Numerical Methods for ODE's and DDE's

L.M. Abia
Dpto. Matemática Aplicada y Computación
University of Valladolid
Spain

Abstract

These lecture notes give an introductorial review to the subject of Numerical Solution of Ordinary Differential Equations (ODEs) and Delay Differential equations. The emphasis is in the more mathematical aspects and numerical experiences illustrating the different issues will be present in the corresponding lectures.

1 Introduction

We will consider in these lecture notes the numerical solution of the initial value problem for a system of ordinary differential equations of the form

$$\mathbf{y}' = \mathbf{f}(x, \mathbf{y}), \quad a \leq x \leq b, \quad \mathbf{y}(a) = \mathbf{A}, \tag{1}$$

with $\mathbf{y} = [y^1, \ldots, y^d]^T$, $\mathbf{f}(x, \mathbf{y}) = [f^1(x, \mathbf{y}), \ldots, f^d(x, \mathbf{y})]^T$, and a given initial data $\mathbf{A} = [A^1, \ldots, A^d]^T$ in $\mathbb{R}^d$. When we will refer to problem (1) we will always assume that $\mathbf{f} : [a, b] \times \mathbb{R}^d \to \mathbb{R}^d$ is well defined and continuous in $[a, b] \times \mathbb{R}^d$ and that satisfy a Lipschitz condition with respect to $\mathbf{y}$ in $[a, b] \times \mathbb{R}^d$ with Lipschitz constant L. These conditions are sufficient to guarantee the existence of a unique solution $\mathbf{y}(x)$ of (1) in $[a, b]$ (Theorem of Picard).

All the numerical methods we will study for the problem (1) proceed in a step-by-step manner by computing successively numerical approximations $\mathbf{y}_n \approx \mathbf{y}(x_n), n = 0, 1, \ldots, N$ to the values of the solution $\mathbf{y}(x)$ of (1) in a discrete set of nodes

$$a = x_0 < x_1 < x_2 < \ldots < x_N = b. \tag{2}$$

We refer to the sequence $\{\mathbf{y}_n\}_{n=0}^N$ as the numerical solution. Typically the numerical method is defined recursively by a rule (difference equation) that generates a numerical value $\mathbf{y}_{n+1}$ using $k \geq 1$ consecutive previously computed numerical approximations $\mathbf{y}_{n-j}, j = 0, 1, \ldots, k-1$. We say then that we have a k step method.

One-step methods start with an initial value $\mathbf{y}_0 \approx \mathbf{A}$ and then generate an approximation $\mathbf{y}_1$ to the value of the solution $\mathbf{y}(x_1)$ at $x = x_1$. We say that the numerical solution has been advanced one step of length h_0 from $x_0 = a$ to $x_1 = x_0 + h_0$. This process is repeated again and again to advance the numerical solution with step sizes $h_n, n = 1, \ldots, N-1$, from $x = x_n$ to $x_{n+1} = x_n + h_n$, $n = 1, \ldots, N-1$, respectively until we get $x_N = b$. For a k step method, $k > 1$, we also say that the method advances one step of length $h_n := x_{n+1} - x_n$ when we

compute $\mathbf{y}_{n+1}$ in terms of the k previous approximations $\mathbf{y}_n, \mathbf{y}_{n-1}, \ldots, \mathbf{y}_{n-k+1}$. A k-step method needs k starting values $\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_{k-1}$ to be well defined. Usually $\mathbf{y}_0 \approx \mathbf{A}$, but some additional procedure must be provided to get the starting values $\mathbf{y}_1, \ldots, \mathbf{y}_{k-1}$.

An important situation is when the numerical method advances the solution with a constant step size h. Then we say that the grid

$$x_n = a + nh, \quad n = 0, \ldots, N, \quad h = \frac{b-a}{N},$$

is uniform and that h is the diameter of the grid. This is usually an academic situation because modern codes take advantage of the possibility of to change the step size h_n as the numerical integration of the problem is progressing to adapt the numerical solution to the local properties of the problem. For non uniform grids, the diameter h of the grid is also defined by

$$h = \max(h_0, h_1, \ldots, h_{N-1}) = \max_{0 \le n \le N-1} h_n.$$

The Euler method is a first example of a numerical method for (1) and is given by the formula

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h_n \mathbf{f}(x_n, \mathbf{y}_n), \quad n = 0, \ldots, N-1. \tag{3}$$

The implicit Euler method is defined by the formula

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h_n \mathbf{f}(x_{n+1}, \mathbf{y}_{n+1}), \quad n = 0, \ldots, N-1, \tag{4}$$

and requires to solve at each step a nonlinear system of equations for the numerical solution $\mathbf{y}_{n+1}$. In this case, the existence of the numerical solution is not a trivial problem. Other examples of one-step methods will be shown in next sections.

