

TMA4215:
Lecture notes on numerical solution of ordinary differential
equations.

Anne Kværnø

November 21, 2016

Contents

1	Eulers method.	3
2	Some background on ODEs.	6
3	Numerical solution of ODEs.	8
3.1	Some examples of one-step methods.	9
3.2	Some examples of linear multistep methods.	10
4	Runge-Kutta methods	11
4.1	Order conditions for Runge-Kutta methods.	13
4.2	Error control and stepsize selection.	18
4.3	Dense output	22
4.4	Collocation methods.	22
5	Stiff equations	24
5.1	Linear stability.	27
5.2	Nonlinear stability	30
5.3	Order reduction	30

6	Linear multistep methods	34
6.1	Consistency and order.	35
6.2	Linear difference equations	36
6.3	Zero-stability and convergence	38
6.4	Adams-Bashforth-Moulton methods	39
6.5	Predictor-corrector methods	40

1 Eulers method.

Let us start this introduction to the numerical solution of *ordinary differential equations* (ODEs) by something familiar. Given a scalar (one equation only) ODE

$$y' = f(t, y), \quad t_0 \leq t \leq t_{end}, \quad y(t_0) = y_0, \quad (1)$$

in which the function f , the integration interval $[t_0, t_{end}]$ and the initial value y_0 is assumed to be given. The *solution* of this initial value problem (IVP) is a function $y(t)$ on the interval $[t_0, t_{end}]$.

Example 1.1. *The ODE/IVP*

$$y' = -2ty, \quad 0 \leq t \leq 1, \quad y(0) = 1.0$$

has as solution the function

$$y(t) = e^{-t^2}.$$

But in many practical situations, it is not possible to express the solution $y(t)$ in closed form, even if a solution exist. In these cases, a numerical algorithm can give an *approximation* to the exact solution. Let us start with Eulers method, which should be known from some calculus classes. Divide the interval $[t_0, t_{end}]$ into N_{step} equal subintervals, each of size $h = (t_{end} - t_0)/N_{step}$, and let $t_n = t_0 + nh$. Euler's method can be derived by several means. One possibility is to use the first few terms of the Taylor expansion of the exact solution, which is given by

$$y(t_0 + h) = y(t_0) + hy'(t_0) + \frac{1}{2}h^2y''(t_0) + \dots + \frac{1}{p!}h^py^{(p)}(t_0) + \frac{1}{(p+1)!}h^{p+1}y^{(p+1)}(\xi), \quad (2)$$

where ξ is somewhere between t_0 and t_{end} . The integer $p \geq 1$ is a number of our own choice, but we have to require y to be sufficiently differentiable, in this case that $y^{(p+1)}$ exist and is continuous. If h is small, we may assume that the solution will be completely dominated by the first two terms, thus

$$y(t_0 + h) \approx y(t_0) + hy'(t_0) = y_0 + hf(t_0, y_0),$$

and we call this approximate solution y_1 . Starting from the point $t_1 = t_0 + h$ and y_1 we can repeat the process. We have now developed Euler's method, given by

$$y_{n+1} = y_n + hf(t_n, y_n), \quad n = 0, 1, \dots, N_{step} - 1,$$

resulting in approximations $y_n \approx y(t_n)$.

Example 1.2. *Eulers method with $h = 0.1$ applied to the ODE of Example 1.1 gives*

t_n	y_n
0.0	1.0000
0.1	1.0000
0.2	0.9800
0.3	0.9408
0.4	0.8844
0.5	0.8136
0.6	0.7322
0.7	0.6444
0.8	0.5542
0.9	0.4655
1.0	0.3817

In this case we know the exact solution, $y(1.0) = e^{-1.0^2} = 0.3679$ and the error at the endpoint is $e_{10} = y(1.0) - y_{10} = -1.38 \cdot 10^{-2}$. If we repeat this experiment (write a MATLAB program to do so) with different stepsizes, and measure the error at the end of the interval, we get

h	$e_{N_{step}} = y(1.0) - y_{N_{step}}$
0.1	$-1.38 \cdot 10^{-2}$
0.05	$-6.50 \cdot 10^{-3}$
0.025	$-3.16 \cdot 10^{-3}$
0.0125	$-1.56 \cdot 10^{-3}$

From this example, it might look like the error at the endpoint $e_{N_{step}} \sim h$, where $h = (t_{end} - t_0)/N_{step}$. But is this true for all problems, and if yes, can we prove it? To do so, we need to see what kind of errors we have and how they behave. This is illustrated in Figure 1. For each step an error is made, and these errors are then propagated til the next steps and accumulate at the endpoint.

Definition 1.1. The local truncation error d_{n+1} is the error done in one step when starting at the exact solution $y(t_n)$. The global error is the difference between the exact and the numerical solution at point t_n , thus $e_n = y(t_n) - y_n$.

The local truncation error of Euler's method is

$$d_{n+1} = y(t_n + h) - y(t_n) - hf(t_n, y(t_n)) = \frac{1}{2}h^2y''(\xi),$$

where $\xi \in (t_n, t_{n+1})$. This is given from the Taylor-expansion of $y(t_n + h)$ around t_n with $p = 1$. To see how the global error propagates from one step to the next, the trick is: We have

$$\begin{aligned} y(t_n + h) &= y(t_n) + hf(t_n, y(t_n)) + d_{n+1}, \\ y_{n+1} &= y_n + hf(t_n, y_n). \end{aligned}$$

Take the difference of these two, and get

$$e_{n+1} = e_n + h(f(t_n, y(t_n)) - f(t_n, y_n)) + d_{n+1} = (1 + hf_y(t_n, v))e_n + d_{n+1}, \quad (3)$$

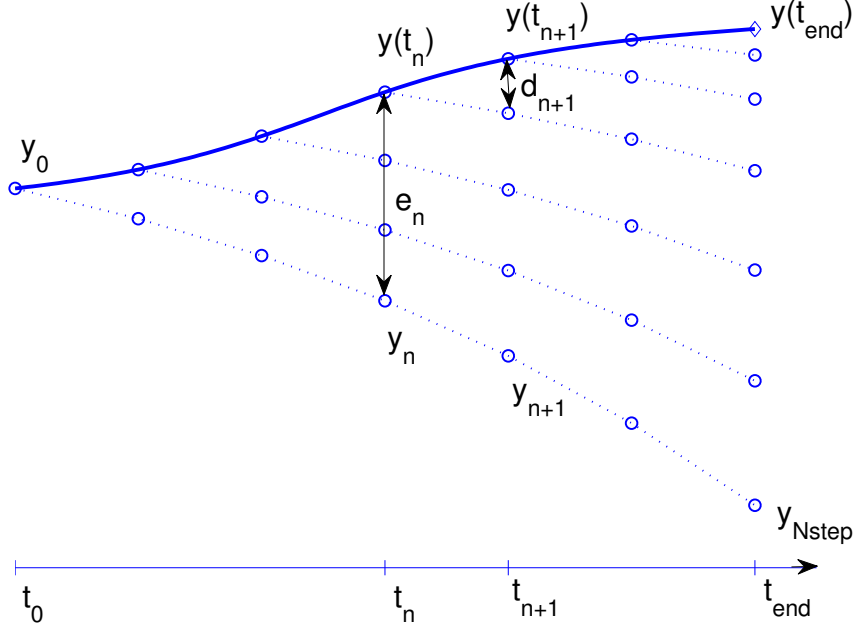


Figure 1: Lady Windermere's Fan

where v is somewhere between $y(t_n)$ and y_n . We have here used the mean value theorem (Theorem 1.8 in Burden and Faires) for $f(t_n, y(t_n)) - f(t_n, y_n)$. This is about as far as we get with exact calculations, since ξ in d_{n+1} as well as v in f_y are unknown, and will also change from one step to the next. So we will look for an *upper bound* of the global error. We will first assume upper bounds for our unknown, that is, we assume there exist positive constants D and L so that

$$\frac{1}{2} |y''| \leq D \text{ for all } t \in (t_0, t_{end}) \quad \text{and} \quad |f_y| \leq L \text{ for all } t \in [t_0, t_{end}] \text{ and for all } y.$$

Taken the absolute value of both sides of (3) and using the triangle inequality gives

$$|e_{n+1}| \leq (1 + hL) |e_n| + Dh^2.$$

Since $e_0 = 0$ (there is no error at the initial point) we can use this formula recursively to get an upper bound for the error at the endpoint:

$$\begin{aligned} |e_1| &\leq Dh^2, \\ |e_2| &\leq (1 + hL)Dh^2 + Dh^2 \\ &\vdots \\ |e_{N_{step}}| &\leq \sum_{i=0}^{N_{step}-1} (1 + hL)^i Dh^2 = \frac{(1 + hL)^{N_{step}} - 1}{1 + hL - 1} Dh^2. \end{aligned}$$

Using the fact that $1+hL \leq e^{hL}$ (why?) and $h \cdot N_{step} = t_{end} - t_0$ we finally reach the conclusion

$$|e_{N_{step}}| \leq \frac{(e^{hL})^{N_{step}} - 1}{L} Dh = \frac{e^{L(t_{end}-t_0)} - 1}{L} D \cdot h = C \cdot h.$$

The constant $C = (e^{hL} - 1) D/L$ depends only on the problem, and we have proved convergence

$$|y(t_{end}) - y_{N_{step}}| \rightarrow 0 \text{ when } h \rightarrow 0 \text{ (or } N_{step} \rightarrow \infty \text{)}.$$

2 Some background on ODEs.

In this section some useful notation on ordinary differential equations will be presented. We will also give existence and uniqueness results, but without proofs.

A system of m first order ordinary differential equation is given by

$$y' = f(t, y) \tag{4}$$

or, written out, as

$$\begin{aligned} y'_1 &= f_1(t, y_1, \dots, y_m), \\ y'_2 &= f_2(t, y_1, \dots, y_m), \\ &\vdots \\ y'_m &= f_m(t, y_1, \dots, y_m). \end{aligned}$$

This is an *initial value problem* (IVP) if the solution is given at some point t_0 , thus

$$y_1(t_0) = y_{1,0}, y_2(t_0) = y_{2,0}, \dots, y_m(t_0) = y_{m,0}.$$

Example 2.1. *The following equation is an example of the Lotka-Volterra equation:*

$$\begin{aligned} y'_1 &= y_1 - y_1 y_2, \\ y'_2 &= y_1 y_2 - 2y_2. \end{aligned}$$

An ODE is called *autonomous* if f is not a function of t , but only of y . The Lotka-Volterra equation is an example of an autonomous ODE. A nonautonomous system can be made autonomous by a simple trick, just add the equation

$$y'_{m+1} = 1, \quad y_{m+1}(t_0) = t_0,$$

and replace t with y_{m+1} . Also higher order ODE/IVPs

$$u^{(m)} = f(t, u, u', \dots, u^{(m-1)}), \quad u(t_0) = u_0, u'(t_0) = u'_0, \dots, u^{(m-1)}(t_0) = u_0^{(m-1)},$$

where $u^{(m)} = d^m u / dt^m$, can be written as a system of first order equations, again by a simple trick: Let

$$y_1 = u, \quad y_2 = u', \quad \dots \quad y_m = u^{(m-1)},$$

and we get the system

$$\begin{array}{ll} y_1' = y_2, & y_1(t_0) = u_0, \\ y_2' = y_3, & y_2(t_0) = u_0', \\ \vdots & \vdots \\ y_{m-1}' = y_m, & y_{m-1}(t_0) = u_0^{(m-2)}, \\ y_m' = f(t, y_1, y_2, \dots, y_m), & y_m(t_0) = u_0^{(m-1)}. \end{array}$$

Example 2.2. *Van der Pol's equation is given by*

$$u'' + \mu(u^2 - 1)u' + u = 0.$$

Using $y_1 = u$ and $y_2 = u'$ this equation can be rewritten as

$$\begin{array}{l} y_1' = y_2, \\ y_2' = \mu(1 - y_1^2)y_2 - y_1. \end{array}$$

This problem was first introduced by Van der Pol in 1926 in the study of an electronic oscillator.

In the rest of this course, we will assume that the following existence and uniqueness results holds:

Definition 2.1. *A function $f : \mathbb{R} \times \mathbb{R}^m \rightarrow \mathbb{R}^m$ satisfies the Lipschitz condition with respect to y on a domain $(a, b) \times D$ where $D \subset \mathbb{R}^m$ if there exist a constant L so that*

$$\|f(t, y) - f(t, \tilde{y})\| \leq L\|y - \tilde{y}\|, \quad \text{for all } t \in (a, b), y, \tilde{y} \in D.$$

The constant L is called the Lipschitz constant.

It is not hard to show that the function f satisfies the Lipschitz condition if $\partial f_i / \partial y_j$, $i, j = 1, \dots, m$ are continuous and bounded on the domain and D is open and convex.

Theorem 2.2. *Consider the initial value problem*

$$y' = f(t, y), \quad y(t_0) = y_0. \tag{5}$$

If

1. $f(t, y)$ is continuous in $(a, b) \times D$,
2. $f(t, y)$ satisfies the Lipschitz condition with respect to y in $(a, b) \times D$.

with given initial values $t_0 \in (a, b)$ and $y_0 \in D$, then (5) has one and only one solution in $(a, b) \times D$.

