method by adding differences of previous function values.

Especially for higher orders, the accuracy of the approximation suffers from the fact that the interpolation polynomial is used outside the interval of interpolation. This is avoided if the, as yet, unknown value (t_{n+1}, f_{n+1}) is added to the interpolation points. Let $P^*(t)$ denote the corresponding interpolation polynomial, which is now used to replace the integrand $f(t, y(t))$. This yields the *implicit Adams method* of order $k+1$, which takes the form

$$y_{n+1} = y_n + hf_{n+1} + h \sum_{i=1}^{k} \gamma_i^* \nabla^i f_{n+1},$$

with $(\gamma_i^*) = (-\frac{1}{2}, -\frac{1}{12}, -\frac{1}{24}, -\frac{19}{720}, \ldots)$. The equation for y_{n+1} is solved approximately by one or at most two fixed point iterations, taking the result

from the explicit Adams method as the starting iterate (the *predictor*) and inserting its function value on the right-hand side of the implicit Adams method (the *corrector*).

In a variable-order, variable step-size implementation, the required starting values are built up by starting with the methods of increasing order $1, 2, 3, \ldots$ one after the other. Strategies for increasing or decreasing the order are based on monitoring the backward difference terms. Changing the step size is computationally more expensive, since it requires a recalculation of all method coefficients. It is facilitated by passing information from one step to the next by the *Nordsieck vector*, which collects the values of the interpolation polynomial and all its derivatives at t_n scaled by powers of the step size.

3.5 Linear multistep methods

Both explicit and implicit Adams methods (with constant step size) belong to the class of *linear multistep methods*

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = h \sum_{j=0}^{k} \beta_j f_{n+j}$$

with $f_i = f(t_i, y_i)$ and $\alpha_k \neq 0$. This class also includes important methods for stiff problems, in particular the backward differentiation formulas (BDF) to be described in the next section. The theoretical study of linear multistep methods was initiated by Germund Dahlquist in 1956. He showed that for such methods

$$\textit{consistency} + \textit{stability} \iff \textit{convergence},$$

which together with the contemporaneous Lax equivalence theorem for discretizations of partial differential equations forms a basic principle of numerical analysis. What this means here is described in more detail below.

In contrast to one-step methods, having high order does not by itself guarantee that a multistep method converges as $h \to 0$. In fact, choosing the method coefficients in such a way that the order is maximized for a given k leads to a method that produces wild oscillations, which increase in magnitude with decreasing step size. One requires in addition a stability condition, which can be phrased as saying that all solutions to the linear difference equation $\sum_{j=0}^{k} \alpha_j y_{n+j} = 0$ stay bounded as $n \to \infty$, or equivalently:

All roots of the polynomial $\sum_{j=0}^{k} \alpha_j \zeta^j$ are in the complex unit disk, and those on the unit circle are simple.

If this stability condition is satisfied and the method is of order p, then the error satisfies $y_n - y(t_n) = \mathcal{O}(h^p)$ on bounded time intervals, provided that the error in the starting values is $\mathcal{O}(h^p)$.

Dahlquist also proved *order barriers*: the order of a stable k-step method cannot exceed $k + 2$ if k is even, $k + 1$ if k is odd, and k if the method is explicit ($\beta_k = 0$).

3.6 General linear methods

Predictor-corrector Adams methods fall neither into the class of multistep methods, since they use the predictor as an internal stage, nor into the class of Runge–Kutta methods, since they use previous function values. Linear multistep methods and Runge–Kutta methods are extreme cases of a more general class of methods

$$u_{n+1} = S u_n + h\Phi(t_n, u_n, h)$$

where u_n is a vector (usually of dimension a multiple of the dimension of the differential equation) from which the solution approximation $y_n \approx y(t_n)$ can be obtained by a linear mapping, S is a square matrix, and Φ depends on function values of f. (For example, for predictor-corrector methods we would have $u_n = (y_n, y_n^{\text{pred}}, y_{n-1}, \ldots, y_{n-k+1})$ in this framework.)

More general methods like these have been studied since the mid-1960s with the objective of looking for the "greatest good as a mean between extremes" (in the words of Aristotle and John Butcher). They include a number of methods of potential interest for both nonstiff and stiff problems, such as two-step Runge–Kutta methods or general linear methods with inherent Runge–Kutta stability, but as of now do not appear to have found their way into applications via competitive software.