2 One-step methods for Ordinary Differential Equations

2.1 The Taylor series method

The Taylor series methods arise quite naturally for solving numerically the system of ordinary differential equations (1). Starting with a numerical approximation $(x_n, \mathbf{y}_n)$ they approximate the local solution $\mathbf{u}(x)$ given by

$$\mathbf{u}' = \mathbf{f}(x, \mathbf{u}), \quad x > x_n, \qquad \mathbf{u}(x_n) = \mathbf{y}_n, \tag{5}$$

by truncation after the h^p term of the Taylor series expansion of $\mathbf{u}$ at $x = x_n$. An example of Taylor series method of first order is the Euler method (3). We get a second order formula by retaining through terms in h^2

$$\mathbf{u}(x_{n+1}) \approx \mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{u}'(x_n) + \frac{h^2}{2}\mathbf{u}''(x_n),$$

and in general,

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{u}'(x_n) + \cdots + \frac{h^p}{p}\mathbf{u}^{(p)}(x_n),$$

where $\mathbf{u}^{(k)}$ denotes the derivative of order k of the local solution $\mathbf{u}$. In principle this method is straightforward but the difficulty of obtaining and computing derivatives of high order of $\mathbf{u}$ by recursive derivation of the right hand side of the system of ordinary differential equations has restricted the popularity of the Taylor series scheme for solving numerically differential equations. For example, for the autonomous scalar case $y' = f(y)$,

$$\begin{aligned} y'' &= f'(y)\, f(y), \\ y''' &= f''(y)\, f(y)^2 + f'(y)\, f'(y)\, f(y), \quad etc. \end{aligned}$$

and the algebra becomes more cumbersome for systems. A second issue from the computational efficiency point of view is that the cost per step of a high order Taylor series method is very expensive in terms of number of evaluations of $\mathbf{f}$ and some of their derivatives.

2.2 Runge-Kutta methods

The class of the Runge-Kutta methods is the most important class of one-step methods for the numerical solution of ordinary differential equations. They were introduced by Runge and Kutta a century ago although their major development has ocurred only since 1960, following the work by Butcher. These methods can be well motivated by using quadrature rules to approximate the integral form for the system satisfied by the local solution at $x = x_n$,

$$\mathbf{u}(x_n + h) = \mathbf{u}(x_n) + \int_{x_n}^{x_n+h} \mathbf{f}(x, \mathbf{u}(x))dx. \tag{6}$$

For example, the Euler method is obtained if we use the rectangle quadrature rule based at $x = x_n$, and the implicit Euler methods arises if we use the rectangle quadrature rule based at $x = x_{n+1}$. By using the trapezoidal quadrature rule we obtain the implicit method

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{h}{2}(\mathbf{f}(x_n, \mathbf{y}_n) + \mathbf{f}(x_{n+1}, \mathbf{y}_{n+1})), \tag{7}$$

that is called the trapezoidal formula. In general, when the integral in equation (6) is approximated by a quadrature rule of order $p+1$

$$\int_{x_n}^{x_n+h} \phi(t)dt = h \sum_{i=1}^{s} b_i \phi(x_n + c_i h) + O(h^{p+1}), \quad (h \to 0) \tag{8}$$

we obtain a formula

$$\mathbf{u}(x_n + h) = \mathbf{y}_n + h \sum_{i=1}^{s} b_i \mathbf{f}(x_n + c_i h, \mathbf{u}(x_n + c_i h)) + O(h^{p+1}), \tag{9}$$

that in principle does not define a numerical method because requires unknown values $\mathbf{u}(x_n + c_i h), i = 1, \ldots, s$,(if $c_i \neq 0$), of the local solution $\mathbf{u}$. However, the identity

$$\mathbf{u}(x_n + c_i h) = \mathbf{u}(x_n) + \int_{x_n}^{x_n+c_i h} \mathbf{f}(x, \mathbf{u}(x))dx \tag{10}$$

suggests to use again a quadrature rule for obtaining approximations $\mathbf{Y}_i$ to $\mathbf{u}(x_n + c_i h)$, $i = 1, \ldots, s$. The key for this procedure is that the quadrature rule used to obtain the approximations $\mathbf{Y}_i, i = 1, \ldots, s$, does not need to be of the same order that the formula (8) to get the order $p + 1$ in (9). With this procedure it is possible to obtain some of the low order RK methods, as for example, the method of Heun or improved Euler method

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{h}{2}(\mathbf{f}(x_n, \mathbf{y}_n) + \mathbf{f}(x_n + h, \mathbf{y}_n + h\mathbf{f}(x_n, \mathbf{y}_n)) \tag{11}$$

or the midpoint rule or modified Euler method

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{f}(x_n + \frac{1}{2}h, \mathbf{y}_n + \frac{1}{2}h\mathbf{f}(x_n, \mathbf{y}_n)). \tag{12}$$