Before concluding this section, we will include another definition, which is useful when we later will discuss stability:

Definition 2.3. The function $f : \mathbb{R}^m \rightarrow \mathbb{R}^m$ satisfy a one-sided Lipschitz condition with constant μ if

$$\langle f(y) - f(\tilde{y}), y - \tilde{y} \rangle \leq \mu \|y - \tilde{y}\|_2^2.$$

And the following theorem holds:

Theorem 2.4. If $f(t, y)$ satisfies the one-sided Lipschitz condition with constant μ , and $y(t)$, $\tilde{y}(t)$ both are solutions of $y' = f(t, y)$, then, for all $t \geq t_0$

$$\|y(t) - \tilde{y}(t)\|_2 \leq e^{\mu(t-t_0)} \|y(t_0) - \tilde{y}(t_0)\|_2.$$

Proof. We have

$$\begin{aligned} \frac{d}{dt} \|y(t) - \tilde{y}(t)\|_2^2 &= 2 \langle f(y(t)) - f(\tilde{y}(t)), y(t) - \tilde{y}(t) \rangle \\ &\leq 2\mu \|y(t) - \tilde{y}(t)\|_2^2 \end{aligned}$$

Multiplying by $e^{-2\mu(t-t_0)}$ gives us

$$\frac{d}{dt} \left(e^{-2\mu(t-t_0)} \|y(t) - \tilde{y}(t)\|_2^2 \right) = e^{-2\mu(t-t_0)} \left(\frac{d}{dt} \|y(t) - \tilde{y}(t)\|_2^2 - 2\mu \|y(t) - \tilde{y}(t)\|_2^2 \right) \leq 0,$$

so $e^{-\mu(t-t_0)} \|y(t) - \tilde{y}(t)\|_2$ is a nondecreasing function, which proves the theorem. \square

Flow of the system

We will sometimes use the term *flow of a system*, φ_t , defined as

$$\varphi_t(y_0) = \varphi_t \circ y_0 = y(t) \quad \text{if} \quad y(0) = y_0.$$

where $y(t)$ is the solution of the ODE. Thus you may think about ϕ_t as all possible solutions of the ODE.

3 Numerical solution of ODEs.

In this section we develop some simple methods for the solution of initial value problems. In both cases, let us assume that we somehow have found solutions $y_l \approx y(t_l)$, for $l = 0, 1, \dots, n$, and we want to find an approximation $y_{n+1} \approx y(t_{n+1})$ where $t_{n+1} = t_n + h$, where h is the stepsize. Basically, there are two different classes of methods in practical use.

1. *One-step methods.* Only y_n is used to find the approximation y_{n+1} . One-step methods usually require more than one function evaluation pr. step. They can all be put in a general abstract form

$$y_{n+1} = y_n + h\Phi(t_n, y_n; h).$$

2. *Linear multistep methods:* y_{n+1} is approximated from y_{n-k+1}, \dots, y_n .

3.1 Some examples of one-step methods.

Assume that t_n, y_n is known. The exact solution $y(t_{n+1})$ with $t_{n+1} = t_n + h$ of (4) passing through this point is given by

$$y(t_n + h) = y_n + \int_{t_n}^{t_{n+1}} y'(\tau) d\tau = y_n + \int_{t_n}^{t_{n+1}} f(\tau, y(\tau)) d\tau. \quad (6)$$

The idea is to find approximations to the last integral. The simplest idea is to use $f(\tau, y(\tau)) \approx f(t_n, y_n)$, in which case we get the Euler method again:

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

The integral can also be approximated by the trapezoidal rule

$$\int_{t_n}^{t_{n+1}} f(\tau, y(\tau)) d\tau = \frac{h}{2}(f(t_n, y_n) + f(t_{n+1}, y(t_{n+1}))).$$

By replacing the unknown solution $y(t_{n+1})$ by y_{n+1} we get the *trapezoidal method*

$$y_{n+1} = y_n + \frac{h}{2}(f(t_n, y_n) + f(t_{n+1}, y_{n+1})).$$

Here y_{n+1} is available by solving a (usually) nonlinear system of equations. Such methods are called implicit. To avoid this extra difficulty, we could replace y_{n+1} on the right hand side by the approximation from Euler's method, thus

$$\begin{aligned} \tilde{y}_{n+1} &= y_n + hf(t_n, y_n); \\ y_{n+1} &= y_n + \frac{h}{2}(f(t_n, y_n) + f(t_{n+1}, \tilde{y}_{n+1})). \end{aligned}$$

This method is called the *improved Euler method*. Similarly, we could have used the midpoint rule for the integral,

$$\int_{t_n}^{t_{n+1}} f(\tau, y(\tau)) d\tau = \left(f\left(t_n + \frac{h}{2}, y\left(t_n + \frac{h}{2}\right)\right) \right) h,$$

and replaced $y(t_n + \frac{h}{2})$ by one half Euler step. The result is the *modified Euler method*:

$$\begin{aligned} \tilde{y}_{n+\frac{1}{2}} &= y_n + \frac{h}{2}f(t_n, y_n), \\ y_{n+1} &= y_n + hf\left(t_n + \frac{h}{2}, \tilde{y}_{n+\frac{1}{2}}\right). \end{aligned}$$

Do we gain anything by constructing these methods? Let us solve the problem from Example 1.1 using improved/modified Euler with $h = 0.1$. For each step, also the global error $e_n =$

$y(t_n) - y_n$ is computed. For comparison, also the result for the Euler method is included.

t_n	Euler		improved Euler		modified Euler	
	y_n	e_n	y_n	e_n	y_n	e_n
0.0	1.000000	0	1.000000	0	1.000000	0
0.1	1.000000	$-9.95 \cdot 10^{-3}$	0.990000	$4.98 \cdot 10^{-5}$	0.990000	$4.98 \cdot 10^{-5}$
0.2	0.980000	$-1.92 \cdot 10^{-2}$	0.960696	$9.34 \cdot 10^{-5}$	0.960597	$1.92 \cdot 10^{-4}$
0.3	0.940800	$-2.69 \cdot 10^{-2}$	0.913814	$1.17 \cdot 10^{-4}$	0.913528	$4.03 \cdot 10^{-4}$
0.4	0.884352	$-3.22 \cdot 10^{-2}$	0.852040	$1.04 \cdot 10^{-4}$	0.851499	$6.45 \cdot 10^{-4}$
0.5	0.813604	$-3.48 \cdot 10^{-2}$	0.778765	$3.60 \cdot 10^{-5}$	0.777930	$8.71 \cdot 10^{-4}$
0.6	0.732243	$-3.46 \cdot 10^{-2}$	0.697773	$-9.69 \cdot 10^{-5}$	0.696636	$1.04 \cdot 10^{-3}$
0.7	0.644374	$-3.17 \cdot 10^{-2}$	0.612924	$-2.98 \cdot 10^{-4}$	0.611507	$1.12 \cdot 10^{-3}$
0.8	0.554162	$-2.69 \cdot 10^{-2}$	0.527850	$-5.58 \cdot 10^{-4}$	0.526202	$1.09 \cdot 10^{-3}$
0.9	0.465496	$-2.06 \cdot 10^{-2}$	0.445717	$-8.59 \cdot 10^{-4}$	0.443904	$9.54 \cdot 10^{-4}$
1.0	0.381707	$-1.38 \cdot 10^{-2}$	0.369053	$-1.17 \cdot 10^{-3}$	0.367153	$7.27 \cdot 10^{-4}$

As we can see, there is a significant improvement in accuracy, compared with the Euler method.

3.2 Some examples of linear multistep methods.

Most of the classical linear multistep methods are based on some kind of interpolation. In the followin, we use $f_i = f(t_i, y_i)$.

The Adams methods

Write the ODE (4) in integral form

$$y(t_n + h) = y(t_n) + \int_{t_n}^{t_n+h} f(\tau, y(\tau)) d\tau.$$

Let $p(t)$ be the polynomial interpolating $(t_i, f(t_i, y_i))$ for $i = n - k + 1, \dots, n$, and let

$$y_{n+1} = y_n + \int_{t_n}^{t_n+h} p(\tau) d\tau.$$

Methods constructed this way is called k -step Adams-Bashforth methods. Examples:

$$\begin{aligned} k = 1 : & & y_{n+1} &= y_n + hf_n \\ k = 2 : & & y_{n+1} &= y_n + \frac{h}{2}(3f_n - f_{n-1}) \end{aligned}$$

These methods are of order k .

Implicit Adams methods, or Adams-Moulton is constructed similarly, but the point t_{n+1}, f_{n+1} is included in the interpolation polynomial. Examples:

$$\begin{aligned} k = 0 : & & y_{n+1} &= y_n + hf_{n+1} \\ k = 1 : & & y_{n+1} &= y_n + \frac{h}{2}(f_{n+1} + f_n) \\ k = 2 : & & y_{n+1} &= y_n + \frac{h}{12}(5f_{n+1} + 8f_n - f_{n-1}) \end{aligned}$$

These are of order $k + 1$.

Backward differentiation formulas (BDF)

Let $p(t)$ be the polynomial interpolating (t_i, y_i) , $i = n - k + 1, \dots, n, n + 1$. The BDF methods are then simply

$$p'(t_{n+1}) = f_{n+1}$$

Examples:

$$\begin{aligned} k = 1 : & & y_{n+1} - y_n &= hf_{n+1} \\ k = 2 : & & \frac{1}{2}(3y_{n+1} - 2y_n + y_{n-1}) &= hf_{n+1} \end{aligned}$$

These methods are of order k , and have good stability properties. The MATLAB solver ODE15s is based on these formulas.

4 Runge-Kutta methods

The Euler method, as well as the improved and modified Euler methods are all examples on *explicit Runge-Kutta methods* (ERK). Such schemes are given by

$$\begin{aligned} k_1 &= f(t_n, y_n), \\ k_2 &= f(t_n + c_2h, y_n + ha_{21}k_1), \\ k_3 &= f(t_n + c_3h, y_n + h(a_{31}k_1 + a_{32}k_2)), \\ &\vdots \\ k_s &= f\left(t_n + c_sh, y_n + h \sum_{j=1}^{s-1} a_{sj}k_j\right), \\ y_{n+1} &= y_n + h \sum_{i=1}^s b_i k_i, \end{aligned} \tag{7}$$

where c_i , a_{ij} and b_i are coefficients defining the method. We always require $c_i = \sum_{j=1}^s a_{ij}$. Here, s is the number of *stages*, or the number of function evaluations needed for each step.

The vectors k_i are called stage derivatives. The improved Euler method is then a two-stage RK-method, written as

$$\begin{aligned} k_1 &= f(t_n, y_n), \\ k_2 &= f(t_n + h, y_n + hk_1), \\ y_{n+1} &= y_n + \frac{h}{2}(k_1 + k_2). \end{aligned}$$

Also implicit methods, like the trapezoidal rule,

$$y_{n+1} = y_n + \frac{h}{2}(f(t_n, y_n) + f(t_n + h, y_{n+1}))$$

can be written in a similar form,

$$\begin{aligned} k_1 &= f(t_n, y_n), \\ k_2 &= f(t_n + h, y_n + \frac{h}{2}(k_1 + k_2)), \\ y_{n+1} &= y_n + \frac{h}{2}(k_1 + k_2). \end{aligned}$$

But, contrary to what is the case for explicit methods, a nonlinear system of equations has to be solved to find k_2 .

Definition 4.1. An s -stage Runge-Kutta method is given by

$$\begin{aligned} k_i &= f(t_n + c_i h, y_n + h \sum_{j=1}^s a_{ij} k_j), \quad i = 1, 2, \dots, s, \\ y_{n+1} &= y_n + h \sum_{i=1}^s b_i k_i. \end{aligned}$$

The method is defined by its coefficients, which is given in a Butcher tableau

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & & & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array} \quad \text{or in short} \quad \begin{array}{c|c} c & A \\ \hline & b^T \end{array} \quad (8)$$

and we will always assume that

$$c_i = \sum_{j=1}^s a_{ij}, \quad i = 1, \dots, s \quad \text{or in short} \quad c = A\mathbf{1}.$$

The method is explicit if $a_{ij} = 0$ whenever $j \geq i$, otherwise implicit.

Example 4.1. The Butcher-tableaux for the methods presented so far are

$\begin{array}{c c} 0 & 0 \\ \hline & 1 \end{array}$	$\begin{array}{c cc} 0 & 0 & 0 \\ \hline 1 & 1 & 0 \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$	$\begin{array}{c cc} 0 & 0 & 0 \\ \hline \frac{1}{2} & \frac{1}{2} & 0 \\ \hline & 0 & 1 \end{array}$	$\begin{array}{c cc} 0 & 0 & 0 \\ \hline 1 & \frac{1}{2} & \frac{1}{2} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$
<i>Euler</i>	<i>improved Euler</i>	<i>modified Euler</i>	<i>trapezoidal rule</i>

When the method is explicit, the zeros on and above the diagonal is usually ignored. We conclude this section by presenting the maybe most popular among the RK-methods over times, *The 4th order Runge-Kutta method* (Kutta – 1901):

$$\begin{array}{lcl}
 k_1 & = & f(t_n, y_n) \\
 k_2 & = & f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1) \\
 k_3 & = & f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2) \\
 k_4 & = & f(t_n + h, y_n + hk_3) \\
 y_{n+1} & = & y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)
 \end{array}
 \quad \text{or} \quad
 \begin{array}{c|cccc}
 0 & & & & \\
 \frac{1}{2} & \frac{1}{2} & & & \\
 \frac{1}{2} & 0 & \frac{1}{2} & & \\
 1 & 0 & 0 & 1 & \\
 \hline
 & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6}
 \end{array}
 . \quad (9)$$

4.1 Order conditions for Runge-Kutta methods.

Theorem 4.2. *Let*

$$y' = f(t, y), \quad y(t_0) = y_0, \quad t_0 \leq t \leq t_{end}$$

be solved by a one-step method

$$y_{n+1} = y_n + h\Phi(t_n, y_n; h), \quad (10)$$

with stepsize $h = (t_{end} - t_0)/N_{step}$. If

1. *the increment function Φ is Lipschitz in y , and*
2. *the local truncation error $d_{n+1} = \mathcal{O}(h^{p+1})$,*

then the method is of order p , that is, the global error at t_{end} satisfies

$$e_{N_{step}} = y(t_{end}) - y_{N_{step}} = \mathcal{O}(h^p).$$

The proof is left as an exercise.