4 Stiff problems

We have seen in the introduction that for important classes of differential equations, called stiff equations, the implicit Euler method yields a drastic improvement over the explicit Euler method. Are there higher-order methods with similarly good properties?

4.1 BDF methods

The k-step implicit Adams methods, though naturally extending the implicit Euler method, perform disappointingly on stiff problems for $k > 1$. Multistep methods from another extension of the implicit Euler method, based on numerical differentiation rather than integration, turn out to be better for stiff problems. Suppose that k solution approximations $y_{n-k+1}, \ldots, y_n$ have already been computed, and consider the interpolation polynomial $u(t)$ passing through y_{n+1-j} at t_{n+1-j} for $j = 0, \ldots, k$, including the as yet unknown approximation y_{n+1}. We then require the *collocation* condition

$$u'(t) = f(t, u(t)) \quad \text{at} \quad t = t_{n+1},$$

or equivalently, in the case of a constant step size h,

$$\sum_{j=1}^{k} \frac{1}{j} \nabla^j y_{n+1} = h f_{n+1}.$$

This *backward differentiation formula* (BDF) is an implicit linear multistep method of order k, which is found to be unstable for $k > 6$. Methods for smaller k, however, up to $k \leq 5$, are currently the most widely used methods for stiff problems, which are implemented in numerous computer codes. The usefulness of these methods was first observed by Curtiss and Hirschfelder (1952), who also coined the notion of stiff differential equations. Bill Gear's BDF code DIFSUB from 1971 was the first widely used code for stiff problems. It brought BDF methods ("Gear's method") to the attention of practitioners in many fields.

4.2 A-stability and related notions

Which properties make BDF methods successful for stiff problems? In 1963, Dahlquist systematically studied the behavior of multistep methods on the scalar linear differential equation

$$y' = \lambda y \quad \text{with} \quad \lambda \in \mathbb{C},\ \operatorname{Re}\lambda \leq 0,$$

whose use as a test equation can be justified by linearization of the differential equation and diagonalization of the Jacobian matrix. The behavior of a numerical method on this deceivingly simple scalar linear differential equation gives much insight into its usefulness for more general stiff problems, as is shown by both numerical experience and theory.

Clearly, the exact solution $y(t) = e^{t\lambda} y_0$ remains bounded for $t \to +\infty$ when $\operatorname{Re}\lambda \leq 0$. Following Dahlquist, a method is called *A-stable* if for every $\lambda \in \mathbb{C}$ with $\operatorname{Re}\lambda \leq 0$, the numerical solution y_n stays bounded as $n \to \infty$ for every step size $h > 0$ and every choice of starting values. The implicit Euler method and the second-order BDF method are A-stable, but the BDF methods of higher order are not. *Dahlquist's second order barrier* states that the order of an A-stable linear multistep method cannot exceed 2. This fundamental, if negative theoretical result has led to much work aimed at circumventing the barrier by using other methods or weaker notions of stability.

The *stability region* S is the set of all complex $z = h\lambda$, such that every numerical solution of the method applied to $y' = \lambda y$ with step size h stays bounded. The stability regions of explicit and implicit k-step Adams methods with $k > 1$ are bounded, which leads to step size restrictions when λ has large absolute value. The BDF methods up to order 6 are $\mathrm{A}(\alpha)$-stable, that is, the stability region contains an unbounded sector $|\arg(-z)| \leq \alpha$ with $\alpha = 90°, 90°, 86°, 73°, 51°, 17°$ for $k = 1, \ldots, 6$, respectively. The higher-order BDF methods therefore perform well for differential equations where the Jacobian has large eigenvalues near the negative real half-axis, but behave poorly when there are large eigenvalues near the imaginary axis.

4.3 Implicit Runge–Kutta methods

It turns out that there is no order barrier for A-stable Runge–Kutta methods.

Explicit Runge–Kutta methods cannot be A-stable, because application of such a method to

the linear test equation yields $y_{n+1} = P(h\lambda)y_n$, where P is a polynomial of degree s, the number of stages. The stability region of such a method is necessarily bounded, since $|P(z)| \to \infty$ as $|z| \to \infty$.

