9 numerical methods for ODEs

goal: solve, for given $y_0 \in \mathbb{R}$ and function f the initial value problem

$$y'(t) = f(t, y(t)), \qquad y(0) = y_0 \tag{9.1}$$

We will be interested in the solution y in the interval $[0, T]$. The numerical methods will seek approximations $y_i \approx y(t_i)$ in the points

$$0 = t_0 < t_1 < \cdots < t_N = T.$$

We denote by $h_i := t_{i+1} - t_i$ the *step lengths* and by $h := \max_i h_i$ the maximal step length.

9.1 Euler's method

The simplest method is the explicit Euler method. Starting from the known value $y_0 = y(t_0)$ we seek an approximation y_1. By Taylor approximation we observe

$$y(t_1) = y(t_0) + h_0 y'(t_0) + O(h_0^2) \overset{(9.1)}{=} y_0 + h_0 f(t_0, y_0) + O(h^2).$$

Hence, we are led to the approximation

$$y_1 := y_0 + h_0 f(t_0, y_0).$$

Since we assume that y_1 is a good approximation to $y(t_1)$, we may repeat the Taylor argument to obtain $y_2 := y_1 + h_1 f(t_1, y_1)$. This leads to the *explicit Euler method*: define the approximations y_i to the exact values $y(t_i)$ successively by

$$y_{i+1} := y_i + h_i f(t_i, y_i), \qquad i = 0, \ldots, N-1. \tag{9.2}$$

It is not obvious that the final approximation $y(T) - y_N = y(t_N) - y_N$ really is a good one as errors over many steps may accumulate. Indeed, while the Taylor approximation is valid in the first step (y_0 is exact), already in the second step we replace $y(t_1)$ with the approximation y_1 in the Taylor approximation and we have to expect that this additional error is potentially amplified by the recursion (9.2). Nevertheless, under reasonable assumptions the explicit Euler method converges. For the proof, we need the notion of the *consistency error* τ: Let $t \mapsto y(t)$ be the exact solution. The consistency error $\tau_{eE}(t, h)$ of the explicit Euler method at t is defined as

$$\tau_{eE}(t, h) := y(t + h) - [y(t) + h f(t, y(t))] \tag{9.3}$$

We recognize that τ_{eE} measures the error of *one* step of the method when started with the exact value $y(t)$. We note that Taylor's formula gives

$$|\tau_{eE}(t, h)| \leq \frac{1}{2} h^2 \|y''\|_{\infty, [0, T+h]} \tag{9.4}$$

Theorem 9.1 (convergence of explicit Euler) *Let $f \in C^1(\mathbb{R}^2)$ with bounded derivatives, i.e., there is $L > 0$ such that $|\nabla f(t, x)| \leq L$ for all $(t, x) \in \mathbb{R}^2$. Then there exists $C > 0$ such that for the approximation y_i obtained by (9.2)*

$$\max_{i=0,\ldots,N-1} |y(t_i) - y_i| \leq C e^{LT} h.$$

Proof: We define the errors

$$e_i := y(t_i) - y_i$$

and note that $e_0 = 0$. For simplicity of notation, we assume a uniform step length $h_i = h$ (exercise: check that the proof also works for variable step size). We seek a recursion for the errors. To that end, we write

$$\begin{aligned} y_{i+1} &= y_i + hf(t_i, y_i), \\ y(t_{i+1}) &= y(t_i) + hf(t_i, y(t_i)) + \tau(t_i, h). \end{aligned}$$

We obtain

$$e_{i+1} = e_i + h \underbrace{[f(t_i, y(t_i)) - f(t_i, y_i)]}_{(y(t_i)-y_i)\partial_y f(t_i,\xi)} + \tau(t_i, h),$$

where used the intermediate value theorem for some ξ between y_i and $y(t_i)$. We obtain

$$|e_{i+1}| \le |e_i| + h \underbrace{|y(t_i) - y_i|}_{=|e_i|} \underbrace{|\partial_y f(t_i, \xi)|}_{\le L} + |\tau(t_i, h)| \le (1 + hL)|e_i| + |\tau(t_i, h)| \le e^{hL}|e_i| + |\tau(t_i, h)|$$

We now repeatedly use this estimate:

$$\begin{aligned} |e_i| &\le e^{hL}|e_{i-1}| + |\tau(t_{i-1}, h)| \le e^{hL}\left[e^{hL}|e_{i-2}| + |\tau(\tau_{i-2}, h)|\right] + |\tau(t_{i-1}, h)| \\ &= e^{2hL}|e_{i-2}| + e^{hL}|\tau(\tau_{i-2}, h)| + |\tau(t_{i-1}, h)| \\ &\le \cdots \le e^{ihL}|e_0| + \sum_{j=0}^{i-1} e^{jhL}|\tau(t_{i-j-1}, h)| \end{aligned}$$

We note that $e_0 = 0$. For the sum, we use that $jh = t_j \le T$ and (9.4) to infer

$$|e_i| \le \sum_{j=0}^{i-1} e^{jhL}|\tau(t_{i-j-1}, h)| \le \frac{1}{2}e^{TL}\sum_{j=0}^{i-1} h^2\|y''\|_{\infty,[0,T+h]} = \frac{1}{2}e^{TL}\|y''\|_{\infty,[0,T+h]} ih^2 \le \frac{1}{2}e^{TL}\|y''\|_{\infty,[0,T+h]} Th,$$

which is the desired first order convergence. □

Example 9.2 *hier ein Beispiel fuer expliziten Euler* ∎

The explicit Euler method was motivated by Taylor expansion around t_i to obtain the value y_{i+1} at t_{i+1}. Alternatively, one could perform Taylor expansion around t_{i+1}. That is,

$$y(t_i) = y(t_{i+1}) + (t_i - t_{i+1}) \underbrace{y'(t_{i+1})}_{=f(t_{i+1}, y(t_{i+1}))} + O(h_i^2),$$

so that, by replacing $y(t_i)$ with y_i and $y(t_{i+1})$ with y_{i+1} and dropping the $O(h_i^2)$, we get the *implicit Euler method*

$$y_{i+1} = y_i + h_i f(t_{i+1}, y_{i+1}). \tag{9.5}$$

The method is *implicit* since y_{i+1} is obtained from y_i by solving a (in general) nonlinear equation.