A RK method is a one-step method with increment function $\Phi(t_n, y_n; h) = \sum_{i=1}^s b_i k_i$. It is possible to show that Φ is Lipschitz in y whenever f is Lipschitz and $h \leq h_{max}$, where h_{max} is some predefined maximal stepsize. What remains is the order of the local truncation error. To find it, we take the Taylor-expansions of the exact and the numerical solutions and compare. The local truncation error is $\mathcal{O}(h^{p+1})$ if the two series matches for all terms corresponding to h^q with $q \leq p$. In principle, this is trivial. In practise, it becomes extremely tedious (give it a try). Fortunately, it is possible to express the two series very elegant by the use of *B-series* and *rooted trees*.

B-series and rooted trees

B-series in different forms, and under different names, is essential the main tool for constructing order theory for time-dependent problems, like ODEs, DAEs and SDEs. In this note, with a B-series we mean a formal series of the form

$$B(\varphi, x_0; h) = x_0 + \sum_{\tau \in \bar{T}} \alpha(\tau) \cdot \varphi(\tau)(h) \cdot F(\tau)(\mathbf{x}_0). \quad (11)$$

Here, T is a set of rooted trees, $\bar{T} = T \setminus \emptyset$ where \emptyset refer to the initial value term, $F(\tau)(x_0)$ the elementary differentials, $\varphi(\tau)(h)$ some integral, and $\alpha(\tau)$ is a symmetry factor. The idea is to express the solutions of the exact and the numerical solution after one step as B-series. For instance, consider the autonomous ODE $y' = f(y), y(t_0) = y_0$, and let us solve this by the Euler method. Thus we have

$$\begin{aligned} B(e, y_0; h) &= y(t_0 + h) = y(t_0) + hf(y_0) + \frac{1}{2}h^2 f'f + \dots \\ B(\phi, y_0; h) &= y_1 = y(t_0) + hf(y_0). \end{aligned}$$

So, if the these solution can be expressed as B-series, which we still have to prove, the first terms will be

τ	α	e	ϕ	F
\bullet	1	h	h	f
$\begin{array}{c} \bullet \\ \\ \bullet \end{array}$	1	$\frac{1}{2}h^2$	0	$f'f$

But at the moment, we do not know how the rest of the terms looks like.

Before a more formal derivation of the series, let present a few definitions and results:

Definition 4.3. Let $y = [y_1, y_2, \dots, y_m]^T \in \mathbb{R}^m$ and $f(y) = [f_1(y), f_2(y), \dots, f_m(y)]^T \in \mathbb{R}^m$. The κ 'th Frechet derivative of f , denoted by $f^{(\kappa)}(y)$ is a κ -linear operator $\mathbb{R}^m \times \mathbb{R}^m \times \dots \times \mathbb{R}^m$ (κ times) $\rightarrow \mathbb{R}^m$. Evaluation of component i of this operator working on the m operands $v_1, v_2, \dots, v_\kappa \in \mathbb{R}^m$ is given by

$$\left[f^{(\kappa)}(y)(v_1, v_2, \dots, v_\kappa) \right]_i = \sum_{j_1=1}^m \sum_{j_2=1}^m \dots \sum_{j_\kappa=1}^m \frac{\partial^\kappa f_i(y)}{\partial y_{j_1} \partial y_{j_2} \dots \partial y_{j_\kappa}} v_{1,j_1} v_{2,j_2} \dots v_{\kappa,j_\kappa}$$

where $v_l = [v_{l,1}, v_{l,2}, \dots, v_{l,m}] \in \mathbb{R}^m$ for $l = 1, 2, \dots, \kappa$.

Note that the κ 'th Frechet derivative is independent of permutations of its operands, thus e.g. $f'''(y)(v_1, v_2, v_3) = f'''(y)(v_3, v_1, v_2)$.

The multivariable Taylor expansion is, for $y, v \in \mathbb{R}^m$:

$$f(y + v) = f(y) + \sum_{\kappa=1}^{\infty} \frac{1}{\kappa!} f^{(\kappa)}(y)(v, v, \dots, v) = \sum_{\kappa=1}^{\infty} \frac{1}{\kappa!} f^{(\kappa)}(y)(v^\kappa), \quad (12)$$

the expression to the right is only a convenient way to write the expression in the middle.

Finally, the multinomial theorem states:

$$(v_1 + v_2 + \dots + v_q)^\kappa = \sum_{r_1 + \dots + r_q = \kappa} \frac{\kappa!}{r_1! \dots r_q!} v_1^{r_1} \dots v_q^{r_q}$$

A similar argument applied to the Frechet derivative gives

$$f^{(\kappa)}(y)(a_1 v_1 + a_2 v_2 + \dots + a_q v_q)^\kappa = \sum_{r_1 + \dots + r_q = \kappa} \frac{\kappa!}{r_1! \dots r_q!} \cdot \prod_{k=1}^{\kappa} a_k \cdot f^{(\kappa)}(y)(v_1^{r_1}, \dots, v_q^{r_q}) \quad (13)$$

where $\alpha_k \in \mathbb{R}$ and $v_k \in \mathbb{R}^m$.

A *list* of trees, denoted by $\{\tau_1, \tau_2, \dots, \tau_\kappa\}$, $\tau_i \in T$, $i = 1, \dots, \kappa$ is an ordered set of trees, where each tree might appear more than once. If $\tau_1, \tau_2 \in T$ then $\{\tau_1, \tau_2, \tau_1\}$ and $\{\tau_2, \tau_1, \tau_1\}$ are two different lists. If a tree appear k times in the list, the tree has multiplicity k . A *multiset* of trees, denoted by $(\tau_1, \tau_2, \dots, \tau_\kappa)$ is a set of trees where multiplicity is allowed and order does not matter. So $(\tau_1, \tau_2, \tau_1) = (\tau_2, \tau_1, \tau_1)$. A tree with multiplicity k will sometimes be denoted by τ^k , so $(\tau_1, \tau_2, \tau_1) = (\tau_1^2, \tau_2)$. The set of all possible lists of trees is denoted \tilde{U} , and the set of all possible multisets is denoted U :

$$\begin{aligned} \tilde{U} &= \{ \{\tau_1, \tau_2, \dots, \tau_\kappa\} : \tau_i \in T, \quad i = 1, \dots, \kappa, \quad \kappa = 0, 1, 2, \dots \}, \\ U &= \{ (\tau_1, \tau_2, \dots, \tau_\kappa) : \tau_i \in T, \quad i = 1, \dots, \kappa, \quad \kappa = 0, 1, 2, \dots \}. \end{aligned}$$

In the lemma below, U_f is the set of trees formed by taking a multiset from U and include a root f .

Lemma 4.1. *If $X(h) = B(\phi, x_0; h)$ is some B-series and $f \in C^\infty(\mathbb{R}^m, \mathbb{R}^m)$ then $f(X(h))$ can be written as a formal series of the form*

$$f(X(h)) = \sum_{u \in U_f} \beta(u) \cdot \psi_\phi(u)(h) \cdot G(u)(x_0) \quad (14)$$

where U_f is a set of trees derived from T , by

a) $[\emptyset]_f \in U_f$, and if $\tau_1, \tau_2, \dots, \tau_\kappa \in T$ then $[\tau_1, \tau_2, \dots, \tau_\kappa]_f \in U_f$.

b) $G([\emptyset]_f)(x_0) = f(x_0)$ and

$$G(u = [\tau_1, \dots, \tau_\kappa]_f)(x_0) = f^{(\kappa)}(x_0)(F(\tau_1)(x_0), \dots, F(\tau_\kappa)(x_0)).$$

c) $\beta([\emptyset]_f) = 1$ and $\beta(u = [\tau_1, \dots, \tau_\kappa]_f) = \frac{1}{r_1! r_2! \dots r_q!} \prod_{k=1}^{\kappa} \alpha(\tau_k)$,

where r_1, r_2, \dots, r_q count equal trees among $\tau_1, \tau_2, \dots, \tau_\kappa$.

d) $\psi_\phi([\emptyset]_f)(h) \equiv 1$ and $\psi_\phi(u = [\tau_1, \dots, \tau_\kappa]_f)(h) = \prod_{k=1}^{\kappa} \phi(\tau_k)(h)$.

Proof. Writing $X(h)$ as a B-series, we have

$$\begin{aligned}
f(X(h)) &= f\left(x_0 + \sum_{\tau \in \bar{T}} \alpha(\tau) \cdot \phi(\tau)(h) \cdot F(\tau)(x_0)\right) \\
&\stackrel{(12)}{=} \sum_{\kappa=0}^{\infty} \frac{1}{\kappa!} f^{(\kappa)}(x_0) \left(\sum_{\tau \in \bar{T}} \alpha(\tau) \cdot \phi(\tau)(h) \cdot F(\tau)(x_0)\right)^{\kappa} \\
&\stackrel{(13)}{=} f(x_0) + \sum_{\kappa=1}^{\infty} \frac{1}{\kappa!} \sum_{(\tau_1, \tau_2, \dots, \tau_{\kappa}) \in U} \frac{\kappa!}{r_1! r_2! \dots r_q!} \\
&\quad \cdot \left(\prod_{k=1}^{\kappa} \alpha(\tau_k) \cdot \phi(\tau_k)(h)\right) f^{(\kappa)}(x_0) (F(\tau_1)(x_0), \dots, F(\tau_{\kappa})(x_0)).
\end{aligned}$$

The number above the equal sign refer to the equation used. The last sum is taken over all possible unordered combinations of κ trees in T . For each set of trees $\tau_1, \tau_2, \dots, \tau_{\kappa} \in T$ we assign a $u = [\tau_1, \tau_2, \dots, \tau_{\kappa}]_f \in U_f$. The theorem is now proved by comparing term by term with (14). \square

To find the B-series of the exact solution, write the ODE in integral form:

$$y(t_0 + h) = y_0 + \int_0^h f(y(t_0 + s)) ds. \quad (15)$$

Assume that the exact solution can be written as a B-series

$$y(t_0 + h) = B(e, y_0; h). \quad (16)$$

Plug this into (15), apply Theorem 4.1 to get

$$y_0 + \sum_{\tau \in \bar{T}} \alpha(\tau) \cdot e(\tau)(h) \cdot F(\tau)(y_0) = y_0 + \sum_{u \in U_f} \beta(u) \cdot \int_0^h \psi_e(u)(s) ds \cdot G(u)(y_0).$$

For each term on the left hand side, there has to be a corresponding term on the right. Or for each $\tau = [\tau_1, \dots, \tau_{\kappa}] \in T$ there is a corresponding $u = [\tau_1, \dots, \tau_{\kappa}]_f \in U_f$, and $\alpha(\tau) = \beta(u)$, $F(\tau)(y_0) = G(u)(y_0)$ and finally $e(\tau)(h) = \int_0^h \psi_e(s) ds$.

This gives us the following theorem:

Theorem 4.4. *The exact solution of (4) can be written as a formal series of the form (16) with*

- i) $\emptyset \in T$, $\bullet = [\emptyset] \in T$, and if $\tau_1, \dots, \tau_{\kappa} \in T$ then $\tau = [\tau_1, \dots, \tau_{\kappa}] \in T$.
- ii) $F(\emptyset)(y_0) = y_0$, $F(\bullet) = f(y_0)$, and $F(\tau)(y_0) = f^{(\kappa)}(y_0) (F(\tau_1)(y_0), \dots, F(\tau_{\kappa})(y_0))$.
- iii) $\alpha(\emptyset) = 1$, $\alpha(\bullet) = 1$ and $\alpha(\tau) = \frac{1}{r_1! r_2! \dots r_q!} \prod_{k=1}^{\kappa} \alpha(\tau_k)$, where r_1, \dots, r_q counts equal trees among the subtrees $\tau_1, \dots, \tau_{\kappa}$.
- iv) $e(\emptyset)(h) = 1$, $e(\bullet)(h) = h$ and $e(\tau)(h) = \int_0^h \prod_{k=1}^{\kappa} e(\tau_k)(s) ds$.

Notice that $e(\tau)(h) = \frac{1}{\gamma(\tau)}h^{\rho(\tau)}$, where $\gamma(\tau)$ is an integer and $\rho(\tau)$ is the number of nodes. This is called *the order* of the tree τ .

To find the B-series of the numerical solution, write one step of the RK-method in the form

$$Y_i = y_0 + h \sum_{j=1}^s a_{ij} f(Y_j), \quad i = 1, \dots, s \quad (17)$$

$$y_1 = y_0 + h \sum_{i=1}^s b_i f(Y_i). \quad (18)$$

and assume that both the stage values Y_i and the numerical solutions can be written as

$$Y_i = B(\phi_i, y_0; h), \quad i = 1, \dots, s, \quad \text{and} \quad y_1 = B(\phi, y_0; h).$$

It is straightforward to see that $\phi_i(\emptyset)(h) = \phi(\emptyset)(h) = 1$ and

$$\phi_i(\bullet) = \sum_{j=1}^s a_{ij} h = c_i h, \quad \phi(\bullet)(h) = \sum_{i=1}^s b_i h.$$

For a general tree $\tau \in T$, insert the B-series for Y_i and y_1 into (18), apply Lemma 4.1 and compare equal terms. This results in the following recurrence formula for the weight functions $\phi_i(\tau)$ and $\phi(\tau)$ for a given $\tau = [\tau_1, \dots, \tau_\kappa]$:

$$\phi_i(\tau)(h) = \sum_{j=1}^s a_{ij} \prod_{k=1}^{\kappa} \phi_j(\tau_k)(h), \quad \phi(\tau)(h) = \sum_{i=1}^s b_i \prod_{k=1}^{\kappa} \phi_i(\tau_k)(h) \quad (19)$$









Notice again that $\phi(\tau)(h) = \hat{\phi}(\tau) \cdot h^{\rho(\tau)}$, where $\hat{\phi}(\tau)$ is a constant depending of the method coefficients. Similar, we can write $\phi_i(\tau)(h) = \hat{\phi}_i(\tau) \cdot h^{\rho(\tau)}$.