On the other hand, an *implicit* Runge–Kutta method has

$$y_{n+1} = R(h\lambda)y_n$$

with a rational function $R(z)$, called the *stability function* of the method, which is an approximation to the exponential at the origin, $R(z) = e^z + \mathcal{O}(z^{p+1})$ as $z \to 0$. The method is A-stable if $|R(z)| \le 1$ for $\mathrm{Re}\, z \le 0$. The subtle interplay between order and stability is clarified by the theory of *order stars*, developed by Wanner, Hairer and Nørsett in 1978. In particular, this theory shows that among the Padé approximations $R_{k,j}(z)$ to the exponential (the rational approximations of numerator degree k and denominator degree j of highest possible order $p = j + k$), precisely those with $k \le j \le k + 2$ are A-stable. Optimal-order implicit Runge–Kutta methods having the diagonal Padé approximations $R_{s,s}$ as stability function are the collocation methods based on the Gauss quadrature nodes, while those having the sub-diagonal Padé approximations $R_{s-1,s}$ are the collocation methods based on the right-hand Radau quadrature nodes. We turn to these important implicit Runge–Kutta methods next.

4.4 Gauss and Radau methods

A *collocation method* based on the nodes $0 \le c_1 < \cdots < c_s \le 1$ determines a polynomial $u(t)$ of degree at most s such that $u(t_0) = y_0$ and the differential equation is satisfied at the s points $t_0 + c_i h$:

$$u'(t) = f(t, u(t)) \ \text{ at } \ t = t_0 + c_i h, \quad i = 1, \ldots, s.$$

The solution approximation at the end-point is then

$$y_1 = u(t_0 + h),$$

which is taken as the starting value for the next step. As has been shown by Ken Wright (in 1970), such a collocation method is equivalent to an implicit Runge–Kutta method, the order of which is equal to the order of the underlying interpolatory quadrature with nodes c_i. The highest order $p = 2s$ is thus obtained with Gauss nodes. Nevertheless, Gauss methods have found little use in stiff initial value problems (as opposed to boundary value problems, see Section 6). The reason for this is that the stability function here satisfies $|R(z)| \to 1$ as $z \to -\infty$, whereas $e^z \to 0$ as $z \to -\infty$.

The desired damping property at infinity is obtained for the sub-diagonal Padé approximation. This is the stability function for the collocation method at Radau points, which are the nodes of the quadrature of order $p = 2s - 1$ with $c_s = 1$. Let us collect the basic properties: the s-stage Radau method is an implicit Runge–Kutta method of order $p = 2s - 1$; it is A-stable and has $R(\infty) = 0$.

The Radau methods have some more remarkable features: they are nonlinearly stable with the so-called *algebraic stability* property; their last internal stage equals the starting value for the next step (this property is useful for very stiff and for differential-algebraic equations); and their internal stages all have order s.

The last property does indeed hold for every collocation method with s nodes. It is important because of the phenomenon of *order reduction*: in the application of an implicit Runge–Kutta method to stiff problems, the method may have only the stage order, or stage order +1, with stiffness-independent error constants, instead of the full classical order p that is obtained with nonstiff problems.

The implementation of Radau methods by Hairer and Wanner (from 1991) is known for its robustness in dealing with stiff problems and differential-algebraic equations of the type $My' = f(t, y)$ with a singular matrix M.

4.5 Linearly implicit methods

BDF and implicit Runge–Kutta methods are fully implicit, and the resulting systems of nonlinear equations need to be solved by variants of Newton's method. To reduce the computational cost while retaining favorable linear stability properties, linearly implicit methods have been proposed, such as the *linearly implicit Euler method*, in which only a single iteration of Newton's

method is done in each step:

$$(I - hJ_n)(y_{n+1} - y_n) = hf_n,$$

where $J_n \approx \partial_y f(t_n, y_n)$. Thus just one linear system of equations is solved in each time step. The method is identical to the implicit Euler method for linear problems and therefore inherits its A-stability. Higher-order linearly implicit methods can be obtained by Richardson extrapolation of the linearly implicit Euler method, or they are specially constructed *Rosenbrock methods*. Like explicit Runge-Kutta methods these methods determine the solution approximation as

$$y_1 = y_0 + h\sum_{i=1}^{s} b_i Y_i', \qquad Y_i = y_0 + h\sum_{j=1}^{i-1} a_{ij} Y_j',$$

but compute the derivative stages consecutively by solving s linear systems of equations (written here for an autonomous problem, $f(t, y) = f(y)$, and $J = \partial_y f(y_0)$):

$$(I - \gamma hJ)Y_i' = f(Y_i) + hJ\sum_{j=1}^{i-1} \gamma_{ij} Y_j'.$$

Such methods are easy to implement and have gained popularity in the numerical integration of spatial semi-discretizations of partial differential equations. For large problems, the dominating numerical cost is in the solution of the systems of linear equations, using either direct sparse solvers or iterative methods such as preconditioned Krylov subspace methods.