Exercise 9.3 *Formulate the Newton method to compute y_{i+1} given y_i.* ∎

Analogous to the consistency error for the explicit Euler method (9.4), we have for the consistency error for the implicit Euler method the equation

$$\tau_{iE}(t,h) = y(t+h) - h\left[y(t) + hf(t+h, y(t+h))\right], \tag{9.6}$$

where $t \mapsto y(t)$ is again the exact solution of $y'(t) = f(t, y(t))$. Taylor expansion again gives $\tau_{iE}(t,h) = O(h^2)$ for exact solutions $y \in C^2$. One can show that the implicit Euler method satisfies

$$\max_i |y(t_i) - y_i| \le Ch.$$

Both explicit and implicit Euler method are *first order* methods. finis 15.DS

9.2 Runge-Kutta methods

The explicit and implicit Euler methods are one-step methods[1] of order 1. A generalization of these two one-step methods are methods of the form

$$y_{i+1} = y_i + h_i \Phi(t_i, h_i, y_i, y_{i+1}) \tag{9.7}$$

for some given *increment function* Φ ($\Phi(t_i, h_i, y_i, y_{i+1}) = f(t_i, y_i)$ is the explicit Euler, $\Phi(t_i, h_i, y_i, y_{i+1}) = f(t_i + h_i, y_{i+1})$ is the implicit Euler method.) We are interested in deriving increment functions Φ such that the method is of order p, i.e., that (given sufficient smoothness of f) one has

$$\max_i |y(t_i) - y_i| \le Ch^p.$$

9.2.1 explicit Runge-Kutta methods

There are many different ways to introduce one-step methods of order higher than 1. Here, we motivate the structure of so-called *Runge-Kutta*-methods by extrapolation techniques, which we encountered already in Section 1.4. The extrapolation technique relies on comparing two different approximations: a) one step of the explicit Euler with step length h and b) two steps of the explicit Euler method with step length $h/2$, viz

$$\begin{aligned} y_1^{(1)} &= y_0 + hf(t_0, y_0), \\ y_1^{(2)} &= y_{1/2} + \frac{h}{2} f(t_{1/2}, y_{1/2}), \qquad y_{1/2} = y_0 + \frac{h}{2} f(t_0, y_0), \quad t_{1/2} = t + \frac{h}{2} \end{aligned}$$

From the above developments, each of these approximations has error $O(h^2)$, i.e.,

$$\begin{aligned} y(t_1) - y_1^{(1)} &= \tau^{(1)}(t_0, h) = O(h^2), \\ y(t_1) - y_1^{(2)} &= \tau^{(2)}(t_0, h) = O(h^2). \end{aligned}$$

[1] that is, the value y_{i+1} is determined by y_i and not, for example, by y_i and y_{i-1}

We define the actual step of the method as a linear combination of $y_1^{(1)}$ and $y_1^{(2)}$ in such a way that the resulting consistency error is $y(t_1) - y_1 = O(h^3)$. To that end, we carefully employ Taylor's theorem:

$$\begin{aligned}
y(t_1) &= \underbrace{y(t_0)}_{y=y_0} + hy'(t_0) + \frac{1}{2}h^2 y''(t_0) + O(h^3),\\
y_1^{(1)} &= y_0 + hy'(t_0)\\
y_1^{(2)} &= y_{1/2} + \frac{h}{2}f(t_{1/2}, y_{1/2}) = y_0 + \frac{h}{2}f(t_0, y_0) + \frac{h}{2}f(t_0 + \frac{h}{2}, y_0 + \frac{h}{2}f(t_0, y_0))\\
&= y_0 + \frac{h}{2}f(t_0, y_0) + \frac{h}{2}\left[f(t_0, y_0) + \partial_t f(t_0, y_0)\frac{h}{2} + \partial_y f(t_0, y_0)\frac{h}{2}f(t_0, y_0) + O(h^2)\right]\\
&= y_0 + hf(t_0, y_0) + \frac{h^2}{4}\left[\partial_t f(t_0, y_0) + \partial_y f(t_0, y_0)f(t_0, y_0)\right] + O(h^3)
\end{aligned}$$

Next, we use that $t \mapsto y(t)$ is a solution of the differential equation, i.e., $y'(t) = f(t, y(t))$. Hence, by differentation with respect to t we get with the chain rule

$$y''(t) = \partial_t f(t, y(t)) + \partial_y f(t, y(t))y'(t) = \partial_t f(t, y(t)) + \partial_y f(t, y(t))f(t, y(t)).$$

In particular, for $t = t_0$ and $y(t_0) = y_0$, we obtain

$$y_1^{(2)} = y_0 + hy'(t_0) + \frac{h^2}{4}y''(t_0) + O(h^3)$$

Therefore, for parameters α, β we can compute

$$\begin{aligned}
&y(t_1) - [\alpha y_1^{(1)} + \beta y_1^{(2)}]\\
&\quad = y_0 + hy'(t_0) + \frac{h^2}{2}y''(t_0) + O(h^3) - \alpha[y_0 + hy'(t_0, y_0)] - \beta[y_0 + hy'(t_0) + \frac{h^2}{4}y''(t_0) + O(h^3)]\\
&\quad = y_0(1 - \alpha - \beta) + y'(t_0)h[1 - \alpha - \beta] + y''(t_0)h^2\left[\frac{1}{2} - \beta\frac{1}{4}\right] + O(h^3)
\end{aligned}$$

The conditions on α and β are therefore

$$\begin{aligned}
1 - \alpha - \beta &= 0\\
2 - \beta &= 0
\end{aligned}$$

with solution $\beta = 2$ and $\alpha = -1$. The method is therefore $y_1 = 2y_1^{(2)} - y_1^{(1)}$ or, more explicitly,

$$k_1 := f(t_0, y_0), \tag{9.8a}$$

$$k_2 := f(t_0 + \frac{h}{2}, y_0 + \frac{h}{2}k_1), \tag{9.8b}$$

$$y_1 := 2\left(y_0 + \frac{h}{2}k_1 + \frac{h}{2}k_2\right) - y_0 + hk_1) = y_0 + hk_2 \tag{9.8c}$$

This method is of order 2, i.e., $\max_i |y(t_i) - y_i| \leq Ch^2$ (for sufficiently smooth exact solution y). In principle, even higher order methods can be constructed by this extrapolation idea. However, a more general class of methods emerges from the structure in (9.8), the *explicit Runge-Kutta methods*:

Definition 9.4 (explicit Runge-Kutta method) *For a given number of* stages $s \in \mathbb{N}$*, parameters* $c_i \in [0,1]$*,* $b_i \in \mathbb{R}$ *and* $a_{ij} \in \mathbb{R}$ *define*

$$\begin{aligned} k_1 &= f(t_0, y_0), \\ k_2 &= f(t_0 + c_2 h, y_0 + a_{21} k_1), \\ &\vdots \\ k_s &= f(t_0 + c_s h, y_0 + \sum_{j=1}^{s-1} a_{sj} k_j) \end{aligned}$$

and the update $y_1 = y_0 + h \sum_{i=1}^{s} b_i k_i$*. The method is compactly described by the* Butcher tableau*:*

$$\begin{array}{c|cccc} 0 & 0 & & & \\ c_2 & a_{21} & 0 & & \\ c_3 & a_{31} & a_{32} & 0 & \\ \vdots & \vdots & & & \\ c_s & a_{s1} & \cdots & a_{s,s-1} & 0 \\ \hline & b_1 & b_2 & \cdots & b_s \end{array}$$

Exercise 9.5 *Write down the Butcher tableau for the explicit Euler method and the method of order* 2 *derived above.* ∎

Example 9.6 (RK4) *A popular explicit Runge-Kutta method is RK4 with 4 stages and order 4 given by* $y_1 = y_0 + h\Phi(t, y_1, h)$*, where*

$$\begin{aligned} \Phi(t,y,h) &:= \frac{1}{6}\left[k_1 + 2k_2 + 2k_3 + k_4\right], \\ k_1 &:= f(t,y), \\ k_2 &:= f\left(t + \frac{h}{2}, y + \frac{1}{2}hk_1\right), \\ k_3 &:= f\left(t + \frac{h}{2}, y + \frac{1}{2}hk_2\right), \\ k_4 &:= f\left(t + h, y + hk_3\right). \end{aligned}$$

The corresponding Butcher tableau is

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{2}{6} & \frac{2}{6} & \frac{1}{6} \end{array}$$

∎

Exercise 9.7 *Program RK4 and apply it to the right-hand side* $f_1(t,y) = y$ *and* $y_0 = 1$*. Plot the error at* $T = 1$ *versus* h *for* $h = 2^{-n}$*,* $n = 1, \ldots, 10$*.* ∎

Exercise 9.8 *The solution of* $y'(t) = f(t)$*,* $y(t_0) = 0$ *is given by* $y(t) = \int_{t_0}^{t} f(\tau)\, d\tau$*. Hence, for right-hand sides of the form* $f(t,y) = f(t)$*, a Runge-Kutta method results in a quadrature formula. Which quadrature formula is obtained for RK4?*

9.2.2 implicit Runge-Kutta methods

The form of the explicit Runge-Kutta methods in Def. 9.9 suggests a generalization, the so-called *implicit Runge-Kutta methods*:

Definition 9.9 (implicit Runge-Kutta method) *For a given number of* stages $s \in \mathbb{N}$*, parameters* $c_i \in [0,1]$*,* $b_i \in \mathbb{R}$ *and* $a_{ij} \in \mathbb{R}$ *define the stages* k_i*,* $i = 1, \dots, s$ *as the solution of the following (nonlinear) system of equations:*

$$k_i = f(t_0 + c_i h, y_0 + h \sum_{j=1}^{s} a_{ij} k_j), \qquad i = 1, \dots, s.$$

One step of the implicit Runge-Kutta is then given by $y_1 = y_0 + h \sum_{i=1}^{s} b_i k_i$*. The method is compactly described by the* Butcher tableau*:*

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & & \ddots & \vdots \\ c_s & a_{s1} & \cdots & a_{s,s-1} & a_s \\ \hline & b_1 & b_2 & \cdots & b_s \end{array}$$

Exercise 9.10 *Show that the implicit Euler method is a* 1*-stage implicit Runge-Kutta method by writing down the corresponding Butcher tableau.* ∎

Example 9.11 (θ-scheme) *For* $\theta \in [0,1]$ *the scheme with Butcher tableau*

$$\begin{array}{c|c} \theta & \theta \\ \hline & 1 \end{array}$$

is called the θ*-scheme. It is given by*

$$k_1 = f(t_0 + \theta h, y_0 + \theta h k_1), \qquad y_1 = y_0 + h k_1$$

The auxiliary variable k_1 *can be eliminated using* $y_0 + \theta h k_1 = \theta(y_0 + h k_1) + (1-\theta) y_0 = \theta y_1 + (1-\theta) y_1$ *so that it is*

$$y_1 = y_0 + h f(t_0 + \theta h, \theta y_1 + (1-\theta) y_0)$$

We recognize the explicit Euler method for $\theta = 0$ *and the implicit Euler method for* $\theta = 1$*. For* $\theta = 1/2$*, the method is called "midpoint rule" (the simplest Gauss rule). We mention that the* θ*-scheme is of order* 1 *for* $\theta \neq 1/2$ *and it is of order* 2 *for* $\theta = 1/2$*.* ∎

9.2.3 why implicit methods?

Explicit Runge-Kutta methods are usually preferred over implicit methods as they do not require solving a (nonlinear) equation in each step. These nonlinear equations are typically solved by Newton's method (or some variant), and the user has to provide the derivative $\partial_y f$. Nevertheless, implicit Runge-Kutta methods (or variants) are the method of choice for certain classes of problems such as *stiff ODEs*. A typical situation where implicit methods shine are problems that describe problems with vastly differing time-scales. In these situations, explicit methods would require very small step sizes for reasonable results whereas implicit methods achieve good accuracy with much larger step sizes.

Example 9.12 Consider the solution of initial value problem

$$\mathbf{y}' = \mathbf{A}\mathbf{y}, \qquad \mathbf{y}(0) = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}, \qquad \mathbf{A} = \begin{pmatrix} -21 & 19 & -20 \\ 19 & -21 & 20 \\ 40 & -40 & -40 \end{pmatrix}.$$

The eigenvalues of $\mathbf{A}$ are given by $\lambda_1 = -2$, $\lambda_2 = -40(1+\mathbf{i})$, $\lambda_3 = -40(1-\mathbf{i})$. The solution is:

$$\begin{aligned} \mathbf{y}_1(t) &= \frac{1}{2}e^{-2t} + \frac{1}{2}e^{-40t}\left(\cos 40t + \sin 40t\right), \\ \mathbf{y}_2(t) &= \frac{1}{2}e^{-2t} - \frac{1}{2}e^{-40t}\left(\cos 40t + \sin 40t\right), \\ \mathbf{y}_3(t) &= -e^{-40t}\left(\cos 40t - \sin 40t\right). \end{aligned}$$