Comparing the series for the exact and the numerical solutions and applying Theorem 4.2 gives the following fundamental theorem:

Theorem 4.5. *A Runge-Kutta method is of order p if and only if*

$$\hat{\phi}(\tau) = \frac{1}{\gamma(\tau)}, \quad \forall \tau \in T, \quad \rho(\tau) \leq p.$$

All trees up to and including order 4 and their corresponding terms are listed below:

τ	$\rho(\tau)$	$\hat{\phi}(\tau) = 1/\gamma(\tau)$
	1	$\sum b_i = 1$
	2	$\sum b_i c_i = 1/2$
	3	$\sum b_i c_i^2 = 1/3$
		$\sum b_i a_{ij} c_j = 1/6$
	4	$\sum b_i c_i^3 = 1/4$
		$\sum b_i c_i a_{ij} c_j = 1/8$
		$\sum b_i a_{ij} c_j^2 = 1/12$
		$\sum b_i a_{ij} a_{jk} c_k = 1/24$

4.2 Error control and stepsize selection.

A user of some numerical black box software will usually require one thing: The accuracy of the numerical solution should be within some user specified tolerance. To accomplish this we have to measure the error, and if the error is too large, it has to be reduced. For ordinary differential equations, this means to reduce the stepsize. On the other hand, we would like our algorithm to be as efficient as possible, that is, to use large stepsizes. This leaves us with two problems: How to measure the error, and how to get the right balance between accuracy and efficiency.

Local error estimate. As demonstrated in Figure 1, the global error $y(t_n) - y_n$ comes from two sources: the local truncation error and the propagation of errors produced in preceding steps. This makes it difficult (but not impossible) to measure the global error. Fortunately it is surprisingly easy to measure the *local error*, l_{n+1} , the error produced in one step when starting at (t_n, y_n) , see Figure 2. Let $y(t; t_n, y_n)$ be the exact solution of the ODE through the point t_n, y_n . For a method of order p we get

$$l_{n+1} = y(t_n + h; t_n, y_n) - y_{n+1} = \Psi(t_n, y_n)h^{p+1} + \mathcal{O}(h^{p+2}),$$

where $\mathcal{O}(h^{p+2})$ refer to higher order terms ¹ The term $\Psi(t_n, y_n)h^{p+1}$ is called *the principal error term*, and we assume that this term is the dominating part of the error. This assumption

¹Strictly speaking, the Landau-symbol \mathcal{O} is defined by

$$f(x) = \mathcal{O}(g(x)) \quad \text{for } x \rightarrow x_0 \quad \text{if} \quad \lim_{x \rightarrow x_0} \frac{\|f(x)\|}{\|g(x)\|} < K < \infty$$

for some unspecified constant K . Thus $f(h) = \mathcal{O}(h^q)$ means that $\|f(h)\| \leq Kh^q$ when $h \rightarrow 0$, and refer to the remainder terms of a truncated series.

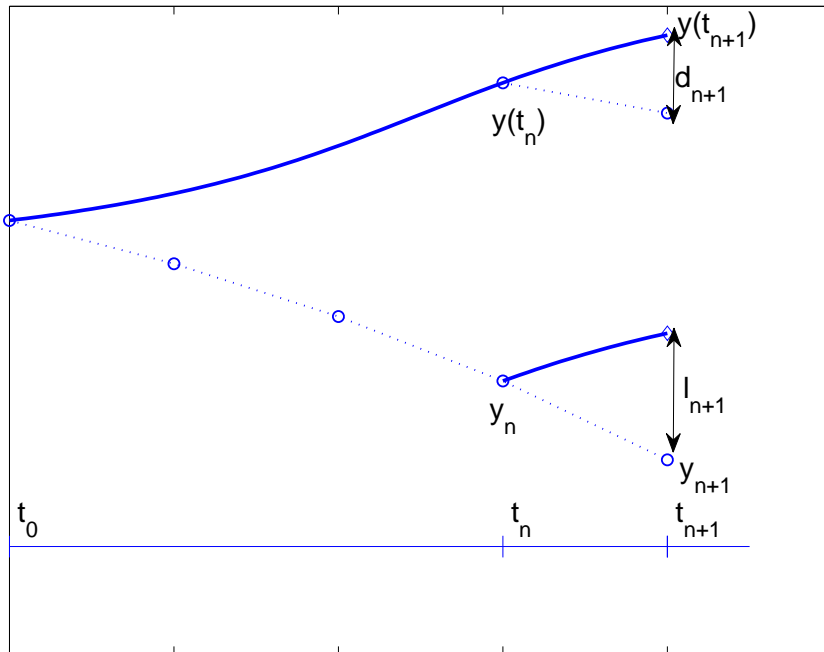


Figure 2: Lady Windermere's Fan

is true if the stepsize h is sufficiently small. Taking a step from the same point t_n, y_n with a method of order $\hat{p} = p + 1$ gives a solution \hat{y}_{n+1} with a local error satisfying

$$y(t_n + h; t_n, y_n) - \hat{y}_{n+1} = \mathcal{O}(h^{p+2}).$$

The *local error estimate* is given by

$$le_{n+1} = \hat{y}_{n+1} - y_{n+1} = \Psi(t_n, y_n)h^{p+1} + \mathcal{O}(h)^{p+2} \approx l_{n+1}.$$

Embedded Runge-Kutta pair Given a Runge-Kutta method of order p . To be able to measure the local error, we need a method of order $p + 1$ (or higher). But we do not want to spend more work (in terms of f -evaluations) than necessary. The solution is *embedded*

Runge-Kutta pairs, which, for explicit methods are given by

0					
c_2	a_{21}				
c_3	a_{31}	a_{32}			
\vdots	\vdots	\ddots			
c_s	a_{s1}	a_{s2}	\cdots	$a_{s,s-1}$	
	b_1	b_2	\cdots	b_{s-1}	
	\hat{b}_1	\hat{b}_2	\cdots	\hat{b}_{s-1}	\hat{b}_s

The method given by the b_i 's is of order p , the error estimating method given by the \hat{b}_i 's is of order $p + 1$. (Sometimes it is the other way round. The important thing is to have two methods of different order.) The local error estimate of y_{n+1} is then given by

$$le_{n+1} = \hat{y}_{n+1} - y_{n+1} = h \sum_{i=1}^s (\hat{b}_i - b_i) k_i.$$

Example 4.2. A combination of the Euler method and improved Euler will result in the following pair

0		
1	1	
	1	
	$\frac{1}{2}$	$\frac{1}{2}$

so that

$$k_1 = f(t_n, y_n), \quad k_2 = f(t_n + h, y_n + hk_1), \quad y_{n+1} = y_n + hk_1, \quad l_{n+1} \approx le_{n+1} = \frac{h}{2}(-k_1 + k_2).$$

Example 4.3. Assume that you have decided to use improved Euler, which is of order 2, as your advancing method, and you would like to find an error estimating method of order 3. There are no 2-stage order 3 ERKs, so you have to add one stage to your method. This gives a method like

0			
1	1		
c_3	a_{31}	a_{32}	
	$\frac{1}{2}$	$\frac{1}{2}$	
	\hat{b}_1	\hat{b}_2	\hat{b}_3

where we require $c_3 = a_{31} + a_{32}$, which give us five free parameters. These have to satisfy all four order condition for an order 3 method. Using c_3 as a free parameter, we get the following class of 3rd order methods:

$$b_1 = \frac{3c_3 - 1}{6c_3}, \quad b_2 = \frac{2 - 3c_3}{6(1 - c_3)}, \quad b_3 = \frac{1}{6c_3(1 - c_3)}, \quad a_{31} = c_3^2, \quad a_{32} = c_3 - c_3^2.$$

It is also possible to use the highest order method to advance the solution. In this case, we still measure the local error estimate of the lowest order order solution, but we get a more accurate numerical solution for free. This idea is called *local extrapolation*.

MATLAB has two integrators based on explicit Runge-Kutta schemes, ODE23 which is based on an order 3/2 pair by Bogacki and Shampine, (a 3th order advancing and a 2nd order error estimating method), and ODE45 based on an order 5/4 pair by Dormand and Prince. Both use local extrapolation.

Stepsize control Let the user specify a tolerance Tol , and a norm $\|\cdot\|$ in which the error is measured. Let us start with t_n, y_n , and do one step forward in time with a stepsize h_n , giving y_{n+1} and le_{n+1} . If $\|le_{n+1}\| \leq Tol$ the step is accepted, and we proceed till the next step, maybe with an increased stepsize. If $\|le_{n+1}\| > Tol$ the step is rejected and we try again with a smaller stepsize. In both cases, we would like to find a stepsize h_{new} which gives a local error estimate smaller than Tol , but at the same time as close to Tol as possible. To find the right stepsize, we make one assumption: The function $\Psi(t_n, y_n)$ of the principle error term do not change much from one step to the next, thus $\|\Psi(t_n, y_n)\| \approx \|\Psi(t_{n+1}, y_{n+1})\| \approx C$. Then

$$\text{we have:} \quad \|le_{n+1}\| \approx C \cdot h_n^{p+1}$$

$$\text{we want:} \quad Tol \approx C \cdot h_{new}^{p+1}$$

We get rid of the unknown C by dividing the two equations with each other, and h_{new} can be solved from

$$\frac{\|le_{n+1}\|}{Tol} \approx \left(\frac{h_n}{h_{new}} \right)^{p+1}.$$

Rejected steps are wasted work, and it should be avoided. Thus we choose the new stepsize somewhat conservative. The new stepsize is computed by

$$h_{new} = P \cdot \left(\frac{Tol}{\|le_{n+1}\|} \right)^{\frac{1}{p+1}} h_n. \quad (20)$$

where P is a *pessimist factor*, usually chosen somewhere in the interval [0.5,0.95]. In the discussion so far we have used the requirement $\|le_{n+1}\| \leq Tol$, that is *error pr. step* (EPS). This do not take into account the fact that the smaller the step is, the more steps you take, and the local errors from each step adds up. From this point of view, it would make sense to rather use the requirement $\|le\|_{n+1} \leq Tol \cdot h_n$, that is *error pr. unit step* (EPUS). The stepsize selection is then given by

$$h_{new} = P \cdot \left(\frac{Tol}{\|le_{n+1}\|} \right)^{\frac{1}{p}} h_n. \quad (21)$$

Careful analysis has proved that the local extrapolation together with EPS gives proportionality between the global error and the tolerance. The same is true for the use of the lower order method to advance the solution in combination with EPUS.

4.3 Dense output

Given a RK–method with coefficients (c, A, b) . To find approximations to solutions between the steps, that is, at some point $t = t_n + \theta h$, $\theta \in (0, 1)$, we search for a continuous RK–scheme, in which

$$y(t_n + \theta h) \approx u_n(\theta) = y_n + \sum_{i=1}^s b_i(\theta) k_i.$$

The coefficients $b_i(\theta)$ are now polynomials, and they should be determined so that

$$u_0(\theta) - y(x_0 + \theta h) = \mathcal{O}(h^{p^*+1}),$$

and $b_i(1) = b_i$, $i = 1, \dots, s$. This condition is satisfied if and only if

$$\rho(\tau) \sum_{i=1}^s b_i(\theta) \hat{\Phi}_i(\tau) = \theta^{\rho(\tau)}, \quad \forall \tau \in \mathcal{T}, \quad \rho(\tau) \leq p^*.$$

4.4 Collocation methods.

We will now see how the idea of orthogonal polynomials can be used to construct high order Runge-Kutta methods.

Given an ordinary differential equation

$$y'(t) = f(t, y(t)), \quad y(t_n) = y_n, \quad t \in [t_n, t_n + h].$$

Recall also the definition of a Runge–Kutta method applied to the ODE:

$$\begin{aligned} k_i &= f(t_n + c_i h, y_n + h \sum_{j=1}^s a_{ij} k_j) \\ y_{n+1} &= y_n + h \sum_{i=1}^s b_i k_i. \end{aligned} \tag{22}$$

The idea of collocation methods for ODEs is as follows: Assume that s distinct points $c_1, c_2, \dots, c_s \in [0, 1]$ are given. we will then search for a polynomial $u \in \mathbb{P}_s$ satisfying the ODE in the points $t_n + c_i h$, $i = 1, \dots, s$. These are the *collocation conditions*, expressed as

$$u'(t_n + c_i h) = f(t_n + c_i h, u(t_n + c_i h)), \quad i = 1, 2, \dots, s, \tag{23}$$

The initial condition is $u(t_n) = y_n$. When finally $u(t)$ has been found, we will set $y_{n+1} = u(t_n + h)$.