4.6 Exponential integrators

While it appears an obvious idea to use the exponential of the Jacobian in a numerical method, this has for a long time been considered impractical, and particularly so for large problems. This attitude changed, however, in the mid-1990s when it was realized that Krylov subspace methods for approximating a matrix exponential times a vector, $e^{\gamma hJ}v$, show superlinear convergence, whereas there is generally only linear convergence for solving linear systems $(I - \gamma hJ)x = v$. Unless a good preconditioner for the linear system is available, computing the action of the matrix exponential is therefore computationally less expensive than solving a corresponding linear system. This fact led to a revival of methods using the exponential or related functions like $\varphi(z) = (e^z - 1)/z$, such as the *exponential Euler method*

$$y_{n+1} = y_n + h\varphi(hJ_n)f_n.$$

The method is exact for linear $f(y) = Jy + c$. It differs from the linearly implicit Euler method in that the entire function $\varphi(z)$ replaces the rational function $1/(1 - z)$. Higher-order exponential methods of one-step and multistep type have also been constructed. Exponential integrators have proven useful for large-scale problems in physics and for nonlinear parabolic equations, as well as for highly oscillatory problems like those considered in Section 5.6.

4.7 Chebyshev methods

For moderately stiff problems one can avoid numerical linear algebra altogether by using *explicit* Runge-Kutta methods of low order (2 or 4) and high stage number, which are constructed to have a large stability domain covering a strip near the negative real semi-axis. The stability function of such methods is a high-degree polynomial related to Chebyshev polynomials. The stage number is chosen adaptively to include the product of the step size with the dominating eigenvalues of the Jacobian in the stability domain. With s stages, one can cover intervals on the negative real axis of a length proportional to s^2. The quadratic growth of the stability interval with the stage number makes these methods suitable for problems with large negative real eigenvalues of the Jacobian, such as spatial semidiscretizations of parabolic partial differential equations.

5 Structure-preserving methods

The methods discussed so far are designed for general differential equations, and a distinction was drawn only between nonstiff and stiff problems. There are, however, important classes of differential equations with a special, often geometric structure, whose preservation in the numerical discretization leads to substantially better methods, especially when integrating over

long times. The most prominent of these are Hamiltonian systems, which are all-important in physics. Their flow has the geometric property of being symplectic.

In respecting the phase space geometry under discretization and analyzing its effect on the long-time behavior of a numerical method, there is a shift of viewpoint from concentrating on the approximation of a single solution trajectory to considering the numerical method as a discrete dynamical system that approximates the flow of a differential equation.

While many differential equations have interesting structures to preserve under discretization — and much work has been done in devising and analyzing appropriate numerical methods for doing so — we will restrict our attention to Hamiltonian systems here. Their numerical treatment has been an active research area for the past two decades.

5.1 Symplectic methods

The time-t *flow* of a differential equation $y' = f(y)$ is the map φ_t that associates with an initial value y_0 at time 0 the solution value at time t: $\varphi_t(y_0) = y(t)$. Consider a Hamiltonian system

$$p' = -\nabla_q H(p,q), \quad q' = \nabla_p H(p,q),$$

or equivalently, for $y = (p,q)$,

$$y' = J^{-1}\nabla H(y) \quad \text{with} \quad J = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix}.$$

The flow φ_t of a Hamiltonian system is *symplectic*, that is, the derivative $D\varphi_t$ with respect to the initial value satisfies

$$D\varphi_t(y)^{\mathrm{T}}\, J\, D\varphi_t(y) = J$$

for all y and t for which $\varphi_t(y)$ exists. This is a quadratic relation formally similar to orthogonality, with J in place of the identity matrix I, but it is related to the preservation of areas rather than lengths in phase space.