All the explicit formulas we have obtained above have the following structure

$$\begin{aligned} \mathbf{y}_{n+1} &= \mathbf{y}_n + h\sum_{i=1}^{s} b_i \mathbf{f}(x_n + c_i h, \mathbf{Y}_i), \\ \mathbf{Y}_i &= \mathbf{y}_n + h\sum_{j=1}^{i-1} a_{ij}\mathbf{f}(x_n + c_i h, \mathbf{Y}_j), \quad i = 1, \ldots, s, \end{aligned} \tag{13}$$

that defines a general explicit Runge-Kutta method of s stages. The quantities $\mathbf{Y}_i, i = 1, \ldots, s$, are called inner stages of the method and the method is completely defined by the $b_i, c_i, i = 1, \ldots, s$, and $a_{ij}, i = 1, \ldots, s, j = 1, \ldots, i-1$. The parameters $c_i, i = 1, \ldots, s$, are also called **abscises** of the method. It is also classical the notation

$$\begin{aligned} \mathbf{y}_{n+1} &= \mathbf{y}_n + h\sum_{i=1}^{s} b_i \mathbf{k}_i, \\ \mathbf{k}_i &= \mathbf{f}(x_n + c_i h, \mathbf{y}_n + h\sum_{j=1}^{i-1} a_{ij}\mathbf{k}_j), \quad i = 1, \ldots, s, \end{aligned}$$

for a explicit RK method of s stages. The term explicit is applied to this kind of formula because each $\mathbf{Y}_i, i = 1, \ldots, s$, depends only on the previously calculated $\mathbf{Y}_j, j = 1, \ldots, i-1$ rather than all the other $\mathbf{Y}$ values. Explicit Runge-Kutta methods are very easy to code and they only require for each step s evaluations of the function $\mathbf{f}$.

It is usual to display the coefficients of a Runge-Kutta method in the form of the following array

$$\begin{array}{c|cccc} c_1 & 0 & & & \\ c_2 & a_{21} & 0 & & \\ \vdots & \vdots & \vdots & \ddots & \\ c_s & a_{s1} & a_{s2} & \cdots & 0 \\ \hline & b_1 & b_2 & \cdots & b_s \end{array} \tag{14}$$

called Butcher's array of the method.

An explicit Runge-Kutta method can be written in the form

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h_n \mathbf{\Phi}(x_n, \mathbf{y}_n, h_n), \quad \mathbf{y}_0 \text{ given} \tag{15}$$

with the increment function

$$\mathbf{\Phi}(x, \mathbf{y}, h) := h \sum_{i=1}^{s} b_i \mathbf{f}(x_n + c_i h, \mathbf{Y}_i),$$

where the dependence of $\mathbf{\Phi}$ on $\mathbf{y}$ is through the recursive formulas that define the inner stages $\mathbf{Y}_i, i = 1, \ldots, s$ of the method. If $\mathbf{f}(x, \mathbf{y})$ satisfy a Lipschitz conditions with respect to $\mathbf{y}$ with Lipschitz constant L it is easy to check that $\mathbf{\Phi}$ also satisfied a Lipschitz condition with respect to $\mathbf{y}$ with Lipschitz constant

$$\tilde{L} = L \left(\sum_i |b_i| + hL \sum_{i,j} |b_i a_{ij}| + h^2 L^2 \sum_{i,j,k} |b_i a_{ij} a_{jk}| + \cdots \right)$$

2.3 General RK methods

General Runge-Kutta methods of s stages are defined by the formulas

$$\begin{aligned} \mathbf{y}_{n+1} &= \mathbf{y}_n + h \sum_{i=1}^{s} b_i \mathbf{f}(x_n + c_i h, \mathbf{Y}_i), \qquad (16) \\ \mathbf{Y}_i &= \mathbf{y}_n + h \sum_{j=1}^{s} a_{ij} \mathbf{f}(x_n + c_i h, \mathbf{Y}_j), \quad i = 1, \ldots, s. \end{aligned}$$

The RK method depend on the parameters $b_i, c_i, a_{ij}, i, j = 1, \ldots, s$, and the Butcher array of the method is now

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array} = \begin{array}{c|c} \mathbf{c} & A \\ \hline & \mathbf{b}^T \end{array}, \qquad (17)$$

where we have introduced for a more compact notation the vector of weights $\mathbf{b}^T = [b_1, \ldots, b_s]$, the vector of abscises $\mathbf{c} = [c_1, \ldots, c_s]$, and the matrix $s \times s$ $A = (a_{ij})_{i,j=1}^{s}$. Generally, the formulas (16) correspond to an implicit method because the equations for the inner stages form a nonlinear system of $s \times d$ equations for the $s \times d$ components of the vectors $\mathbf{Y}_i, i = 1, \ldots, s$. Some decoupling of the equations is obtained if $a_{ij} = 0$ when $j > i$ (i.e. A is lower triangular) because then the system is reduced to s different nonlinear systems one for each vector $\mathbf{Y}_i, i = 1, \ldots, s$. If A is strictly lower triangular then the RK method is explicit.