Let k_i be some (so far unknown) approximation to $y'(t_n + c_i h)$, and let $u' = p \in \mathbb{P}_{s-1}$ be the interpolation polynomial satisfying $p(t_n + c_i h) = k_i$. For simplicity, introduce the change of variables $t = t_n + \tau h$, $\tau \in [0, 1]$. Then

$$p(t) = p(t_n + h\tau) = \sum_{j=1}^s k_j \ell_j(\tau), \quad \ell_j(\tau) = \prod_{\substack{k=1 \\ k \neq j}}^s \frac{\tau - c_k}{c_j - c_k}.$$

The polynomial $u(t)$ is given by

$$u(\tilde{t}) = u(t_n) + \int_{t_n}^{\tilde{t}} p(t) dt = y_n + h \sum_{j=1}^s k_j \int_0^{\tilde{\tau}} \ell_j(\tau) d\tau, \quad \tilde{t} \in [t_n, t_n + h] \quad (\text{thus } \tilde{\tau} \in [0, 1])$$

The collocation condition becomes

$$u'(t_n + c_i h) = f \left(t_n + c_i h, y_n + h \sum_{j=1}^s k_j \int_0^{c_i} \ell_j(\tau) d\tau \right)$$

and in addition we get

$$u(t_n + h) = y_n + h \sum_{j=1}^s k_j \int_0^1 \ell_j(\tau) d\tau.$$

But $u'(t_n + c_i h) = k_i$, and $y_{n+1} = u(t_n + h)$ so this is exactly the Runge–Kutta method (22) with coefficients

$$a_{ij} = \int_0^{c_i} \ell_j(\tau) d\tau, \quad i, j = 1, \dots, s, \quad b_i = \int_0^1 \ell_i(\tau) d\tau, \quad i = 1, \dots, s. \quad (24)$$

Runge–Kutta methods constructed this way are called *collocation methods* and they are of order at least s . But a clever choice of the c_i 's will give better results.

Example 4.4. Let $P_2(x) = x^2 - 1/3$ be the second order Legendre polynomial. Let c_1 and c_2 be the zeros of $P_2(2x - 1)$ (using a change of variable to transfer the interval $[-1, 1]$ to $[0, 1]$), that is

$$c_1 = \frac{1}{2} - \frac{\sqrt{3}}{6}, \quad c_2 = \frac{1}{2} + \frac{\sqrt{3}}{6}.$$

The corresponding cardinal functions becomes

$$\ell_1(\tau) = -\sqrt{3} \left(\tau - \frac{1}{2} - \frac{\sqrt{3}}{6} \right), \quad \ell_2(\tau) = \sqrt{3} \left(\tau - \frac{1}{2} + \frac{\sqrt{3}}{6} \right).$$

From (24) we get a Runge–Kutta method with the following Butcher tableau:

$$\begin{array}{c|cc} \frac{1}{2} - \frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{1}{4} - \frac{\sqrt{3}}{6} \\ \frac{1}{2} + \frac{\sqrt{3}}{6} & \frac{1}{4} + \frac{\sqrt{3}}{6} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}.$$

This method has order 4 (check it yourself).

Famous collocation methods can be constructed by choosing c_i as the zeros of a polynomial P :

- Gauss-Legendre methods: $P(x) = \frac{d^s}{dx^s} x^s (x^s - 1)$, order $2s$.
- Radau IA: $P(x) = \frac{d^{s-1}}{dx^{s-1}} (x^s (x - 1)^{s-1})$.
Radau IIA: $P(x) = \frac{d^{s-1}}{dx^{s-1}} (x^{s-1} (x - 1)^s)$.
Both of order $2s - 1$.
- Lobatto IIIA: $P(x) = \frac{d^{s-2}}{dx^{s-2}} (x^{s-1} (x - 1)^{s-1})$, of order $2s - 2$.

Simplifying assumptions

These are tools for reducing the number of order conditions, and to make it easier to construct higher order methods:

$$B(p) : \sum_{i=1}^s b_i c_i^{q-1} = \frac{1}{q}, \quad q = 1, 2, \dots, p \quad (25a)$$

$$C(\eta) : \sum_{j=1}^s a_{ij} c_j^{q-1} = \frac{c_i^q}{q}, \quad i = 1, \dots, s, \quad q = 1, 2, \dots, \eta \quad (25b)$$

$$D(\zeta) : \sum_{i=1}^s b_i c_i^{q-1} a_{ij} = \frac{b_j}{q} (1 + c_j^q), \quad j = 1, \dots, s \quad q = 1, \dots, \zeta \quad (25c)$$

Theorem 4.6. *If $B(p)$, $C(\eta)$ and $D(\zeta)$ are satisfied with $p \leq 2\eta + 2$ and $p \leq \eta + \zeta + 1$, then the method is of order p .*

For a proof, see [1, II.7]

The collocation methods mentioned above all satisfy $C(s)$.

5 Stiff equations

Example 5.1. *Given the ODE*

$$y' = -1000y, \quad y(0) = 1.$$

with exact solution

$$y(t) = e^{-1000t}.$$

Thus $y(t) \rightarrow 0$ as $t \rightarrow \infty$. The Euler method applied to this problem yields

$$y_{n+1} = y_n - 1000h y_n = (1 - 1000h) y_n.$$

so that $y_n = (1 - 1000h)^n$. This gives us two situations:

If $|1 - 1000h| < 1$ then $y_n \rightarrow 0$ as $n \rightarrow \infty$.

If $|1 - 1000h| > 1$ then $|y_n| \rightarrow \infty$ as $n \rightarrow \infty$

Clearly, the second situation does not make sense at all, as the numerical solution is unstable even if the exact solution is stable. We have to choose a stepsize $h < 0.002$ to get a stable numerical solution for this problem.

To be more general: Consider a linear ODE

$$y' = My, \quad y(0) = y_0, \quad (26)$$

where M is a constant, $m \times m$ matrix. We assume that M is diagonalizable, that is

$$V^{-1}MV = \Lambda$$

where

$$\Lambda = \text{diag}\{\lambda_1, \lambda_2, \dots, \lambda_m\}, \quad V = [v_1, v_2, \dots, v_m],$$

where λ_i , $i = 1, \dots, m$ are the eigenvalues of M and v_i are the corresponding eigenvectors. By premultiplying (26) with V^{-1} , we get

$$V^{-1}y' = V^{-1}MVV^{-1}y, \quad V^{-1}y(t_0) = V^{-1}y_0$$

or, using $u = V^{-1}y$,

$$u' = \Lambda u, \quad u(t_0) = V^{-1}y_0 = u_0.$$

The system is now decoupled, and can be written componentwise as

$$u'_i = \lambda_i u_i, \quad u_i(0) = u_{i,0}, \quad \lambda_i \in \mathbb{C}, \quad i = 1, \dots, m. \quad (27)$$

We have to accept the possibility of complex eigenvalues, however, as M is a real matrix, then complex eigenvalues appears in complex conjugate pairs. In the following, we will consider the situation when

$$\text{Re}(\lambda_i) < 0 \text{ for } i = 1, \dots, m, \quad \text{thus} \quad y(t) \rightarrow 0 \text{ as } t \rightarrow \infty. \quad (28)$$

Apply the Euler method to (26):

$$y_{n+1} = y_n + hMy_n.$$

We can do exactly the same linear transformations as above, so the system can be rewritten as

$$u_{i,n+1} = (1 + h\lambda_i)u_{i,n}, \quad i = 1, \dots, m.$$

For the numerical solution to be stable, we have to require

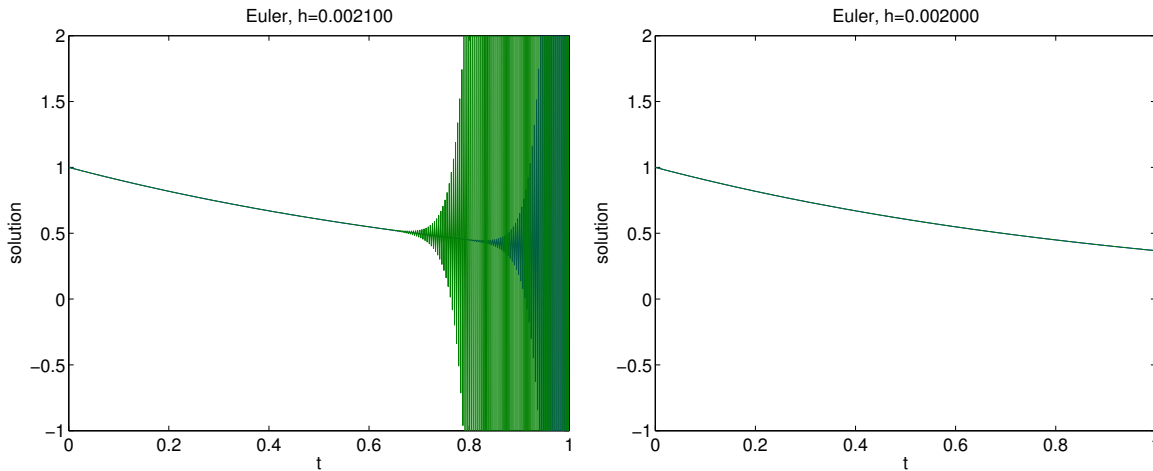
$$|1 + h\lambda_i| \leq 1, \quad \text{for all the eigenvalues } \lambda_i. \quad (29)$$

(The case $|1 + h\lambda_i h| = 1$ is included, as this is sufficient to prevent the solution from growing.)

Example 5.2. *Given*

$$y' = \begin{bmatrix} -2 & 1 \\ 998 & -999 \end{bmatrix} y, \quad y(0) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

with exact solution $y_1(t) = y_2(t) = e^{-t}$. The matrix has eigenvalues -1 and -1000 . The initial values are chosen so that the fast decaying mode is missing in the exact solution. This problem is solved by Eulers method, with two almost equal stepsizes, $h = 0.0021$ and $h = 0.002$. The difference is striking, but completely in correspondence with (29) and the result of Example 5.1.



Example 5.2 is a typical example of a stiff equation. The stepsize is restricted by a fast decaying component.

Example 5.3. *Let*

$$M = \begin{bmatrix} -2 & -2 \\ 1 & 0 \end{bmatrix} \quad \text{with eigenvalues} \quad \lambda_{1,2} = -1 \pm i.$$

The requirement (29) becomes

$$|1 + h(-1 \pm i)| \leq 1 \quad \text{or} \quad (1 - h)^2 + h^2 \leq 1 \quad \text{which is satisfied if and only if} \quad 0 \leq h \leq 1.$$

Stiffness occurs in situations with fast decaying solutions (transients) in combination with slow solutions. If you solve an ODE by an adaptive explicit scheme, and the stepsize becomes unreasonable small, stiffness is the most likely explanation. If the stepsizes in additions seems to be independent of your choice of tolerances, then you can be quite sure. The stepsize is restricted by stability related to the transients, and not by accuracy. The *backward Euler* method is one way to overcome this problem:

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1}) \tag{30}$$

or, applied to the problem of (27)

$$u_{i,n+1} = u_{i,n} + h\lambda_i u_{i,n+1}, \quad \Rightarrow \quad u_{i,n+1} = \frac{1}{1 - h\lambda_i} u_{i,n}.$$

Since $|1/(1 - h\lambda_i)| \leq 1$ whenever $\text{Re}(\lambda_i) \leq 0$ there is no stepsize restriction caused by stability issues. In fact, $u_{i,n+1} \rightarrow 0$ as $\text{Re}(h\lambda_i) \rightarrow -\infty$, so fast transients decay quickly, as they are supposed to do. But this nice behaviour is not for free: for a nonlinear ODE a nonlinear system of equations has to be solved for each step. We will return to this topic later.

5.1 Linear stability.

Given the linear test equation

$$y' = \lambda y, \quad \lambda \in \mathbb{C}. \quad (31)$$

Thus $\lambda = \alpha + i\beta$. The solution can be expressed by

$$y(t_n + h) = e^{\alpha h} e^{i\beta h} y(t_n).$$

Clearly, the solution is stable if $\alpha \leq 0$, that is $\lambda \in \mathbb{C}^-$. For the numerical solution we then require the stepsize h to be chosen so that

$$|y_{n+1}| \leq |y_n| \quad \text{whenever } \lambda \in \mathbb{C}^- \quad (32)$$

When a RK method is applied (31), we simply get

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

where R is a polynomial or a rational function. R is called *the stability function* of the RK method. The numerical solution is stable if $|R(z)| \leq 1$, otherwise it is unstable. This motivates the following definition of *the region of absolute stability* as

$$\mathcal{D} = \{z \in \mathbb{C} : |R(z)| \leq 1\}.$$

The condition (32) is satisfied for all $h > 0$ if

$$\mathbb{C}^- \subseteq \mathcal{D},$$

Methods satisfying this condition are called *A-stable*. The Backward Euler method (30) is an example of an *A-stable* method.

Example 5.4. A 2-stage ERK applied to (31) is given by:

$$k_1 = \lambda y_n, \quad k_2 = \lambda(y_n + ha_{21}\lambda y_n), \quad y_{n+1} = y_n + h\lambda(b_1 + b_2)y_n + (h\lambda)^2 b_2 a_{21} y_n$$

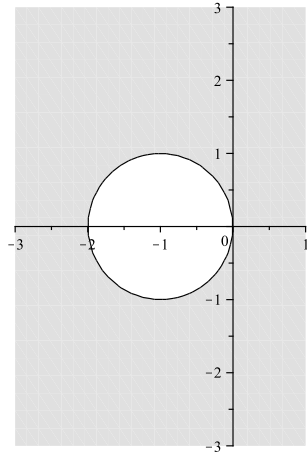
If this method is of order 2, then $b_1 + b_2 = 1$ and $b_2 a_{21} = 1/2$, so that

$$R(z) = 1 + z + \frac{1}{2}z^2.$$

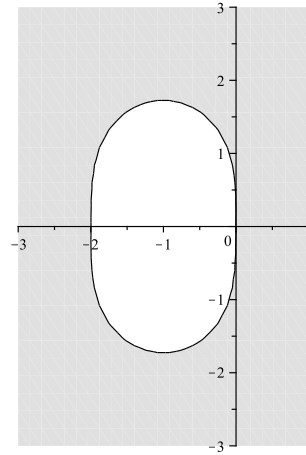
The stability function of an s -stage ERKs is a polynomial of degree s . As a consequence, no ERKs can be *A-stable*! If the order of the method is s , then

$$R(z) = \sum_{i=0}^s \frac{z^i}{i!}.$$

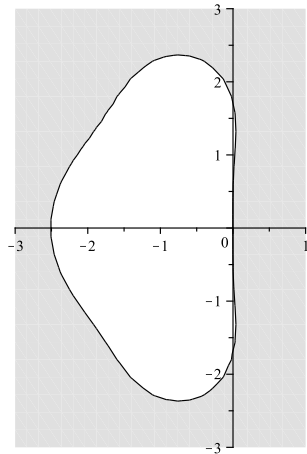
See Figure 3 for plots of the stability regions. But it has been proved that ERK with $p = s$ only exist for $s \leq 4$. To get an order 5 ERK, 6 stages are needed.