A numerical one-step method $y_{n+1} = \Phi_h(y_n)$ is called symplectic if the numerical flow Φ_h is a symplectic map:

$$D\Phi_h(y)^{\mathrm{T}}\, J\, D\Phi_h(y) = J.$$

Such methods exist: the "symplectic Euler method" of Section 1.3 is indeed symplectic. Great interest in symplectic methods was spurred when, in 1988, Lasagni, Sanz-Serna and Suris independently characterized symplectic Runge–Kutta methods as those whose coefficients satisfy the condition

$$b_i a_{ij} + b_j a_{ji} - b_i b_j = 0.$$

Gauss methods (see Section 4.4) were already known to satisfy this condition, and were thus found to be symplectic. Soon after these discoveries it was realized that a numerical method is symplectic if and only if the modified differential equation of backward analysis (Section 2.5) is again Hamiltonian. This made it possible to prove rigorously the almost-conservation of energy over times that are exponentially long in the inverse step size, as well as further favorable long-time properties such as the almost-preservation of KAM (Kolmogorov–Arnold–Moser) tori of perturbed integrable systems over exponentially long times.

5.2 The Störmer–Verlet method

By the time symplecticity was entering the field of numerical analysis, scientists in molecular simulation had been doing symplectic computations for more than twenty years without knowing it: the standard integrator of molecular dynamics, the method used successfully ever since Luc Verlet introduced it to the field in 1967, is symplectic. For a Hamiltonian $H(p,q) = \frac{1}{2}p^{\mathrm{T}}M^{-1}p + V(q)$ with a symmetric positive definite mass matrix M, the method is explicit and given by the formulas

$$\begin{aligned} p_{n+1/2} &= p_n - \frac{h}{2}\nabla V(q_n) \\ q_{n+1} &= q_n + hM^{-1}p_{n+1/2} \\ p_{n+1} &= p_{n+1/2} - \frac{h}{2}\nabla V(q_{n+1}). \end{aligned}$$

Such a method was also formulated by the astronomer Störmer in 1907, and in fact can even be traced back to Newton's *Principia* from 1687, where it was used as a theoretical tool in the proof of the preservation of angular momentum in the two-body problem (Kepler's second law), which is indeed preserved by this method. Given

that there are already sufficiently many Newton methods in numerical analysis, it is fair to refer to the method as the *Störmer–Verlet method* (Verlet method and leapfrog method are also often-used names). As will be discussed in the next three subsections, the symplecticity of this method can be understood in various ways by relating the method to different classes of methods that have proven useful in a variety of applications.

5.3 Composition methods

Let us denote the one-step map $(p_n, q_n) \mapsto (p_{n+1}, q_{n+1})$ of the Störmer–Verlet method by Φ_h^{SV}, that of the symplectic Euler method of Section 1.3 by Φ_h^{SE}, and that of the adjoint symplectic Euler method by Φ_h^{SE*}, where instead of the argument (p_{n+1}, q_n) one uses (p_n, q_{n+1}). Then, the second-order Störmer–Verlet method can be interpreted as the *composition* of the first-order symplectic Euler methods with halved step size:

$$\Phi_h^{SV} = \Phi_{h/2}^{SE*} \circ \Phi_{h/2}^{SE}.$$

Since the composition of symplectic maps is symplectic, this shows the symplecticity of the Störmer–Verlet method. We further note that the method is *time-reversible* (or *symmetric*): $\Phi_{-h} \circ \Phi_h = \mathrm{id}$, or equivalently $\Phi_h = \Phi_h^*$ with the adjoint method $\Phi_h^* := \Phi_{-h}^{-1}$. This is known to be another favorable property for conservative systems.

Moreover, from first-order methods we have obtained a second-order method. More generally, starting from a low-order method Φ_h, methods of arbitrary order can be constructed by suitable compositions

$$\Phi_{c_s h} \circ \cdots \circ \Phi_{c_1 h} \quad \text{or} \quad \Phi_{b_s h}^* \circ \Phi_{a_s h} \circ \cdots \circ \Phi_{b_1 h}^* \circ \Phi_{a_1 h}.$$

Systematic approaches to high-order compositions were first given by Suzuki and Yoshida in 1990. Whenever the base method is symplectic, then so is the composed method. The coefficients can be chosen such that the resulting method is symmetric.