For implicit RK methods the existence of the numerical solution $\{\mathbf{y}_n\}_{n=0}^{N_h}$ is not trivial . Typically, a standard argument of fixed point iteration can prove that the nonlinear system

$$\mathbf{Y}_i = \mathbf{y}_n + h \sum_{j=1}^{s} a_{ij} \mathbf{f}(x_n + c_i h, \mathbf{Y}_j), \quad i = 1, \ldots, s. \qquad (18)$$

has a unique solution if the step size h is enough small, more precisely if

$$hL \max_{i=1,\ldots,s} \sum_{j=1}^{s} |a_{ij}| < 1, \tag{19}$$

where L is the Lipschitz constant of the function $\mathbf{f}$ with respect to $\mathbf{y}$.

Implicit RK methods are necessary in some situations. For example, important classes of problems in the applications are stiff, and for them stability considerations preclude the use of explicit RK methods. Moreover, these problems have a very large Lipschitz constant L and for them the condition on the step size (19) is too much demanding. Therefore, the nonlinear equations of the method for each step must be solved with some variant of the Newton's method.

2.4 Theory of convergence for one-step methods

We illustrate the main concepts with general explicit one-step methods of the form

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h_n \mathbf{\Phi}(x_n, \mathbf{y}_n, h_n), \quad \mathbf{y}_0 \text{ dado} . \tag{20}$$

Here $\mathbf{\Phi}$ is a continuous function of its arguments such that

$$\|\mathbf{\Phi}(x_n, \mathbf{y}_n, h_n) - \mathbf{\Phi}(x_n, \mathbf{y}_n^*, h_n)\| \leq \tilde{L} \|\mathbf{y}_n - \mathbf{y}_n^*\|. \tag{21}$$

Also we assume that $\mathbf{\Phi} \equiv 0$ when the function $\mathbf{f} \equiv 0$.

Given an initial approximation $\mathbf{y}_0 \approx \mathbf{A}$ let $\{\mathbf{y}_n\}_{n=0}^N$ be the numerical solution computed with the method (15) stepping from $x_0 = a$ to $x_N = b$ with successive step sizes $h_0, h_1, \ldots, h_{N-1}$. We introduce the global errors in the numerical solution as the differences

$$\mathbf{e}_n := \mathbf{y}(x_n) - \mathbf{y}_n, \quad n = 0, \ldots, N, \tag{22}$$

between the values of the exact solution $\mathbf{y}(x)$ at the nodes $x_n = x_0 + h_0 + \cdots + h_{n-1}, n = 0, \ldots, N$, and the corresponding numerical approximations given by the method. We define the magnitude of these errors by the quantity

$$\max_{0 \leq n \leq N} \|\mathbf{y}(x_n) - \mathbf{y}_n\|, \tag{23}$$

where $\|\cdot\|$ is a norm in $\mathbb{R}^d$, and we introduce the discretization parameter $h := \max_{0 \leq n \leq N-1} h_n$, representing the diameter of the discrete grid. We consider successive numerical integrations of the problem each time on finer discrete grids

$$a = x_0^h < x_1^h < \cdots < x_{N_h}^h = b, \quad h := \max_{1 \leq i \leq N_h} (x_i^h - x_{i-1}^h)$$

with $\mathbf{y}_0^h \to \mathbf{A}$ and $N_h \to \infty$, $h \to 0$.

Definition 2.1 *The method (15) is said to be convergent if for each initial value problem (1)*

$$\lim_{h \to 0^+} \max_{0 \leq n \leq N_h} \|\mathbf{y}(x_n) - \mathbf{y}_n\| = 0. \tag{24}$$

when $\mathbf{y}_0(h) \to \mathbf{y}(x_0)$. The method is said to be convergent of order p if, assuming that $\mathbf{y}_0(h) \to \mathbf{y}(x_0)$, this is the greater integer such that

$$\max_{0 \leq n \leq N_h} \|\mathbf{y}(x_n) - \mathbf{y}_n\| = O(h^p), \quad (h \to 0^+) \tag{25}$$

for all problems (1) with $\mathbf{f} \in C^p$, .