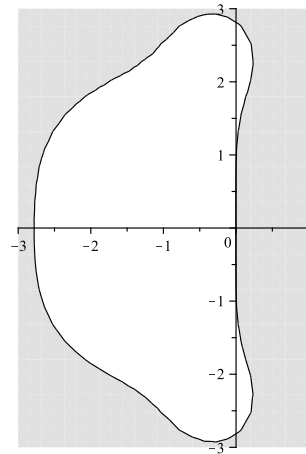
$$p = s = 1$$



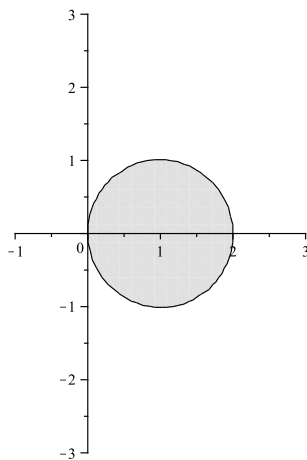
$$p = s = 2$$



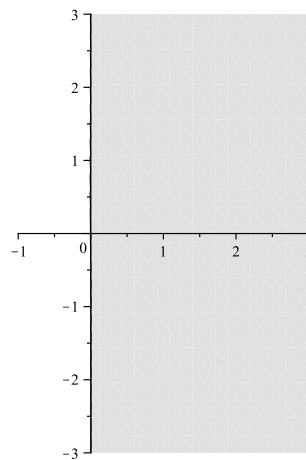
$$p = s = 3$$



$$p = s = 4$$



Backward Euler



Trapezoidal rule

Figure 3: Stability regions in \mathbb{C}^- : The first four are the stability regions for explicit RK methods of order $p = s$. The white regions are stable, the grey unstable.

Example 5.5. The trapezoidal rule (see section 3.1) applied to (31) gives

$$y_{n+1} = y_n + \frac{h}{2}(\lambda y_n + \lambda y_{n+1}) \quad \Rightarrow \quad R(z) = \frac{1+z}{1-z}.$$

In this case $\mathcal{D} = \mathbb{C}^-$, which is perfect.

To find a general expression of the stability function, apply a RK-method to the linear test problem (31):

$$k_i = \lambda(y_n + h \sum_{j=1}^s a_{ij} k_j), \quad i = 1, \dots, s$$

$$y_{n+1} = y_n + h \sum_{i=1}^s b_i$$

or, in matrix form with $K = [k_1, \dots, k_s]^T$ as

$$K = \lambda(y_n \mathbb{1} + hAK), \quad y_{n+1} = y_n + hb^T K.$$

Solving the linear system with respect to K , and inserting this into the expression for y_{n+1} gives

$$y_{n+1} = R(z)y_n \quad \text{with} \quad R(z) = 1 + zb^T(I - zA)^{-1}\mathbb{1}. \quad (33)$$

To summarise:

- For a given $\lambda \in \mathbb{C}^-$, choose a stepsize h so that $h\lambda \in \mathcal{D}$.
- If your problem is stiff, use an A -stable method.
- There are no A -stable explicit methods.

The Gauss methods, as well as the Radau methods are A -stable.

The *stability constant* is defined as

$$R(\infty) = \lim_{|z| \rightarrow \infty} R(z).$$

For a method with a invertible coefficient matrix A , the stability constant is

$$R(\infty) = 1 - b^T A^{-1} \mathbb{1}$$

Definition 5.1. A RK-method is called L -stable if it is A -stable and in addition $R(\infty) = 0$.

L -stable methods are appropriate for solving problems for which $Re(\lambda) \ll 0$.

Definition 5.2. A method is stiffly accurate if

$$b_i = a_{si}, \quad i = 1, \dots, s.$$

Notice that in this case $c_s = 1$. For stiffly accurate methods b^T is the last row of A , and thus $b^T A^{-1}$ is the last row of the identity matrix, we can conclude that

$$R(\infty) = 0 \quad \text{for all stiffly accurate methods.}$$

Be aware that this is *not* sufficient for A -stability.

5.2 Nonlinear stability

The next step is to extend the concept of linear stability to nonlinear problems. Consider an ODE

$$y' = f(y)$$

satisfying the condition

$$\langle f(y) - f(z), y - z \rangle \leq 0 \tag{34}$$

for all $y, z \in D \in \mathbb{R}^m$. Here, $\langle \cdot, \cdot \rangle$ denotes the Euclidean inner product. Let $y(t)$ and $z(t)$ be two solutions of an ODE. If (34) is satisfied, then

$$\|y(t) - z(t)\| \leq \|y(t_0) - z(t_0)\|$$

assuming the inner product norm. This can be proved by

$$\frac{d}{dt} \|y(t) - z(t)\|^2 = 2\langle f(y(t)) - f(z(t)), y(t) - z(t) \rangle \leq 0.$$

Definition 5.3. A Runge-Kutta method is *B-stable* if for all $h \geq 0$ and for all problems satisfying (34), the numerical solutions satisfies

$$\|y_1 - z_1\| \leq \|y_0 - z_0\|$$

Let M be the matrix with elements $m_{ij} = b_i a_{ij} + b_j a_{ji} - b_i b_j$, for $i, j = 1, \dots, s$.

Theorem 5.4. If $b_i \geq 0$ for $i = 1, \dots, s$ and M is positive semidefinite, then the RK-method is *B-stable*.

For a proof, see e.g. [2, IV.12]. Runge-Kutta methods satisfying the criterias of Theorem 5.4 are often called *algebraically stable*.

Example 5.6. The Gauss-methods as well as the Radau methods are all *B-stable*. So is the method

$$\begin{array}{c|ccc} \frac{5}{6} & \frac{5}{6} & & \\ \frac{29}{108} & -\frac{61}{108} & \frac{5}{6} & \\ \frac{1}{6} & -\frac{23}{183} & -\frac{33}{61} & \frac{5}{6} \\ \hline & \frac{26}{61} & \frac{324}{671} & \frac{1}{11} \end{array} \tag{35}$$

The method (35) is called *NTI* (from Nørsett and Thomsen).

5.3 Order reduction

Stiff problems may also suffer from another phenomena, the observed order can be lower than the theoretical order of the method. This can be demonstrated by the Prothero and Robinson example:

$$y' = \lambda(y - g(t)) + g'(t), \quad \lambda \in \mathbb{C}^-. \tag{36}$$

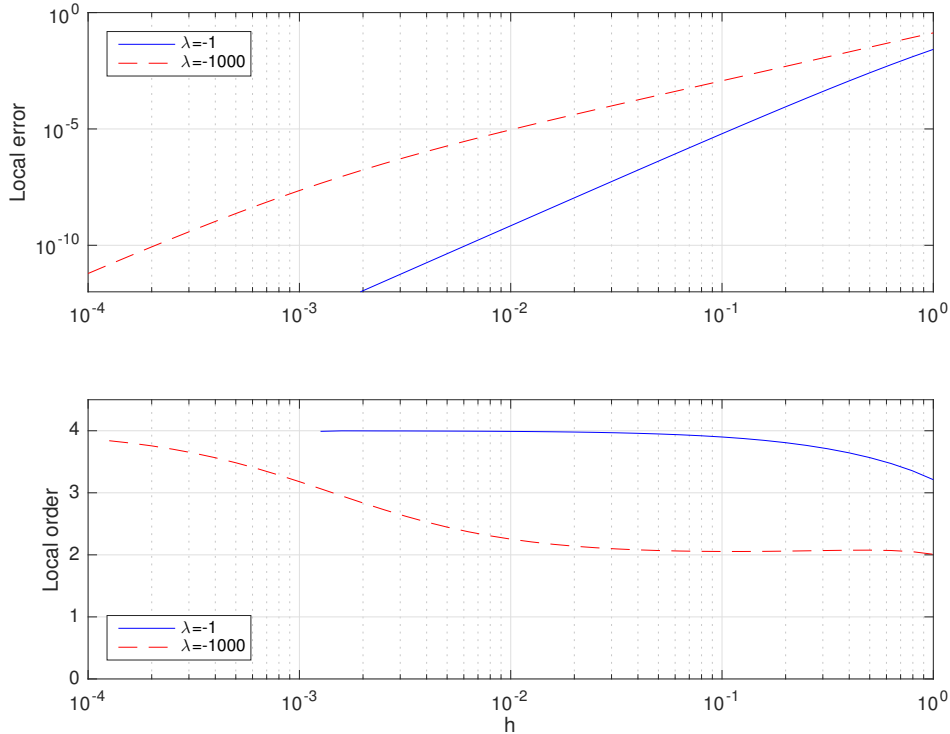


Figure 4: Local error and local order when the Prothero-Robinson example is solved by NTI.

The exact solution of this problem is

$$y(t) = e^{\lambda(t-t_0)}(y_0 - g(t_0)) + g(t)$$

so the solution consist of a smooth solution $g(t)$ and a (normally fast) transient towards this solution.

Example 5.7. Let $g(t) = \sin(t)$, $t_0 = 1$ and $y_0 = \sin(t_0)$. Do one step by the order 3 NTI method given in (35), and measure the local error and the local order. We expect to see order 4 ($p + 1$). The results of the experiment is given in Figure 4. So, for $\lambda = -1$, the result is as expected, but for $\lambda = -1000$, the order is only 2 for $h > 0.01$, then it slowly grows to the expected order as h becomes smaller.

How can we explain this? One step of a RK-method applied to (36) is given by

$$k_i = \lambda(y_0 + h \sum_j a_{ij} k_j - g(t_0 + c_i h)) + g'(t_0 + c_i h) \quad i = 1, \dots, s$$

$$y_1 = y_0 + \sum_i b_i k_i.$$

Let $K = [k_1, \dots, k_s]^T$, $G = [g(t_0 + c_1 h), \dots, g(t_0 + c_s h)]^T$ and $G' = [g'(t_0 + c_1 h), \dots, g'(t_0 +$

$c_s h)]^T$

$$\begin{aligned} K &= \lambda(\mathbb{1}_s y_0 + hAK - G) + G' \\ y_1 &= y_0 + hb^T K \end{aligned}$$

After some manipulation, we find that this can be written as

$$y_1 = R(z)(y_0 - g(t_0)) + g(t_0) + b^T(I - zA)^{-1}(z(g(t_0)\mathbb{1} - G) + hG') \quad (37)$$

where $z = \lambda h$ and $R(z) = 1 + zb^T(1 - zA)^{-1}\mathbb{1}$. The local truncation error is then

$$\hat{d}_1 = y(t_0 + h) - y_1 = (e^z - R(z))(y_0 - g(t_0)) + \hat{d}_1.$$

In our experiment, $y_0 = g(t_0)$, so the first term did not give any contribution to the error. So let us look at

$$\hat{d}_1 = g(t_0 + h) - g(t_0) - b^T(I - zA)^{-1}(z(g(t_0)\mathbb{1} - G) + hG')$$

The local order is found by Taylor expansions. The Taylor-expansions of each element of G and G' gives

$$G = \sum_{q=0}^{\infty} \frac{h^q}{q!} c^q g^{(q)}(t_0), \quad G' = \sum_{q=0}^{\infty} \frac{h^q}{q!} c^q g^{(q+1)}(t_0) = \sum_{q=1}^{\infty} \frac{h^{q-1}}{(q-1)!} c^{q-1} g^{(q)}(t_0),$$

where $c^q = [c_1^q, \dots, c_s^q]^T$. Thus (taking the minus sign into account)

$$z(G - g(t_0)\mathbb{1}) - hG' = \sum_{q=1}^{\infty} (zc^q - qc^{(q-1)}) \frac{h^q}{q!} g^{(q)}(t_0)$$

so

$$\hat{d}_1 = \sum_{q=1}^{\infty} \left(1 + b^T(I - zA)^{-1}(zc^q - qc^{(q-1)})\right) \frac{h^q}{q!} g^{(q)}(t_0) = \sum_{q=1}^{\infty} \psi_q(z) \frac{h^q}{q!} g^{(q)}(t_0) \quad (38)$$

The problem now is that we have to deal with a situation where z is large, while h is small. If this is not the case, the next step would be to find the power series of $(I - zA)^{-1}$, that is the Neumann series

$$(I - zA)^{-1} = \sum_{k=0}^{\infty} (zA)^{-k}.$$

Using this series, $z = h\lambda$ and collecting equal terms of h will give some of the classical order conditions. (Why only some of them?). But the Neumann series only converges for $\|zA\| \leq 1$, that is for

$$h \leq \frac{1}{|\lambda|} \frac{1}{\|A\|}.$$

In general, $\|A\| \sim 1$ (for NTII $\|A\|_2 = 1.22$). So, for $\lambda = -1$, this is $h < 1$, and we observe that in this case, the order is about 4, as expected. For $\lambda = -1000$, the order is approaching 4 only when $h < 0.001$, so this seems to be correct.