5.4 Splitting methods

Splitting the Hamiltonian $H(p,q) = T(p) + V(q)$ into its kinetic energy $T(p) = \frac{1}{2} p^{\mathrm{T}} M^{-1} p$ and potential energy $V(q)$, we have that the flows φ_t^T and φ_t^V of the systems with Hamiltonians T and V, respectively, are obtained by solving the trivial differential equations

$$\varphi_t^T : \left\{ \begin{array}{l} p' = 0 \\ q' = M^{-1} p \end{array} \right. \qquad \varphi_t^V : \left\{ \begin{array}{l} p' = -\nabla V(q) \\ q' = 0\,. \end{array} \right.$$

We then note that the Störmer–Verlet method can be interpreted as a composition of the exact flows of the split differential equations:

$$\Phi_h^{SV} = \varphi_{h/2}^T \circ \varphi_h^V \circ \varphi_{h/2}^T.$$

Since the flows $\varphi_{h/2}^T$ and φ_h^V are symplectic, so is their composition.

Splitting the vector field of a differential equation and composing the flows of the subsystems is a structure-preserving approach that yields methods of arbitrary order and is useful in the time integration of a variety of ordinary and partial differential equations, such as linear and nonlinear Schrödinger equations.

5.5 Variational integrators

For the Hamiltonian $H(p,q) = \frac{1}{2} p^{\mathrm{T}} M^{-1} p + V(q)$, the Hamilton equations of motion $\dot{p} = -\nabla V(q)$, $\dot{q} = M^{-1} p$ can be combined to give the second-order differential equation

$$M\ddot{q} = -\nabla V(q),$$

which can be interpreted as the Euler-Lagrange equations for minimizing the action integral

$$\int_{t_0}^{t_N} L(q(t), \dot{q}(t))\, dt \;\text{ with }\; L(q,\dot{q}) = \tfrac{1}{2} \dot{q}^{\mathrm{T}} M \dot{q} - V(q)$$

over all paths $q(t)$ with fixed end-points. In the Störmer-Verlet method, eliminating the momenta yields the second-order difference equations

$$M(q_{n+1} - 2q_n + q_{n-1}) = -h^2 \nabla V(q_n),$$

which are the discrete Euler-Lagrange equations for minimizing the discretized action integral

$$\sum_{n=0}^{N-1} \frac{h}{2} \Big(L\big(q_n, \tfrac{q_{n+1}-q_n}{h}\big) + L\big(q_{n+1}, \tfrac{q_{n+1}-q_n}{h}\big) \Big),$$

which results from a trapezoidal rule approximation to the action integral and piecewise linear approximation to $q(t)$. The Störmer–Verlet

method can thus be interpreted as resulting from the direct discretization of the Hamilton variational principle. Such an interpretation can in fact be given for every symplectic method. Conversely, symplectic methods can be *constructed* by minimizing a discrete action integral. In particular, approximating the action integral by a quadrature formula and the positions $q(t)$ by a piecewise polynomial leads to a symplectic partitioned Runge–Kutta method, which in general uses different Runge–Kutta formulas for positions and momenta. With Gauss quadrature one reinterprets in this way the Gauss methods of Section 4.4, and with higher-order Lobatto quadrature formulas one obtains higher-order relatives of the Störmer–Verlet method.

5.6 Oscillatory problems

Highly oscillatory solution behavior in Hamiltonian systems typically arises when the potential is a multiscale sum $V = V^{[\mathrm{slow}]} + V^{[\mathrm{fast}]}$, where the Hessian of $V^{[\mathrm{fast}]}$ has positive eigenvalues that are large compared with those of $V^{[\mathrm{slow}]}$. (Here we assume $M = I$ for simplicity.) With standard methods such as the Störmer–Verlet method, very small time steps would be required, for reasons of both accuracy and stability. Various numerical methods have been devised with the aim to overcome such a limitation. Here we just describe one such method, a *multiple time-stepping method* that reduces the computational work significantly when the slow force $f^{[\mathrm{slow}]} = -\nabla V^{[\mathrm{slow}]}$ is far more expensive to evaluate than the fast force $f^{[\mathrm{fast}]} = -\nabla V^{[\mathrm{fast}]}$. A basic principle is to rely on averages instead of pointwise force evaluations. In the *averaged-force method*, the force $f_n = -\nabla V(q_n)$ in the Störmer–Verlet method is replaced with an averaged force $\overline{f}_n$ as follows: we freeze the slow force at q_n and consider the auxiliary differential equation