Given a numerical approximation $\mathbf{y}_n$ to $\mathbf{y}(x_n)$, the difference between the local solution at $x = x_{n+1}$ and the numerical approximation $\mathbf{y}_{n+1}$

$$\mathbf{el}_{n+1} := \mathbf{u}(x_{n+1}) - \mathbf{y}_{n+1} = \mathbf{u}(x_{n+1}) - \mathbf{y}_n - h_n \mathbf{\Phi}(x_n, \mathbf{y}_n, h_n) \tag{26}$$

is called the local error of the method at $x = x_{n+1}$. The method is of order p if the local errors at each grid point are of order $p+1$, i.e.

$$\|\mathbf{el}_{n+1}\| = O(h^{p+1}), \quad (h \to 0^+), \qquad n = 0, 1, \ldots, N_h - 1, \tag{27}$$

for all initial value problem (1) with $\mathbf{f} \in C^p$. The local truncation error $\mathbf{t}_{n+1}$ at $x = x_{n+1}$, $n = 0, \ldots, N_h - 1$, is by definition the residual that we obtain when the exact values of the solution of (1) are inserted in the difference equation (15)

$$\mathbf{t}_{n+1} := \mathbf{y}(x_{n+1}) - \mathbf{y}(x_n) - h_n \mathbf{\Phi}(x_n, \mathbf{y}(x_n), h_n). \tag{28}$$

We also put $\mathbf{t}_0 := \mathbf{y}(x_0) - \mathbf{y}_0$.

Definition 2.2 *The method (15) is said to be consistent if for all initial value problem (1)*

$$\lim_{h \to 0^+} \sup_{0 \le n \le N_h - 1} \|\frac{\mathbf{t}_{n+1}}{h_n}\| = 0. \tag{29}$$

The method is said to be consistent of order p if

$$\sup_{0 \le n \le N_h - 1} \|\frac{\mathbf{t}_{n+1}}{h_n}\| = O(h^p) \quad (h \to 0^+). \tag{30}$$

when $\mathbf{f} \in C^p$.

We notice that the sequences $\{\mathbf{y}_n\}_{n=0}^{N_h}$ and $\{\mathbf{y}(x_n)\}_{n=0}^{N_h}$ satisfy respectively the difference equations

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h_n \mathbf{\Phi}(x_n, \mathbf{y}_n, h_n), \quad n = 0, \ldots, N_h - 1 \tag{31}$$

$$\mathbf{z}_{n+1} = \mathbf{z}_n + h_n \mathbf{\Phi}(x_n, \mathbf{z}_n, h_n) + \mathbf{d}_{n+1}, \quad n = 0, \ldots, N_h - 1, \tag{32}$$

with starting values $\mathbf{y}_0 \approx \mathbf{A}$ and $\mathbf{z}_0 = \mathbf{y}_0 + \mathbf{d}_0$. To compare these two solutions we introduce the important concept of stability

Definition 2.3 *The method (15) is said stable if for all initial value problem (1) the solutions $\{\mathbf{y}_n\}_{n=0}^{N_h}$ and $\{\mathbf{z}_n\}_{n=0}^{N_h}$ of respectively (31) and (32) satisfy*

$$\max_{0 \le n \le N_h} \|\mathbf{y}_n - \mathbf{z}_n\| \le S \sum_{j=0}^{N_h} \|\mathbf{d}_j\|, \tag{33}$$

where S is a positive constant that only depend on the problem (1) and does not depend on the discretization parameter h. S is called the stability constant of the method.

The condition (21) it is sufficient to prove the stability property of the method (15). If we subtract the difference equation (31) from the difference equation (32), and if we add the resulting equation from $n = 0$ to $n - 1$, it results

$$\mathbf{z}_n - \mathbf{y}_n = \sum_{j=0}^{n-1} h_j [\mathbf{\Phi}(x_j, \mathbf{z}_j, h_j) - \mathbf{\Phi}(x_j, \mathbf{y}_j, h_j)] + \sum_{j=0}^{n} \mathbf{d}_j. \tag{34}$$

Taking norms and using the condition (21) we have

$$\|\mathbf{z}_n - \mathbf{y}_n\| \leq \tilde{L} \sum_{j=0}^{n-1} h_j \|\mathbf{z}_j - \mathbf{y}_j\| + \|\sum_{j=0}^{n} \mathbf{d}_j\|, \quad n = 0, 1, \ldots, N_h.$$

Then we can derive by induction

$$\|\mathbf{z}_n - \mathbf{y}_n\| \leq \left(\prod_{j=0}^{n-1} (1 + \tilde{L} h_j) \right) \max_{0 \leq k \leq n} \|\sum_{j=0}^{k} \mathbf{d}_j\|, \quad n = 0, \ldots, N_h.$$