But how to get a theory for $h > 1/|\lambda|$? Write

$$(I - hA)^{-1} = -(zA)^{-1} \left(I - (zA)^{-1} \right)^{-1} = - \sum_{k=1}^{\infty} (zA)^{-k}.$$

The Neumann series used here will converge as long as

$$\|(zA)^{-1}\| = \frac{1}{|z|} \|A^{-1}\| < 1 \quad \Rightarrow \quad h > \frac{1}{|\lambda|} \|A^{-1}\|. \quad (39)$$

Using (39) in (38) we get

$$\begin{aligned} \psi_q(z) &= 1 + b^T (I - zA)^{-1} (zc^q - qc^{(q-1)}) = 1 - b^T \sum_{k=1}^{\infty} (zA)^{-k} (zc^q - qc^{(q-1)}) \\ &= 1 - b^T A^{-1} c^q - \sum_{k=1}^{\infty} \frac{1}{z^k} b^T A^{-k-1} (c^q - pAc^{q-1}). \end{aligned} \quad (40)$$

Since $c = A\mathbb{1}$, we can conclude that $\psi_1 = 0$.

Theorem 5.5. *If the simplifying assumptions $B(p)$ and $C(\eta)$ is satisfied, then*

$$\psi_q = 0, \quad q = 1, \dots, \min(p, \eta).$$

The proof is left as an exercise. See (25) for the definition of the simplifying assumptions. Write them in matrix form before proving the theorem.

Example 5.8. *(Example 5.7 continued.) For the NTI method we have*

$$\psi_2(z) = 0.2274 - \frac{1}{z} \cdot 0.7058 + \mathcal{O}\left(\frac{1}{z^2}\right).$$

Assuming $|z|$ large, the first term dominates the solution, and the order will be 2. As we observe in the experiment.

Comment: In such order analysis, we always assume that we can identify one term that dominates the error. However, as h becomes larger, and/or $|z|$ becomes smaller, the influence of higher order terms are more significant. This explains the smooth transition from order 2 to the classical order 4.

Finally, for stiffly accurate methods, the first term $1 - b^T A^{-1} c^q = 0$ for all q . So, for a method satisfying $C(1)$ we get

$$\psi_2(z) = -\frac{1}{z} b^T A^{-2} (c^2 - 2Ac) + \mathcal{O}\left(\frac{1}{z^2}\right).$$

Since $z = \lambda h$ the order is reduced to 1, but since $|z|$ is large, the error will be small.

Example 5.9. *Repeat the experiment from Example 5.7, but now with the Alexander's method, given by*

$$\begin{array}{c|cc} \alpha & & \alpha \\ \hline 1 & 1 - \alpha & \alpha \\ \hline & 1 - \alpha & \alpha \end{array}, \quad \alpha = 1 - \frac{\sqrt{2}}{2}$$

The order of this method is classical order 2. In our experiment, we have used $\lambda = -10^5$. The result is given in Figure 5.

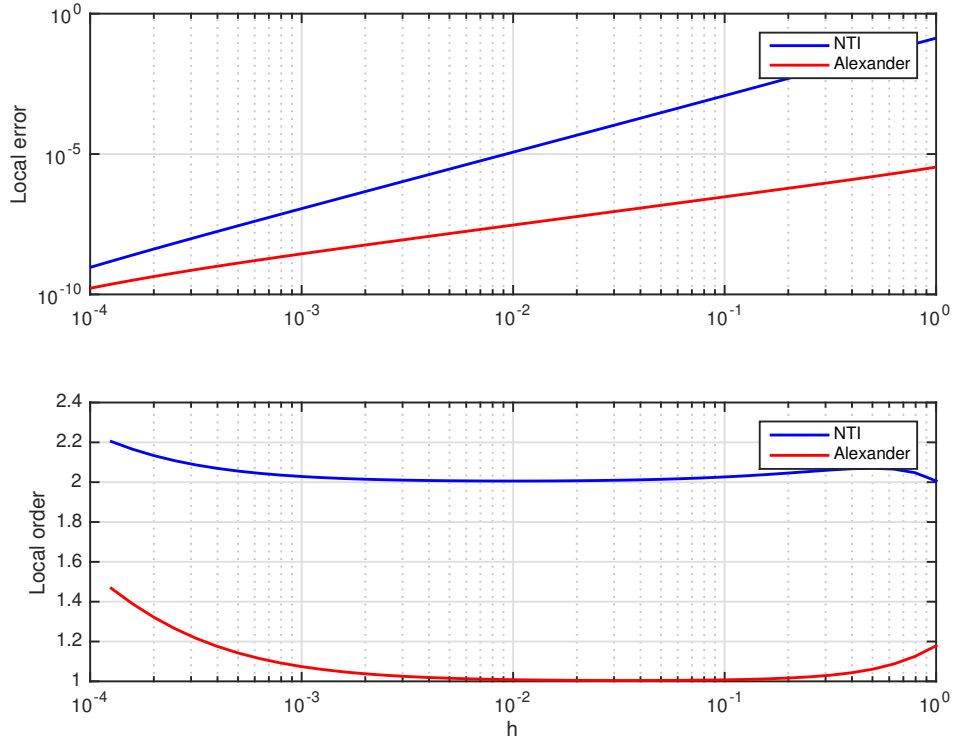


Figure 5: Local error and local order when the Prothero-Robinson example is solved by NTI and Alexander’s method. Here, $\lambda = 10^5$.

The severe order reduction is clearly visible, but, on the other hand, Alexander’s method clearly produces a smaller error.

6 Linear multistep methods

A k -step linear multistep method (LMM) applied to the ODE

$$y' = f(t, y), \quad y(t_0) = y_0, \quad t_0 \leq t \leq t_{end}.$$

is given by

$$\sum_{l=0}^k \alpha_l y_{n+l} = h \sum_{l=0}^k \beta_l f_{n+l}, \quad (41)$$

where α_l, β_l are the method coefficients, $f_j = f(t_j, y_j)$ and $t_j = t_0 + jh$, $h = (t_{end} - t_0)/N_{step}$. Usually we require

$$\alpha_k = 1 \quad \text{and} \quad |\alpha_0| + |\beta_0| \neq 0.$$

To get started with a k -step method, we also need starting values $y_l \approx y(t_l)$, $l = 0, 1, \dots, k-1$. A method is explicit if $\beta_k = 0$, otherwise implicit. The *leapfrog method*

$$y_{n+2} - y_n = 2hf(t_{n+1}, y_{n+1}) \quad (42)$$

and the method given by

$$y_{n+2} - y_{n+1} = h \left(\frac{3}{2}f_{n+1} - \frac{1}{2}f_n \right) \quad (43)$$

are both examples of explicit 2-step methods.

Example 6.1. Given the problem

$$y' = -2ty, \quad y(0) = 1$$

with exact solution $y(t) = e^{-t^2}$. Let $h = 0.1$, and $y_1 = e^{-h^2}$. This problem is solved by (43), and the numerical solution and the error is given by

t_n	y_n	$ e_n $
0.0	1.000000	0.00
0.1	0.990050	0.00
0.2	0.960348	$4.41 \cdot 10^{-4}$
0.3	0.912628	$1.30 \cdot 10^{-3}$
0.4	0.849698	$2.45 \cdot 10^{-3}$
0.5	0.775113	$3.69 \cdot 10^{-3}$
0.6	0.692834	$4.84 \cdot 10^{-3}$
0.7	0.606880	$5.75 \cdot 10^{-3}$
0.8	0.521005	$6.29 \cdot 10^{-3}$
0.9	0.438445	$6.41 \cdot 10^{-3}$
1.0	0.361746	$6.13 \cdot 10^{-3}$

6.1 Consistency and order.

We define the *local discretization error* $\tau_{n+k}(h)$ by

$$h\tau_{n+k}(h) = \sum_{l=0}^k (\alpha_l y(t_{n+l}) - h\beta_l y'(t_{n+l})). \quad (44)$$

You can think about the $h\tau_{n+k}$ as the defect obtained when plugging the exact solution into the difference equation (41). A method is *consistent* if $\tau_{n+k}(h) \xrightarrow{h \rightarrow 0} 0$. The term $h\tau_{n+k}(h)$ can be written as a power series in h

$$h\tau_{n+k}(h) = C_0 y(t_n) + C_1 h y'(t_n) + C_2 h^2 y''(t_n) + \dots + C_q h^q y^{(q)}(t_n) + \dots,$$

by expanding $y(t_n + lh)$ and $y'(t_n + lh)$ into their Taylor series around t_n ,

$$\begin{aligned} y(t_n + lh) &= y(t_n) + (lh)y'(t_n) + \frac{1}{2}(lh)^2 y''(t_n) + \dots + \frac{(lh)^q}{q!} y^{(q)}(t_n) + \dots \\ y'(t_n + lh) &= y'(t_n) + (lh)y''(t_n) + \frac{1}{2}(lh)^2 y'''(t_n) + \dots + \frac{(lh)^{q-1}}{q-1!} y^{(q)}(t_n) + \dots \end{aligned}$$

for sufficiently differentiable solutions $y(t)$. Insert this into (44), get the following expressions for C_q :

$$C_0 = \sum_{l=0}^k \alpha_l, \quad C_q = \frac{1}{q!} \sum_{l=0}^k (l^q \alpha_l - q l^{q-1} \beta_l), \quad q = 1, 2, \dots \quad (45)$$

The method is consistent if $C_0 = C_1 = 0$. It is of order p if

$$C_0 = C_1 = \dots = C_p = 0, \quad C_{p+1} \neq 0.$$

The constant C_{p+1} is called the *error constant*.

Example 6.2. The LMM (43) is defined by

$$\alpha_0 = 0, \quad \alpha_1 = -1, \quad \alpha_2 = 1, \quad \beta_0 = -\frac{1}{2}, \quad \beta_1 = \frac{3}{2}, \quad \beta_2 = 0,$$

thus

$$\begin{aligned} C_0 &= \alpha_0 + \alpha_1 + \alpha_2 = 0. \\ C_1 &= \alpha_1 + 2\alpha_2 - (\beta_0 + \beta_1 + \beta_2) = 0 \\ C_2 &= \frac{1}{2!} (\alpha_1 + 2^2 \alpha_2 - 2(\beta_1 + 2\beta_2)) = 0 \\ C_3 &= \frac{1}{3!} (\alpha_1 + 2^3 \alpha_2 - 3(\beta_1 + 2^2 \beta_2)) = \frac{5}{12}. \end{aligned}$$

The method is consistent and of order 2.

Example 6.3. Is it possible to construct an explicit 2-step method of order 3? There are 4 free coefficients $\alpha_0, \alpha_1, \beta_0, \beta_1$, and 4 order conditions to be solved ($C_0 = C_1 = C_2 = C_3 = 0$). The solution is

$$\alpha_0 = -5, \quad \alpha_1 = 4, \quad \beta_0 = 2, \quad \beta_1 = 4.$$

Test this method on the ODE of Example 2.1. (Replace the method coefficients in `lmm.m`.) The result is nothing but disastrous. Taking smaller steps only increase the problem.

To see why, you have to know a bit about how to solve difference equations.

6.2 Linear difference equations

A linear difference equation with constant coefficients is given by

$$\sum_{l=0}^k \alpha_l y_{n+l} = \varphi_n, \quad n = 0, 1, 2, \dots \quad (46)$$

The solution of this equation is a sequence $\{y_n\}$ of numbers (or vectors). Let $\{\tilde{y}_n\}$ be the general solution of the homogeneous problem

$$\sum_{l=0}^k \alpha_l y_{n+l} = 0. \quad (47)$$

Let ψ_n be one particular solution of (46). The general solution of (46) is then $\{y_n\}$ where $y_n = \tilde{y}_n + \psi_n$. To find a unique solution, we will need the starting values y_0, y_1, \dots, y_{k-1} .

Let us try $\tilde{y}_n = r^n$ as a solution of the homogeneous equation (47). This is true if

$$\sum_{l=0}^k \alpha_l r^{n+l} = r^n \sum_{l=0}^k \alpha_l r^l = 0.$$

The polynomial $\rho(r) = \sum_{l=0}^k \alpha_l r^l$ is called *the characteristic polynomial*, and $\{r^n\}$ is a solution of (47) if r is a root of $\rho(r)$. The k th degree polynomial $\rho(r)$ has k roots altogether, r_1, r_2, \dots, r_k , they can be distinct and real, they can be distinct and complex, in which case they appear in complex conjugate pairs, or they can be multiple. In the latter case, say $r_1 = r_2 = \dots = r_\mu$ we get a set of linear independent solutions $\{r_1^n\}, \{nr_1^n\}, \dots, \{n^{\mu-1}r_1^n\}$. Altogether we have found k linear independent solutions $\{\tilde{y}_{n,l}\}$ of the homogeneous equation, and the general solution is given by

$$y_n = \sum_{l=1}^k \kappa_l \tilde{y}_{n,l} + \psi_n.$$

The coefficients κ_l can be determined from the starting values.

Example 6.4. *Given*

$$\begin{aligned} y_{n+4} - 6y_{n+3} + 14y_{n+2} - 16y_{n+1} + 8y_n &= n \\ y_0 = 1, y_1 = 2, y_2 = 3, y_3 = 4. \end{aligned}$$

The characteristic polynomial is given by

$$\rho(r) = r^4 - 6r^3 + 14r^2 - 16r + 8$$

with roots $r_1 = r_2 = 2, r_3 = 1 + i, r_4 = 1 - i$. As a particular solution we try $\psi_n = an + b$. Inserted into the difference equation we find this to be a solution if $a = 1, b = 2$. The general solution has the form

$$y_n = \kappa_1 2^n + \kappa_2 n 2^n + \kappa_3 (1 + i)^n + \kappa_4 (1 - i)^n + n + 2.$$

From the starting values we find that $\kappa_1 = -1, \kappa_2 = \frac{1}{4}, \kappa_3 = -i/4$ and $\kappa_4 = i/4$. So, the solution of the problem is

$$\begin{aligned} y_n &= 2^n \left(\frac{n}{4} - 1 \right) - \frac{i(1+i)^n}{4} + \frac{i(1-i)^n}{4} + n + 2 \\ &= 2^n \left(\frac{n}{4} - 1 \right) - 2^{\frac{n-2}{2}} \sin \left(\frac{n\pi}{4} \right) + n + 2. \end{aligned}$$

Example 6.5. *The homogeneous part of the difference equation of Example 6.3 is*

$$\rho(r) = r^2 + 4r - 5 = (r - 1)(r + 5).$$

One root is 5. Thus, one solution component is multiplied by a factor -5 for each step, independent of the stepsize. Which explain why this method fails.