All 3 solution components vary rather rapidly in the regime $0 \le t \le 0.1$ so that a step length restriction $h \ll 1$ seems plausible. For $t > 0.1$, however, the components $\mathbf{y}_1$ and $\mathbf{y}_2$ vary rather slowly (the rapidly oscillatory contribution has been damped out due to the factor e^{-40t}) and $\mathbf{y}_3$ is close to zero. From an approximation point of view, therefore, one would hope that larger time steps are possible. However, Fig. 9.1 shows that, for example, for $h = 0.05$, the explicit Euler method yields completely unacceptable results. Indeed, one can show that the explicit Euler method can only be expected to yield acceptable results if the step length h satisfies the *stability condition*

$$|1 + hz| \le 1 \qquad z \in \{\lambda_1, \lambda_2, \lambda_3\}$$

i.e., it has to satisfy $h \le \frac{1}{40} = 0.025$. In contrast, the implicit Euler method, which is also visible in Fig. 9.1 performs much better since it does not have to satisfy such a stability condition. ∎

slide 37
slide 38
slide 39
slide 40

finis 16.DS

9.2.4 the concept of A-stability (CSE)

The above examples have shown that for certain examples of ODEs explicit methods "fail" in the sense that convergence only sets in for very small step sizes. In contrast, (certain) implicit methods perform well for much larger step sizes. Mathematically, the notion of *A-stability* captures the difference in behavior.

the stability function R

Consider the scalar model equation

$$y' = \lambda y, \qquad y(0) = y_0 \tag{9.9}$$

where $\lambda \in \mathbb{C}$. The exact solution is $y(t) = e^{\lambda t} y_0$. One step of length h of an RK-method has the form

$$y_1 = R(\lambda h) y_0, \tag{9.10}$$

where $R(z)$ is a polynomial for an explicit method and a rational function for an implicit method:

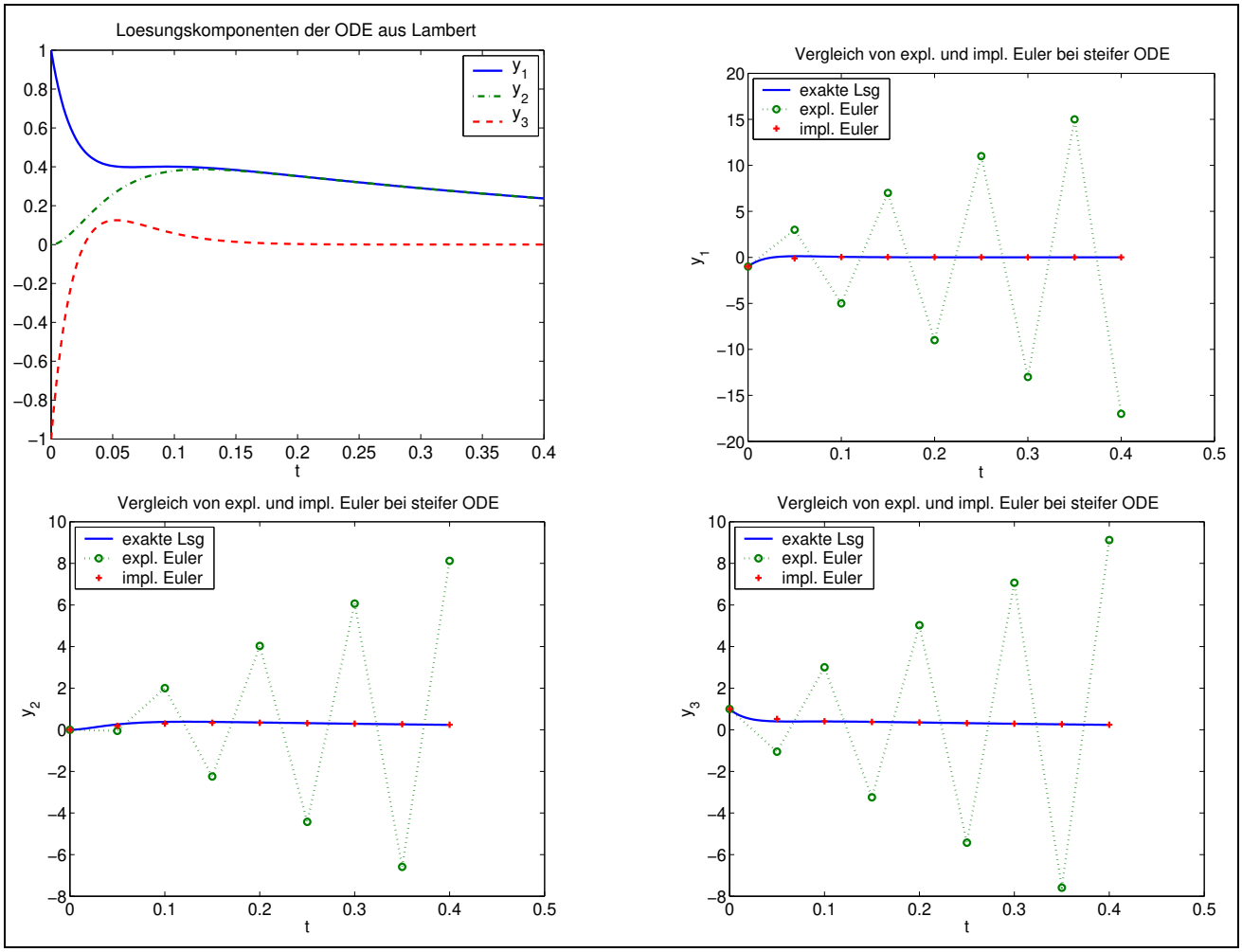

Figure 9.1: Comparison of explicit and implicit Euler method for the stiff problem of Example 9.12: exact solution (top left) and numerical approximation.

Exercise 9.13 *). show:*

1. *explicit Euler method:* $R(z) = 1 + z$
2. *implicit Euler method:* $R(z) = 1/(1 - z)$
3. *θ-scheme with $\theta = 1/2$:* $R(z) = \frac{1+z/2}{1-z/2}$
4. *RK4:* $R(z) = 1 + z + \frac{z^2}{2!} + \frac{z^3}{3!} + \frac{z^4}{4!}$

Without proof, we mention that for any RK-method that leads to a convergent method the stability function R has the form $R(z) = 1 + z + O(z^2)$ as $z \to 0$.