Finally, using the inequality $1 + h_j \tilde{L} \leq \exp(h_j \tilde{L})$ at each factor in the product in the last inequality, we obtain

$$\max_{0 \leq n \leq N_h} \|\mathbf{z}_n - \mathbf{y}_n\| \leq \exp(\tilde{L}(b-a)) \max_{0 \leq n \leq N_h} \|\sum_{j=0}^{n} \mathbf{d}_j\|. \tag{35}$$

Convergence of the method is now established by using the stability estimate with the numerical solution $\{\mathbf{y}_n\}_{n=0}^{N}$ of (31) and the exact solution $\{\mathbf{y}(x_n)\}_{n=0}^{N_h}$, that satisfy (32) with perturbations given by the local truncation errors. One-step implicit methods requires minimal changes in the previous arguments. These methods are formulated in the form

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h_n \mathbf{\Psi}(x_n, \mathbf{y}_n, \mathbf{y}_{n+1}, h_n), \quad \mathbf{y}_0 \text{ given}, \tag{36}$$

with $\mathbf{\Psi}$ satisfying a Lipschitz condition with respect the arguments $\mathbf{y}_n$ and $\mathbf{y}_{n+1}$. However, if the step size h_n is enough small to be applicable the theorem of the implicit function, we can assume that $\mathbf{y}_{n+1}$ is given again in the form (15).

3 Obtaining RK methods

Composition of quadrature rules, as was introduced at the beginning of this section, is not the most efficient way to construct Runge-Kutta methods because for a given order of the method, the number of stages of the method can be higher than the minimum necessary. The natural approach for constructing RK methods of order p is to impose on the coefficients of the Butcher's array of the method the conditions derived of asking that

$$\|\mathbf{u}(x_n + h) - \mathbf{y}_{n+1}\| = O(h^{p+1}), \quad (h \to 0). \tag{37}$$

The condition (37) amounts to match the Taylor series expansions in powers of h of $\mathbf{u}(x_n + h)$ and $\mathbf{y}_{n+1}(h)$ respectively through all the terms until the term in h^p for a function $\mathbf{f}$ enough smooth. The algebra involved is enormous and could only be simplified after the introduction by Butcher of his algebraic theory of order for RK methods. For example, for the equation $y' = f(x, y)$, we have retaining until third order terms

$$\begin{aligned} u(x_n + h) = y_n + hf + \frac{1}{2}h^2(f_x + ff_y) \\ + \frac{1}{6}h^3(f_y(f_x + ff_y) + (f_{xx} + 2ff_{xy} + f^2 f_{yy})) + O(h^4), \end{aligned}$$

where $f_x, f_y, f_{xx}, \ldots$, denote respectively partial derivatives of f and all the functions are evaluated at $(x_n, \mathbf{y}_n)$. Similarly, the numerical solution $y_{n+1}(h)$ obtained with an explicit RK method of three stages has an expansion in powers of h of the form

$$\begin{aligned} y_{n+1} &= y_n + h(b_1+b_2+b_3)f + h^2(b_2c_2+b_3c_3)(f_x+ff_y) \\ &+ \frac{1}{2}h^3[2b_3c_2a_{32}(f_x+ff_y)f_y + (b_2c_2^2+b_3c_3^2)(f_{xx}+2ff_{xy}+f^2f_{yy})] \\ &+ O(h^4). \end{aligned}$$

Explicit RK methods of one-stage have coefficients $b_2 = b_3 = a_{21} = a_{32} = 0$, and then it is only possible to obtain first order methods with $b_1 = 1$. It is easy to check the order conditions

$$b_1 + b_2 = 1, \quad b_2c_2 = 1/2,$$

that determines a one-parameter family of explicit RK methods of 2 stages and order two; for example, the improved Euler method and the modified Euler method are methods in this family. Finally, RK methods of three stages and order three are determined by the order conditions

$$\begin{aligned} b_1 + b_2 + b_3 &= 1 \\ b_2c_2 + b_3c_3 &= 1/2 \\ b_2c_2^2 + b_3c_3^2 &= 1/3 \\ b_3c_2a_{32} &= 1/6. \end{aligned}$$

Some methods in this class are the method of Heun and the method of Kutta with Butcher's arrays given respectively by

$$\begin{array}{c|ccc} 0 & & & \\ 1/3 & 1/3 & & \\ 2/3 & 0 & 2/3 & \\ \hline & 1/4 & 0 & 3/4 \end{array} \qquad \begin{array}{c|ccc} 0 & & & \\ 1/2 & 1/2 & & \\ 1 & -1 & 2 & \\ \hline & 1/6 & 2/3 & 1/6 \end{array} \tag{38}$$