$$\ddot{u} = f^{[\mathrm{slow}]}(q_n) + f^{[\mathrm{fast}]}(u)$$

with initial values $u(0) = q_n$, $\dot{u}(0) = 0$. We then define the averaged force as

$$\overline{f}_n = \int_{-1}^{1} (1 - |\theta|)\big(f^{[\mathrm{slow}]}(q_n) + f^{[\mathrm{fast}]}(u(\theta h))\big)\, d\theta,$$

which equals $\overline{f}_n = \frac{1}{h^2}\big(u(h) - 2u(0) + u(-h)\big)$. The value $u(h)$ is computed approximately with smaller time steps, noting $u(h) = u(-h)$.

The argument of $f^{[\mathrm{slow}]}$ might preferably be replaced with an averaged value $\overline{q}_n$, in order to mitigate the adverse effect of possible step size resonances that appear when the product of h with an eigenfrequency of the Hessian is close to an integral multiple of π.

If the fast potential is quadratic, $V^{[\mathrm{fast}]}(q) = \frac{1}{2} q^{\mathrm{T}} A q$, the auxiliary differential equation can be solved exactly in terms of trigonometric functions of the matrix $h^2 A$. The resulting method can then be viewed as an exponential integrator as considered in Section 4.6.

6 Boundary value problems

In a two-point boundary value problem, the differential equation is coupled with boundary conditions of the same dimension:

$$\begin{aligned} &y'(t) = f(t, y(t)), \quad a \le t \le b, \\ &r(y(a), y(b)) = 0. \end{aligned}$$

As an important class of examples, such problems arise as the Euler–Lagrange equations of variational problems, typically with separated boundary conditions $r_a(y(a)) = 0$, $r_b(y(b)) = 0$.

6.1 The sensitivity matrix

The problem of existence and uniqueness of a solution is more subtle than for initial value problems. For a linear boundary value problem

$$\begin{aligned} &y'(t) = C(t)y(t) + g(t), \quad a \le t \le b, \\ &Ay(a) + By(b) = q, \end{aligned}$$

a unique solution exists if and only if the *sensitivity matrix* $E = A + B\,U(b, a)$ is invertible, where $U(t, s)$ is the propagation matrix yielding $v(t) = U(t, s)v(s)$ for every solution of the linear differential equation $v'(t) = C(t)v(t)$.

A solution of a nonlinear boundary value problem is locally unique if the linearization along this solution has an invertible sensitivity matrix.

6.2 Shooting

Just as Newton's method replaces a nonlinear system of equations with a sequence of linear systems, the shooting method replaces a boundary

value problem with a sequence of initial value problems. The objective is to find an initial value x such that the solution of the differential equation with this initial value, denoted $y(t;x)$, satisfies the boundary conditions:

$$F(x) := r(x, y(b;x)) = 0.$$

Newton's method is now applied to this nonlinear system of equations: starting from an initial guess x^0, one iterates

$$x^{k+1} = x^k + \Delta x^k \quad \text{with} \quad DF(x^k)\Delta x^k = -F(x^k).$$

Here, the derivative matrix $DF(x^k)$ turns out to be the sensitivity matrix E^k of the linearization of the boundary value problem along $y(t;x^k)$. In the kth iteration, one solves numerically the initial value problem with initial value x^k together with its linearization

$$(Y^k)'(t) = \partial_y f(t, y(t;x^k))\, Y^k(t), \quad Y^k(a) = I.$$

6.3 Multiple shooting

The conceptual elegance of the shooting method — that it reduces everything to the solution of initial value problems over the whole interval — can easily turn into its computational obstruction. Newton's method may be very sensitive to the choice of the initial value x^0. The norms of the matrices E^{-1} and $U(b,a)$, which determine the effect of perturbations in the boundary value problem and the initial value problem, respectively, are unrelated and may differ widely.