Definition 9.14 *An RK-method is said to be* A-stable, *if*

$$|R(z)| \leq 1 \qquad \forall z \text{ with } \operatorname{Re} z \leq 0.$$

Exercise 9.15 *If R is a polynomial, then the corresponding RK-method cannot be A-stable. Since the function R associated with an explicit RK-method is a polyomial, explicit RK-methods cannot be A-stable.*
In particular, the explicit Euler method is not *A-stable whereas the implicit Euler method is. The θ-scheme with $\theta = 1/2$ is A-stable. See also Fig. 9.2.* ∎

Consider the case $\operatorname{Re}\lambda \leq 0$. Then the exact solution $y(t) = e^{\lambda t} y_0$ stays bounded for $t \to \infty$. (For $\operatorname{Re}\lambda < 0$ the solution even decays to 0.) From (9.10) we see that the discrete solutions y_i are given by

$$y_i = (R(\lambda h))^i y_0, \qquad i = 0, 1, \ldots,$$

Hence, for the discrete approximations to be bounded (as $i \to \infty$), we have to require $|R(\lambda h)| \leq 1$. Since $\operatorname{Re}\lambda \leq 0$ and $h > 0$, we see that this is ensured for A-stable methods irrespective of $h > 0$.
Put differently: A-stability of an RK-method ensures that the property that the solution $y(t) = e^{\lambda t} y_0$ is bounded (for $\operatorname{Re}\lambda \leq 0$) is reproduced by the numerical method for *any* $h > 0$.

Example 9.16 *A-stability ensures boundedness of the discrete solution for* $\operatorname{Re}\lambda \leq 0$ *and any h. For* $\operatorname{Re}\lambda < 0$ *and* sufficiently small *h the condition $|R(\lambda h)| \leq 1$ is ensured. We illustrate this for the explicit Euler method: For the explicit Euler method, one has $R(\lambda h) = 1 + \lambda h$. Hence, for $\lambda < 0$ one has*

$$|R(\lambda h)| \leq 1 \quad \Longleftrightarrow \quad |1 + \lambda h| \leq 1 \quad \Longleftrightarrow \quad h \leq \frac{2}{|\lambda|}.$$

If $\lambda \ll -1$, then this condition on h is very restrictive. ∎

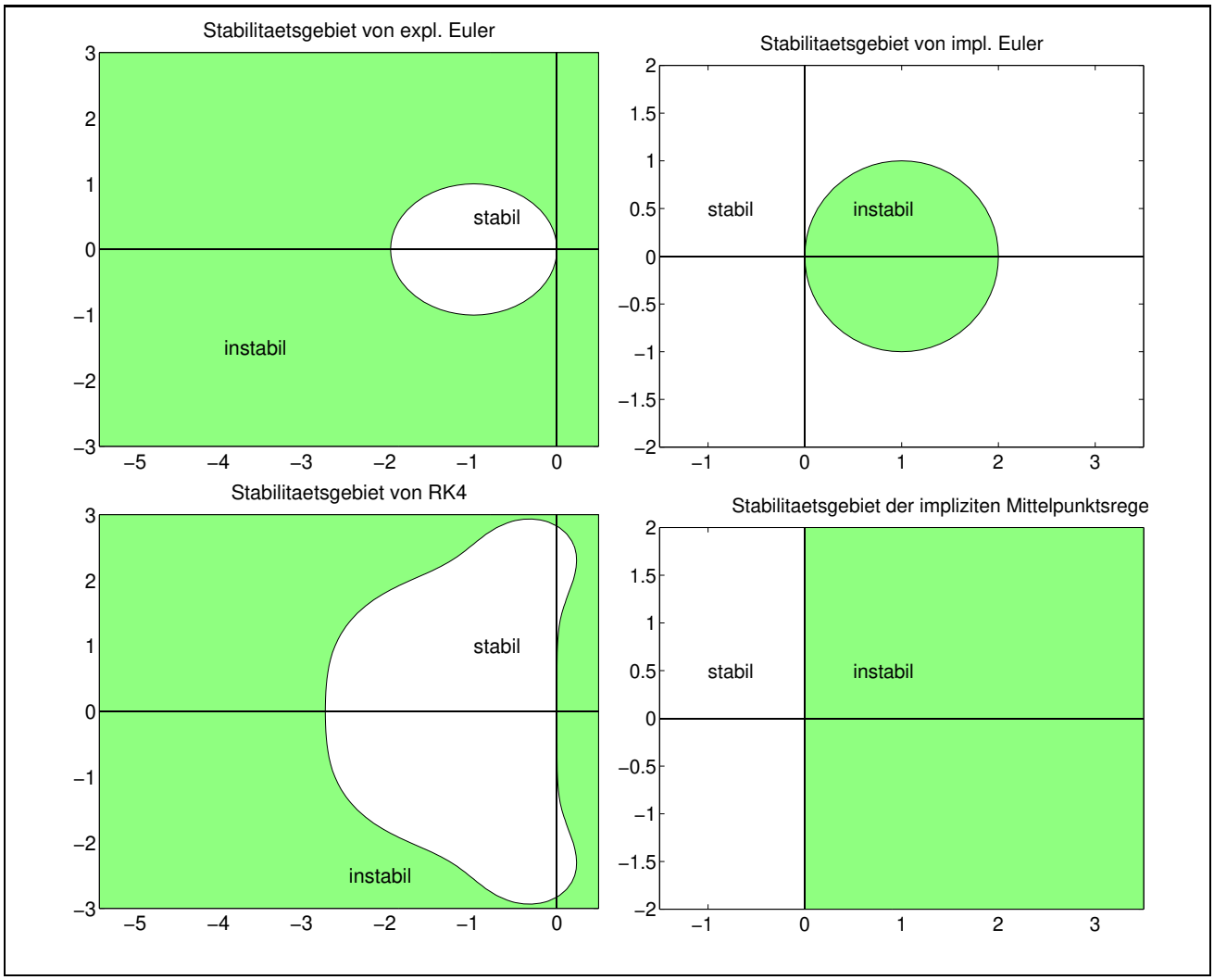

Figure 9.2: stability regions $\{z \in \mathbb{C} \,|\, |R(z)| \leq 1\}$ for explicit Euler, implicit Euler, RK4, and implicit midpoint rule.