6.3 Zero-stability and convergence

Let us start with the definition of convergence. As before, we consider the error at t_{end} , using N_{step} steps with constant stepsize $h = (t_{end} - t_0)/N_{step}$.

Definition 6.1.

- A linear multistep method (41) is convergent if, for all ODEs satisfying the conditions of Theorem 2.2 we get

$$y_{N_{step}} \xrightarrow{h \rightarrow 0} y(t_{end}), \quad \text{whenever} \quad y_l \xrightarrow{h \rightarrow 0} y(t_0 + lh), \quad l = 0, 1, \dots, k-1.$$

- The method is convergent of order p if, for all ODEs with f sufficiently differentiable, there exists a positive h_0 such that for all $h < h_0$

$$\|y(t_{end}) - y_{N_{step}}\| \leq Kh^p \quad \text{whenever} \quad \|y(t_0 + lh) - y_l\| \leq K_0 h^p, \quad l = 0, 1, \dots, k-1.$$

The first characteristic polynomial of an LMM (41) is

$$\rho(r) = \sum_{l=0}^k \alpha_l r^l,$$

with roots r_1, r_2, \dots, r_k . From the section on difference equation, it follows that for the boundedness of the solution y_n we require:

1. $|r_i| \leq 1$, for $i = 1, 2, \dots, k$.
2. $|r_i| < 1$ if r_i is a multiple root.

A method satisfying these two conditions is called *zero-stable*.

We can now state (without proof) the following important result:

Theorem 6.2. (Dahlquist)

$$\text{Convergence} \quad \Leftrightarrow \quad \text{Zero-stability} + \text{Consistency}.$$

For a consistent method, $C_0 = \sum_{l=0}^k \alpha_l = 0$ so the characteristic polynomial $\rho(r)$ will always have one root $r_1 = 1$.

The zero-stability requirement puts a severe restriction on the maximum order of a convergent k -step method:

Theorem 6.3. (The first Dahlquist-barrier) The order p of a zero-stable k -step method satisfies

$$\begin{aligned} p &\leq k + 2 && \text{if } k \text{ is even,} \\ p &\leq k + 1 && \text{if } k \text{ is odd,} \\ p &\leq k && \text{if } \beta_k \leq 0. \end{aligned}$$

Notice that the last line include all explicit LMMs.

6.4 Adams-Bashforth-Moulton methods

The most famous linear multistep methods are constructed by the means of interpolation. For instance by the following strategy:

The solution of the ODE satisfy the integral equation

$$y(t_{n+1}) - y(t_n) = \int_{t_n}^{t_{n+1}} f(t, y(t)) dt. \quad (48)$$

Assume that we have found $f_i = f(t_i, y_i)$ for $i = n - k + 1, \dots, n$, with $t_i = t_0 + ih$. Construct the polynomial of degree $k - 1$, satisfying

$$p_{k-1}(t_i) = f(t_i, y_i), \quad i = n - k + 1, \dots, n.$$

The interpolation points are equidistributed (constant stepsize), so Newton's backward difference formula can be used in this case (see Exercise 2), that is

$$p_{k-1}(t) = p_{k-1}(t_n + sh) = f_n + \sum_{j=1}^{k-1} (-1)^j \binom{-s}{j} \nabla^j f_n$$

where

$$(-1)^j \binom{-s}{j} = \frac{s(s+1)\cdots(s+j-1)}{j!}$$

and

$$\nabla^0 f_n = f_n, \quad \nabla^j f_n = \nabla^{j-1} f_n - \nabla^{j-1} f_{n-1}.$$

Using $y_{n+1} \approx y(t_{n+1})$, $y_n \approx y(t_n)$ and $p_{k-1}(t) \approx f(t, y(t))$ in (48) gives

$$\begin{aligned} y_{n+1} - y_n \int_{t_n}^{t_{n+1}} p_{k-1}(t) dt &= h \int_0^1 p_{k-1}(t_n + sh) ds \\ &= h f_n + h \sum_{j=1}^{k-1} \left((-1)^j \int_0^1 \binom{-s}{j} ds \right) \nabla^j f_n. \end{aligned} \quad (49)$$

This gives the *Adams-Bashforth methods*

$$y_{n+1} - y_n = h \sum_{j=0}^{k-1} \gamma_j \nabla^j f_n, \quad \gamma_0 = 1, \quad \gamma_j = (-1)^j \int_0^1 \binom{-s}{j} ds.$$

Example 6.6. We get

$$\gamma_0 = 1, \quad \gamma_1 = \int_0^1 s ds = \frac{1}{2}, \quad \gamma_2 = \int_0^1 \frac{s(s+1)}{2} ds = \frac{5}{12}$$

and the first few methods becomes:

$$\begin{aligned} y_{n+1} - y_n &= h f_n \\ y_{n+1} - y_n &= h \left(\frac{3}{2} f_n - \frac{1}{2} f_{n-1} \right) \\ y_{n+1} - y_n &= h \left(\frac{23}{12} f_n - \frac{4}{3} f_{n-1} + \frac{5}{12} f_{n-2} \right) \end{aligned}$$

A k -step Adams-Bashforth method is explicit, has order k (which is the optimal order for explicit methods) and it is zero-stable. In addition, the error constant $C_{p+1} = \gamma_k$. Implicit Adams methods are constructed similarly, but in this case we include the (unknown) point (t_{n+1}, f_{n+1}) into the set of interpolation points. So the polynomial

$$p_k^*(t) = p_k^*(t_n + sh) = f_{n+1} + \sum_{j=1}^k (-1)^j \binom{-s+1}{j} \nabla^j f_{n+1}$$

interpolates the points (t_i, f_i) , $i = n - k + 1, \dots, n + 1$. Using this, we get the *Adams-Moulton* methods

$$y_{n+1} - y_n = h \sum_{j=0}^k \gamma_j^* \nabla^j f_{n+1}, \quad \gamma_0^* = 1, \quad \gamma_j^* = (-1)^j \int_0^1 \binom{-s+1}{j} ds.$$

Example 6.7. We get

$$\gamma_0^* = 1, \quad \gamma_1^* = \int_0^1 (s-1) ds = -\frac{1}{2}, \quad \gamma_2^* = \int_0^1 \frac{(s-1)s}{2} ds = -\frac{1}{12}$$

and the first methods becomes

$$y_{n+1} - y_n = h f_{n+1} \quad (\text{Backward Euler})$$

$$y_{n+1} - y_n = h \left(\frac{1}{2} f_{n+1} + \frac{1}{2} f_n \right) \quad (\text{Trapezoidal method})$$

$$y_{n+1} - y_n = h \left(\frac{5}{12} f_{n+1} + \frac{2}{3} f_n - \frac{1}{12} f_{n-1} \right).$$

A k -step Adams-Moulton method is implicit, of order $k + 1$ and is zero-stable. The error constant $C_{p+1} = \gamma_{k+1}^*$. Despite the fact that the Adams-Moulton methods are implicit, they have some advantages compared to their explicit counterparts: They are of one order higher, the error constants are much smaller, and the linear stability properties (when the methods are applied to the linear test problem $y' = \lambda y$) are much better.

k	0	1	2	3	4	5	6
γ_k	1	$\frac{1}{2}$	$\frac{5}{12}$	$\frac{3}{8}$	$\frac{251}{720}$	$\frac{95}{288}$	$\frac{19087}{60480}$
γ_k^*	1	$-\frac{1}{2}$	$-\frac{1}{12}$	$-\frac{1}{24}$	$-\frac{19}{720}$	$-\frac{3}{160}$	$-\frac{863}{60480}$

Table 1: The γ 's for the Adams methods.

6.5 Predictor-corrector methods

A predictor-corrector (PC) pair is a pair of one explicit (predictor) and one implicit (corrector) methods. The nonlinear equations from the application of the implicit method are solved by a fixed number of fixed point iterations, using the solution by the explicit method as starting values for the iterations.

Example 6.8. We may construct a PC method from a second order Adams-Bashforth scheme and the trapezoidal rule as follows:

$$y_{n+1}^{[0]} = y_n + \frac{h}{2}(3f_n - f_{n-1}) \quad (P : \text{Predictor})$$

for $l = 0, 1, \dots, m$

$$f_{n+1}^{[l]} = f(t_{n+1}, y_{n+1}^{[l]}) \quad (E : \text{Evaluation})$$

$$y_{n+1}^{[l+1]} = y_n + \frac{h}{2}(f_{n+1}^{[l]} + f_n) \quad (C : \text{Corrector})$$

end

$$y_{n+1} = y_{n+1}^{[m]}$$

$$f_{n+1} = f(t_{n+1}, y_{n+1}). \quad (E : \text{Evaluation})$$

Such schemes are commonly referred as $P(EC)^m E$ schemes.

The predictor and the corrector is often by the same order, in which case only one or two iterations are needed.

Error estimation in predictor-corrector methods.

The local discretization error of some LMM is given by

$$h\tau_{n+1} = \sum_{l=0}^k (\alpha_l y(t_{n-k+1+l}) - h\beta_l y'(t_{n-k+1+l})) = h^{p+1} C_{p+1} y^{(p+1)}(t_{n-k+1}) + \mathcal{O}(h^{p+2}).$$

But we can do the Taylor expansions of y and y' around t_n rather than t_{n-k+1} . This will not alter the principal error term, but the terms hidden in the expression $\mathcal{O}(h^{p+2})$ will change. As a consequence, we get

$$h\tau_{n+1} = h^{p+1} C_{p+1} y^{(p+1)}(t_n) + \mathcal{O}(h^{p+2}).$$

Assume that $y_i = y(t_i)$ for $i = n - k + 1, \dots, n$, and $\alpha_k = 1$. Then

$$h\tau_{n+1} = y(t_{n+1}) - y_{n+1} + \mathcal{O}(h^{p+2}) = h^{p+1} C_{p+1} y^{(p+1)}(t_n) + \mathcal{O}(h^{p+2}).$$

Assume that we have chosen a predictor-corrector pair, using methods of the same order p . Then

$$(P) \quad y(t_{n+1}) - y_{n+1}^{[0]} \approx h^{p+1} C_{p+1}^{[0]} y^{(p+1)}(t_n),$$

$$(C) \quad y(t_{n+1}) - y_{n+1} \approx h^{p+1} C_{p+1} y^{(p+1)}(t_n),$$

and

$$y_{n+1} - y_{n+1}^{[0]} \approx h^{p+1} (C_{p+1}^{[0]} - C_{p+1}) y^{(p+1)}(t_n).$$

From this we get the following local error estimate for the corrector, called *Milne's device*:

$$y(t_{n+1}) - y_{n+1} \approx \frac{C_{p+1}}{C_{p+1}^{[0]} - C_{p+1}} (y_{n+1} - y_{n+1}^{[0]}).$$

Example 6.9. Consider the PC-scheme of Example 6.8. In this case

$$C_{p+1}^{[0]} = \frac{5}{12}, \quad C_{p+1} = -\frac{1}{12}, \quad \text{so} \quad \frac{C_{p+1}}{C_{p+1}^{[0]} - C_{p+1}} = -\frac{1}{6}.$$

Apply the scheme to the linear test problem

$$y' = -y, \quad y(0) = 1,$$

using $y_0 = 1$, $y_1 = e^{-h}$ and $h = 0.1$. One step of the PC-method gives

l	$y_2^{[l]}$	$ y_2 - y_2^{[l]} $	$ y(0.2) - y_2^{[l]} $	$\frac{1}{6} y_2^{[l]} - y_2^{[0]} $
0	0.819112	$4.49 \cdot 10^{-4}$	$3.81 \cdot 10^{-4}$	
1	0.818640	$2.25 \cdot 10^{-5}$	$9.08 \cdot 10^{-5}$	$7.86 \cdot 10^{-5}$
2	0.818664	$1.12 \cdot 10^{-6}$	$6.72 \cdot 10^{-5}$	$7.47 \cdot 10^{-5}$
3	0.818662	$5.62 \cdot 10^{-8}$	$6.84 \cdot 10^{-5}$	$7.49 \cdot 10^{-5}$

After 1-2 iterations, the iteration error is much smaller than the local error, and we also observe that Milne's device gives a reasonable approximation to the error.

Remark Predictor-corrector methods are not suited for stiff problems. You can see this by e.g. using the trapezoidal rule on $y' = \lambda y$. The trapezoidal rule has excellent stability properties. But the iteration scheme

$$y_{n+1}^{[l+1]} = y_n + \frac{h}{2} \lambda (y_{n+1}^{[l]} + y_n)$$

will only converge if $|h\lambda/2| < 1$.

For stiff system, the *Backward differentiation formulas* (BDF) is to be preferred. Those are derived in exercise 5.

References

- [1] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving ordinary differential equations. I*, volume 8 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, second revised edition, 1993. Nonstiff problems.
- [2] E. Hairer and G. Wanner. *Solving ordinary differential equations. II*, volume 14 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 2010. Stiff and differential-algebraic problems, Second revised edition.