The problem can be avoided by subdividing the interval $a = t_0 < t_1 < \cdots < t_N = b$, shooting on every subinterval, and requiring continuity of the solution at the nodes t_n. With $y(t;t_n,x_n)$ denoting the solution of the differential equation that starts at t_n with initial value x_n, this approach leads to a larger nonlinear system with the continuity conditions

$$F_n(x_{n-1}, x_n) = y(t_n; t_{n-1}, x_{n-1}) - x_n = 0$$

for $n = 1, \ldots, N$ together with the boundary conditions

$$F_0(x_0, x_N) := r(x_0, x_N) = 0.$$

Newton's method is now applied to this system of equations. In each iteration, one solves initial value problems on the subintervals together with their linearization, and then a linear system with a large sparse matrix is solved for the increments in $(x_0, \ldots, x_N)$.

6.4 Collocation

In the collocation approach to the boundary value problem, one determines an approximation $u(t)$ that is a continuous, piecewise polynomial of degree at most s, and that satisfies the boundary conditions and the differential equation at a finite number of collocation points $t_{n,i} = t_{n-1} + c_i(t_n - t_{n-1})$ (for $n = 1, \ldots, N$ and $i = 1, \ldots, s$):

$$\begin{aligned} &u'(t) = f(t, u(t)) \quad \text{at } t = t_{n,i} \\ &r(u(a), u(b)) = 0. \end{aligned}$$

The method can be interpreted, and implemented, as a multiple shooting method in which a single step with a collocation method for initial value problems, as considered in Section 4.4, is made to approximate the solution in each subinterval. The most common choice, as first implemented by Ascher, Christiansen and Russell in 1979, is collocation at Gauss nodes, which has good stability properties in the forward and backward directions. The order of approximation at the grid points t_n is $p = 2s$. Moreover, if the boundary value problem results from a variational problem, then Gauss collocation can be interpreted as a direct discretization of the variational problem (see Section 5.5).

7 Summary

The numerical solution of ordinary differential equations is an area driven both by applications and theory, with efficient computer codes alongside beautiful theorems, both relying on insight and knowledge of the researchers that are active in this field. It is an area that interacts with neighboring fields in computational mathematics (numerical linear algebra, the numerical solution of partial differential equations and optimization), with the theory of differential equations and dynamical systems, and time and again with the application areas in science and engineering where numerical methods for differential equations are used.

Further Reading

1. Ascher, U.M., Mattheij, R.M.M. and Russell, R.D. 1995 *Numerical solution of boundary value problems for ordinary differential equations.* SIAM, Philadelphia.
2. Ascher, U.M. and Petzold, L.R. 1998 *Computer methods for ordinary differential equations and differential-algebraic equations*, SIAM, Philadelphia.
3. Butcher, J.C. 2008 *Numerical methods for ordinary differential equations*, second revised ed., Wiley, Chichester.
4. Crouzeix, M. and Mignot, A.L. 1989 *Analyse numérique des équations différentielles*, 2e éd. révisée et augmentée. Masson, Paris.
5. Deuflhard, P. and Bornemann, F. 2002 *Scientific computing with ordinary differential equations*, Springer, New York.
6. Gear, C.W. 1971 *Numerical initial value problems in ordinary differential equations*, Prentice-Hall, Englewood Cliffs, NJ.
7. Hairer, E., Nørsett, S.P. and Wanner, G. 1993 *Solving ordinary differential equations. I: Nonstiff problems.* 2nd revised ed., Springer, Berlin.
8. Hairer, E. and Wanner, G. 1996 *Solving ordinary differential equations. II: Stiff and differential-algebraic problems.* 2nd revised ed., Springer, Berlin.
9. Hairer, E., Lubich, C. and Wanner, G. 2006 *Geometric numerical integration. Structure-preserving algorithms for ordinary differential equations.* 2nd revised ed., Springer, Berlin.
10. Henrici, P. 1962 *Discrete variable methods in ordinary differential equations*, Wiley, New York.
11. Iserles, A. 2009 *A first course in the numerical analysis of differential equations.* Second edition, Cambridge Univ. Press, Cambridge.
12. Leimkuhler, B. and Reich, S. 2004 *Simulating Hamiltonian dynamics*, Cambridge Univ. Press, Cambridge.

Biographies of contributors

Ernst Hairer, born in 1949, Dr. phil. at Univ. Innsbruck in 1972, since 1985 Professor of Numerical Mathematics at Université de Genève.

Christian Lubich, born in 1959, Dr. rer. nat. at Univ. Innsbruck in 1983, since 1994 Professor of Numerical Mathematics at Universität Tübingen.