stability of RK-methods

To get insight into the performance of an RK-method, we consider the *model*

$$\mathbf{y}' = \mathbf{A}\mathbf{y}, \qquad \mathbf{y}(t_0) = \mathbf{y}_0 \tag{9.11}$$

where $\mathbf{A} \in \mathbb{C}^{n\times n}$ is a (constant) matrix. Such a model may be viewed as a linearization of a more complex ODE and one hopes that studying the RK-method applied to the linearization captures the key properties. We assume additionally that $\mathbf{A}$ can be diagonalized:

$$\mathbf{A} = \mathbf{T}^{-1}\mathbf{D}\mathbf{T}$$

so that after the change of variables $\widehat{\mathbf{y}} = \mathbf{T}\mathbf{y}$ the ODE (9.11) is equivalent to

$$\widehat{\mathbf{y}}' = \mathbf{D}\widehat{\mathbf{y}}, \qquad \widehat{\mathbf{y}}(t_0) = \widehat{\mathbf{y}}_0 = \mathbf{T}\mathbf{y}_0. \tag{9.12}$$

It can be checked that for RK-methods, one step of the RK-method could be computed in two different ways: either one applies the RK-method directly to (9.11) or one applies it to the transformed equation (9.12) and transforms back. This is depicted in Fig. 9.3. The RK-method applied to (9.12) is simpler to understand since it reduces to the application of the RK-method to *scalar* problems of the form (9.9) where $\lambda \in \mathbb{C}$ is a diagonal entry of $\mathbf{D}$, i.e., an eigenvalue of $\mathbf{A}$. One step of length h of an RK-method applied to (9.9) has the form $\widehat{\mathbf{y}}_k^{i+1} = R(\lambda_k h)\widehat{\mathbf{y}}_k^i$ where $R(z)$ is the *stability function* and we use the subscript k to indicate the component of the vector $\widehat{\mathbf{y}}$ while the superscripts $i+1$ and i indicate the association with time steps t_{i+1} and t_i. Hence,

$$\widehat{\mathbf{y}}_k^{i+1} = R(\lambda_k h)\widehat{\mathbf{y}}_k^i = \cdots = (R(\lambda_k h))^{i+1}\widehat{\mathbf{y}}_k^0, \qquad k = 1, \ldots, n,$$

If $\operatorname{Re}\lambda_k \ll -1$ then one is well-advised to ensure $|R(\lambda_k h)| \leq 1$ to reproduce this boundedness of the exact solution component. For A-stable methods, this is ensured no matter what h is. The following considerations argue why this is a sensible condition. For simplicity of notation, we assume that the eigenvalues λ_k are real (so as to be able to formulate conditions on λ_k instead of on $\operatorname{Re}\lambda_k$):

- One may expect a good approximation for those components $\widehat{\mathbf{y}}_k$ for which $|\lambda_k h|$ is small. For these components, one has $R(\lambda_k h) \approx 1$ (note that $R(z) = 1 + O(z)$ in the examples of Exercise 9.13). Suppose that for some $\lambda_k \ll -1$ and an $h > 0$ one has $|R(\lambda_k h)| > 1$. If the RK-method is applied to the diagonalized form (9.12), then the error in these component $\widehat{\mathbf{y}}_k$ is very large while the other components may be reasonably well approximated. One may be tempted to argue that this is acceptable since that solution component is practically zero (and thus known!) so that there is no need to approximate it numerically anyway. However, if the RK-method is applied to the original form (9.11), then the presence of a single eigenvalue λ_k with $|R(\lambda_k h)| > 1$ will ruin all components since the transformation $\mathbf{y} = \mathbf{T}^{-1}\widehat{\mathbf{y}}$ mixes all components of $\widehat{\mathbf{y}}$ so that one expects that all components of $\mathbf{y}^1$ have contributions of $\widehat{\mathbf{y}}_k^1$ (unless $\mathbf{T}^{-1}$ has special structure). In other words: **When applying the RK-method to (9.11), the time step $h > 0$ is dictated by the maximum of $\{-\lambda_j \,|\, j = 1, \ldots, n\}$.** However, is very unsatisfactory that solution components $\widehat{\mathbf{y}}_k$ with large $-\lambda_k$ dictate the step size although they hardly contribute to the exact solution.

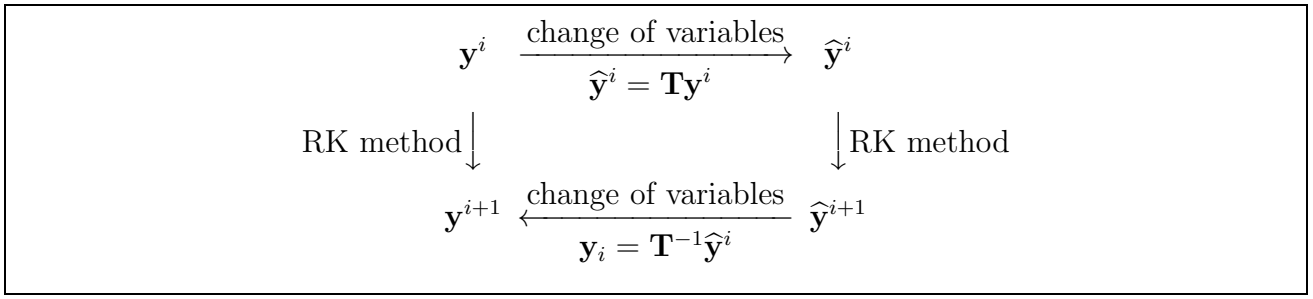

Figure 9.3: RK-method applied to $\mathbf{y}' = \mathbf{A}\mathbf{y}$ and to $\widehat{\mathbf{y}} = \mathbf{D}\widehat{\mathbf{y}}$ after change of variables. The superscripts i and $i+1$ refer to the time steps t_i and t_{i+1}.

- Related to the above point is a consideration of error propagation. In each step of the RK, some consistency error is made. Consider again the RK-method in the variable $\widehat{\mathbf{y}}$ and an initial error $\widehat{\mathbf{e}}^0$. For $\lambda_k \ll -1$ and $|R(\lambda_k h)| > 1$ the error in the kth component is actually damped by the exact evolution (by a factor $e^{\lambda_k h}$) whereas it is amplified by a factor $|R(\lambda_k h)|$ by the RK-method. Thus, initial errors are amplified by a factor $|R(\lambda_k h)|^i$ in the ith step. Fixing $t_i = ih$, we rewrite this amplification factor as

  $$|R(\lambda_k h)|^i = |R(\lambda_k h)|^{t_i/h} = \left(|R(\lambda_k h)|^{1/h}\right)^{t_i}.$$

  For fixed t_i and $|R(\lambda_k h)| > 1$, we have that $|R(\lambda_k h)|^{1/h}$ is very large. In conclusion, we have to expect that the method will dramatically amplify initial errors for small h. Again, this error amplification in one component will affect all components if one applies the RK-method to the original form (9.11).