To get explicit RK methods of order 4 it is necessary consider methods with four stages, as for example the classical Runge-Kutta method and the methods of 3/8, with Butcher arrays respectively

$$\begin{array}{c|cccc} 0 & & & & \\ 1/2 & 1/2 & & & \\ 1/2 & 0 & 1/2 & & \\ 1 & 0 & 0 & 1 & \\ \hline & 1/6 & 2/6 & 2/6 & 1/6 \end{array} \quad , \quad \begin{array}{c|cccc} 0 & & & & \\ 1/3 & 1/3 & & & \\ 2/3 & -1/3 & 1 & & \\ 1 & 1 & -1 & 1 & \\ \hline & 1/8 & 3/8 & 3/8 & 1/8 \end{array} \quad . \tag{39}$$

3.1 The algebraic theory of order for RK methods

Butcher developed at the beginning of the sixties an algebraic theory to study systematically the order conditions of a general RK method or s stages. In this section we only want to give a concise description of this theory. To start we

will assume without loss of generality that the system of ordinary differential equations is autonomous and given by

$$\mathbf{y}' = \mathbf{f}(\mathbf{y}), \quad a \leq x \leq b, \tag{40}$$

and that we have a general RK method with Butcher's array

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array} \tag{41}$$

such that the coefficients satisfy

$$c_i = \sum_{j=1}^{s} a_{ij}, \quad i = 1, \ldots, s. \tag{42}$$

We denote the component i-th component of the vector $\mathbf{f}$ by f^i and the partial derivative with respect to the j-th component of vector $\mathbf{y}$, y^j, by a colon followed by a subscript j. For example,

$$\frac{\partial f^i}{\partial y^j} = f^i_{,j}, \quad \frac{\partial^2 f^i}{\partial y^j \partial y^k} = f^i_{,j,k}. \tag{43}$$

Moreover we will use the convention that if an index is repeated in an expression, the expression is to be summed over all values of the index. The context will make clear in what range is varying the index. For example, in the Taylor series expansion of $\mathbf{u}(x_n + h)$ at $x = x_n$ we will use

$$(\mathbf{u}')^i(x_n) = f^i(\mathbf{y}_n), \quad i = 1, \ldots, d, \tag{44}$$

$$(\mathbf{u}'')^i(x_n) = \sum_{j=1}^{d} f^i_{,j}(\mathbf{y}_n) f^j(\mathbf{y}_n) = f^i_{,j}(\mathbf{y}_n) f^j(\mathbf{y}_n), \quad i = 1, \ldots, d. \tag{45}$$

and, for the third and the fourth derivative of $\mathbf{u}$,

$$(\mathbf{u}''')^i(x_n) = f^i_{,j,k} f^j f^k + f^i_{,j} f^j_{,k} f^k, \tag{46}$$

$$\begin{aligned} (\mathbf{u}^{(4)})^i(x_n) &= f^i_{,j,k,l} f^j f^k f^l + f^i_{,j,k} f^j_{,l} f^l f^k + f^i_{,j,k} f^j f^k_{,l} f^l \\ &+ f^i_{,j,l} f^l f^j_{,k} f^k + f^i_{,j} f^j_{,k,l} f^k f^l + f^i_{,j} f^j_{,k} f^k_{,l} f^l, \end{aligned} \tag{47}$$

where we have already assumed the sum convention over repeated index and we have omitted the argument $\mathbf{y}_n$ on all the components of $\mathbf{f}$ and its derivatives. It is clear that each of the derivatives of $\mathbf{u}$ is sum of different terms, each one representing a sum over a set of index $j, k, l, \ldots$ in which each summand is a product of partial derivatives of components of $\mathbf{f}$. Each of these terms is called an elementary differential of the function $\mathbf{f}$. The other characteristic ingredient of the algebraic theory of Butcher is the association of each elementary differential

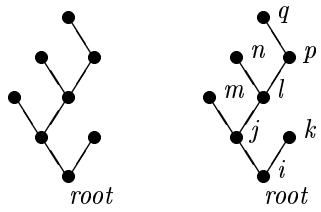

Figure 1: Rooted tree τ and labelled rooted tree $\lambda\tau$

of $\mathbf{f}$ to a rooted tree. A rooted tree is just a graph as the one at the left in the figure In terms of a monotone labelling $i < j < k < l < \ldots$ of that tree, for example, the one at the right of the figure, we can associate the elementary differential given by

$$(\mathbf{F})^i(\lambda\tau) = \sum_{j,k,l,m,n,p,q=1}^{d} f^i_{,j,k} f^k f^j_{,l,m} f^m f^l_{,p,n} f^n f^p_{,q} f^q.$$

in the following way. The index i at the root represents the i-th component of the function $\mathbf{f}$, which is differentiated with respect to the variables with the indexes j and k because these are the indexes of the nodes branching from the root. Node k represents the k-component of $\mathbf{f}$ and node j represents the j component of $\mathbf{f}$ again differentiated with respect to the variables with indexes l and m because these are the labels of the nodes branching from node j. Following in this way node m represents f^m, node l represents $f^l_{,p,n}$, node n represents f^n, node p represents $f^p_{,q}$ and finally node q represents f^q. With this convention we can write down the following expansion in powers of the step size h for $\mathbf{u}^{(q)}(x_n + h)$

$$\mathbf{u}^{(q)}(x_n + h) = \sum_{\tau \in T_q} \alpha(\tau)\mathbf{F}(\tau)(\mathbf{y}_n) \tag{48}$$