What about $\lambda_k > 0$? In a nutshell: large (positive) λ_k also impose step size restrictions, i.e., they also require that $\lambda_k h$ be small. However, this step size restriction is acceptable since it is necessary to approximate the solution. To be more specific, we note that the error amplification discussed above arises for $|R(\lambda_k h)| > 1$ and this situation occurs also for $\lambda_k > 0$ (e.g., for the explicit Euler method is $R(\lambda_k h) = 1 + \lambda_k h$). However, the exact solution grows as well so that the amplification of the relative error is not dramatic. To fix ideas, consider the explicit Euler method. Then with initial error $\widehat{\mathbf{e}}_k$ the relative error at t_i is

$$\frac{|R(\lambda_k h)|^i |\widehat{\mathbf{e}}_k^0|}{|\widehat{\mathbf{y}}_k^0|} e^{\lambda_k t_i} = \frac{|\widehat{\mathbf{e}}_k^0|}{|\widehat{\mathbf{y}}_k^0|} \frac{(1+\lambda_k h)^{t_i/h}}{e^{\lambda_k t_i}} \leq \frac{|\widehat{\mathbf{e}}_k^0|}{|\widehat{\mathbf{y}}_k^0|} = \text{rel. error at } t_0,$$

where we used $(1+x) \leq e^x$ so that $(1+\lambda_k h)^{t_i \lambda_k/(\lambda_k h)} \leq e^{t_i \lambda_k}$.

9.3 Multistep methods (CSE)

goal: high order methods with less function evaluations than RK-methods
observation: one-step methods such as RK-methods do *not* make use of the "history" available
$\rightarrow$ reuse previous function evaluations for efficiency increase
setting: we employ uniform step size h (multistep methods with variable step size are rather complicated!)
notation: $t_i = t_0 + ih$, $f_i := f(t_i, y_i)$

9.3.1 Adams-Bashforth methods

Let $r \in \mathbb{N}_0$. Given $(t_{i-k}, y_{i-k}), \ldots, (t_i, y_i)$ we wish to find y_{i+1}. To motivate the method, we note that the exact solution satisfies

$$y(t_{i+1}) = y(t_i) + \int_{t_i}^{t_{i+1}} y'(t)\,dt = y(t_i) + \int_{t_i}^{t_{i+1}} f(t, y(t))\,dt. \tag{9.13}$$

Noting that we have approximations to $f(t_{i-j}, y(t_{i-j})) \approx f_{i-j}$, $j = 0, \ldots, k$, in our hand, we approximate the integrand by its interpolating polynomial of degree k, i.e.,

$$f(t, y(t)) \approx \sum_{j=0}^{k} f_{i-j}\ell_{i-j}(t), \qquad \ell_{i-j}(t) := \prod_{\substack{s=0 \\ s\neq j}}^{k} \frac{t - t_{i-s}}{t_{j-s} - t_{i-s}}$$

In view of $t_i = t_0 + hi$ and the change of variables $t = t_i + h\tau$ we compute

$$\int_{t_i}^{t_{i+1}} \sum_{j=0}^{k} f_{i-j}\ell_{i-j}(t)\,dt = \sum_{j=0}^{k} f_{i-j} \int_{t_i}^{t_{i+1}} \prod_{\substack{s=0 \\ s\neq j}}^{k} \frac{t - t_{i-s}}{t_{j-s} - t_{i-s}} = \sum_{j=0}^{k} f_{i-j} h \underbrace{\int_{\tau=0}^{\tau=1} \prod_{\substack{s=0 \\ s\neq j}}^{k} \frac{\tau + s}{j - i}\,d\tau}_{\beta_j}$$

We note that β_j is independent of h and f and can be precomputed (see Example 9.17). We have thus arrived at the *Adams-Bashforth method*

$$y_{i+1} = y_i + h\sum_{j=0}^{k} \beta_j f_{i-j} \tag{9.14}$$

Example 9.17 Adams-Bashforth methods *(explicit):*

$$\begin{aligned}
k = 0 \quad & y_{i+1} = y_i + hf_i \qquad \textit{(explicit Euler)} \\
k = 1 \quad & y_{i+1} = y_i + h\frac{1}{2}(3f_i - f_{i-1}) \\
k = 2 \quad & y_{i+1} = y_i + h\frac{1}{12}(23f_i - 16f_{i-1} + 5f_{i-2}) \\
k = 3 \quad & y_{i+1} = y_i + h\frac{1}{24}(55f_i - 59f_{i-1} + 37f_{i-2} - 9f_{i-3})
\end{aligned}$$

Theorem 9.18 *The $k+1$-step Adams-Bashforth method is a method of order $k+1$, i.e., if f is sufficiently smooth, then the consistency error (i.e., the error in one step) is $O(h^{k+2})$. Specifically, provided the initial errors $|y(t_i) - y_i|$, $i = 0, \ldots, k$, are $O(h^{k+1})$ then*

$$\max_i |y(t_i) - y_i| \leq Ch^{k+1},$$

where the constant C depends on f and the terminal time $T = t_N$.

We note that the method requires only one evaluation of f per step. It is therefore more economical than the RK-methods (if the number of function evaluations is taken as the cost measure).

9.3.2 Adams-Moulton methods

The Adams-Bashforth method above is an explicit method. The Adams method exist also in implicit variant. This method is derived in a way very similar to (9.13) by simply changing the domain of integration. The exact solution satisfies

$$y(t_{i+1}) = y(t_i) + \int_{t_i}^{t_{i+1}} f(t, y(t))\, dt. \tag{9.15}$$

Again, replacing the integrand with the polynomial interpolating in (t_{i+1-j}, y_{i+1-j}), $j = 0, \ldots, k$, we arrive at the method

$$y_{i+1} = y_i + \sum_{j=0}^{k} f_{i+1-j} \int_{t_i}^{t_{i+1}} \ell_{i+1-j}(t)\, dt, \qquad \ell_{i+1-j}(t) = \prod_{\substack{s=0 \\ s\neq j}}^{k} \frac{t - t_{i+1-s}}{t_{i+1-j} - t_{i+1-s}}\, dt$$

Again, the terms $\int_{t_i}^{t_{i+1}} \ell_{i+1-j}(t)\, dt$ are of the form $h\beta_j$ for some coefficients β_j depending only on k (Exercise!). The first few Adams-Moulton methods are given in the following example:

Example 9.19 Adams-Moulton method *(implicit):*

$$\begin{aligned}
k = 0 \quad & y_{i+1} = y_i + hf_{i+1} && \textit{(implicit Euler)} \\
k = 1 \quad & y_{i+1} = y_i + h\frac{1}{2}(f_{i+1} + f_i) && \textit{(trapezoidal rule)} \\
k = 2 \quad & y_{i+1} = y_i + h\frac{1}{12}(5f_{i+1} + 8f_i - f_{i-1}) \\
k = 3 \quad & y_{i+1} = y_i + h\frac{1}{24}(9f_{i+1} + 19f_i - 5f_{i-1} + f_{i-2})
\end{aligned}$$

Theorem 9.20 *The k-step Adams-Moulton method[2] is a method of order $k+1$, i.e., if f is sufficiently smooth, then the consistency error (i.e., the error in one step) is $O(h^{k+2})$. Specifically, provided the initial errors $|y(t_i) - y_i|$, $i = 0, \ldots, k$, are $O(h^{k+1})$ then*

$$\max_i |y(t_i) - y_i| \leq Ch^{k+1},$$

where the constant C depends on f and the terminal time $T = t_N$.

[2]for $k = 0$, the Adams-Moulton method is the implicit Euler method, i.e., also a 1-step method

The Adams-Moulton methods are *implicit* methods since the involve $f_{i+1} = f(t_{i+1}, y_{i+1})$ to compute the y_{i+1}.

Exercise 9.21 *The Adams-Moulton method have fixed point form for the unknown y_{i+1} and could be solved with the fixed point iteration (cf. (6.1)). Assume that f is smooth. Show, using Theorem 6.5, that the fixed point iteration converges for sufficiently small h.*

In practice of Adams-Moulton methods, the fixed point iteration discussed in Exercise 9.21 is refined:

Remark 9.22 (predictor-corrector method) *A reasonable starting guess for y_{i+1} is y_i. (By Taylor, one expects $y_{i+1} - y_i = O(h)$.) A better approximation could be obtained by an explicit Adams-Bashforth method. Then, one performs m steps of the fixed-point iteration (9.16). That is,*

$$
\begin{aligned}
&y_{i+1}^{(0)} := \textit{approximation obtained from an Adams-Bashforth method} && (P)\\
&\textit{for } n = 0, \dots, m-1 \textit{ do}\{\\
&\quad f_{i+1}^{(n)} := f(t_{i+1}, y_{i+1}^{(n)}) && (E)\\
&\quad y_{i+1}^{(n+1)} := y_i + h\beta_0 f_{i+1}^{(n)} + h\sum_{j=1}^{k} f(t_{i+1-j} y_{i+1-j}), && (C)\\
&\}
\end{aligned}
$$

In this context, the first step is called "prediction" (denoted P) and the m steps of evaluating f are called "evaluate" (denoted E) and "correction" (denoted C). Correspondingly method is abbreviated $P(EC)^m$. ∎

9.3.3 BDF methods

The most important class of linear multistep methods are the BDF methods ("backward differentiation methods"). The BDF methods are suitable for stiff problems.
The BDF-k methods is obtained by interpolating (t_{i+1-j}, y_{i+1-j}), $j = 0, \dots, k$ by a polynomial of degree k and *collocating* the differential equation in the point t_{i+1}:

$$
\text{interpolating polynomial:} \qquad \pi_k(t) := \sum_{j=0}^{k} y_{i+1-j}\ell_{i+1-j}(t), \qquad \ell_{i+1-j}(t) = \prod_{\substack{s=0\\ s\neq j}}^{k} \frac{t - t_{i+1-s}}{t_{i+1-j} - t_{i+1-s}},
$$

$$
\text{collocate in } t_{i+1}: \quad \pi_k'(t_{i+1}) \stackrel{!}{=} f(t_{i+1}, \pi_k(t_{i+1})) = f(t_{i+1}, y_{i+1}).
$$

Example 9.23 *We consider the simplest case, $k = 1$:*

$$
\begin{aligned}
&\textit{interpolating polynomial:} && \pi_k(t) := y_{i+1}\frac{t - t_i}{h} - y_i\frac{t - t_{i+1}}{h}\\
&\textit{collocate in } t_{i+1}: && \pi_k'(t_{i+1}) = \frac{y_{i+1} - y_i}{h} \stackrel{!}{=} f(t_{i+1}, \pi_k(t_{i+1})) = f(t_{i+1}, y_{i+1}).
\end{aligned}
$$

This is the implicit Euler method. ∎

More generally, we have:

Example 9.24

$$
\begin{aligned}
&k=1 \qquad y_{i+1}-y_i = hf_{i+1} \qquad \textit{(implict Euler)}\\
&k=2 \qquad y_{i+1}-\frac{4}{3}y_i+\frac{1}{3}y_{i-1} = h\frac{2}{3}f_{i+1} \qquad \textit{BDF2}\\
&k=3 \qquad y_{i+1}-\frac{18}{11}y_i+\frac{9}{11}y_{i-1}-\frac{2}{11}y_{i-2} = h\frac{6}{11}f_{i+1} \qquad \textit{BDF3.}
\end{aligned}
$$

■

Theorem 9.25 *The BDFk-method is a method of order k, i.e., if f is sufficiently smooth then the consistency error (i.e., the error in one step of the method) is $O(h^{k+1})$. Specifically, provided the initial errors $|y(t_i)-y_i|$, $i=0,\ldots,k-1$, are $O(h^k)$ then*

$$\max_i |y(t_i)-y_i| \le Ch^k$$

Remark 9.26 *Since the BDF methods are typically employed for stiff problems, the implicit equations are* not *solved by a simple fixed point iteration but by Newton's method or a Newton-like method.* ■

9.3.4 Remarks on multistep methods

The Adams methods and the BDF-formulas are special cases of so-called *linear multistep methods*, which are update formulas of the form

$$\sum_{j=0}^{k} \alpha_j y_{i+1-j} = h\sum_{j=0}^{k} \beta_j f_{i+1-j} \tag{9.16}$$

for coefficients α_j, β_j. These methods are explicit if $\beta_j = 0$, otherwise they are implicit.

Starting value:

A multistep method determines y_{i+1} from several previous values y_{i+1-j}, $j=1,\ldots,k$. Since at the beginning only y_0 is given, one needs to compute the initial values $y_1,\ldots,y_k$ by some other method. For example, a Runge-Kutta method is employed. These values need to be computed to accuracy $O(h^p)$, where p is the order of the multistep method ($p=k+1$ for Adams-Bashforth and Adams-Moulton, $p=k$ for BDFk).

Bibliography

[1] Timothy A. Davis. *direct methods for sparse linear systems.* SIAM, 2006.