Here T_q is the set of all the rooted trees with order (number of nodes) q, and $\alpha(\tau)$ represents the number of different monotone labelling for the rooted tree τ.

We consider now the expansion in powers of the step size h of the numerical solution $\mathbf{y}_{n+1} = \mathbf{y}_{n+1}(h)$, given by

$$\mathbf{y}_{n+1}(h) = \mathbf{y}_n + h\sum_{i=1}^{s} b_i \mathbf{f}(x_n + c_i h, \mathbf{Y}_i) \tag{49}$$

$$\mathbf{Y}_i(h) = \mathbf{y}_n + h\sum_{j=1}^{s} a_{ij} \mathbf{f}(x_n + c_j h, \mathbf{Y}_j), \quad i = 1, \ldots, s. \tag{50}$$

For each monotone labelling $\lambda\tau$ of a rooted tree τ of order q with labels $j_1 < j_2 < \ldots < j_q$ we introduce the coefficients

$$\Psi^{j_1}(\lambda\tau) = \sum_{j_2,\ldots,j_q=1}^{s} a_{\lambda\tau(j_2),j_2} \cdots a_{\lambda\tau(j_q),j_q}, \quad j_1 = 1, 2, \ldots, s$$

and we put $\mathbf{\Psi}(\tau) = [\Psi^1(\lambda\tau), \ldots, \Psi^s(\lambda\tau)]$. It is easy to check that $\mathbf{\Psi}(\tau)$ does not depend on the monotone labelling $\lambda\tau$ chosen for the rooted tree τ. For example,

for the monotone labelling of the rooted tree τ in the figure (1) we have

$$\Psi^i(\tau) = \sum_{j,k,l,m,n,p,q=1}^{s} a_{ij}a_{ik}a_{jm}a_{jl}a_{ln}a_{lp}a_{pq} = \sum_{j,l,p} c_i a_{ij} c_j a_{jl} c_l a_{lp} c_p, \qquad (51)$$

where in the second sum we have made use of the relations (42) to contract the indexes in the first sum corresponding to final nodes in the tree. The density $\gamma(\tau)$ of a rooted tree τ is defined recursively as follows: first we put

$$\tau = [\tau_1, \tau_2, \ldots, \tau_M] \qquad (52)$$

if the rooted tree τ is obtained by connecting the roots of the rooted trees $\tau_1, \ldots, \tau_M$ to a new node that is going to become the root of the rooted tree τ. Then

$$\gamma(\tau) := \rho(\tau)\gamma(\tau_1)\gamma(\tau_2)\ldots\gamma(\tau_M),$$

with $\rho(\tau)$ representing the order of the rooted tree τ. When τ is the rooted tree of order one then $\gamma(\tau) = 1$.

With this definitions we have that the q-derivatives of $\mathbf{y}_{n+1}(h)$ and $\mathbf{Y}_i(h), i = 1, 2, \ldots, s$ at $h = 0$ are given respectively by the expressions

$$\begin{aligned} (\mathbf{Y}_i)^{(q)}\Big|_{h=0} &= \sum_{\tau\in T_q} \alpha(\tau)\gamma(\tau) \sum_{j=1}^{s} a_{ij}\Psi^j(\tau)\mathbf{F}(\tau)(\mathbf{y}_n) \\ (\mathbf{y}_{n+1})^{(q)}\Big|_{h=0} &= \sum_{\tau\in T_q} \alpha(\tau)\gamma(\tau) \sum_{i=1}^{s} b_i\Psi^i(\tau)\mathbf{F}(\tau)(\mathbf{y}_n) \end{aligned}$$

Therefore we have

Theorem 3.1 *A general Runge-Kutta method of s stages has order p if and only if*

$$\sum_{i=1}^{s} b_i\Psi^i(\tau) = \frac{1}{\gamma(\tau)}, \qquad (53)$$

holds for all the rooted trees τ with order less than or equal to p, and does not hold for at least a rooted tree of order $p+1$.

We illustrate the theorem by writing down the table with all the order conditions for order $p \leq 4$. The rooted trees are denoted in terms of the rooted tree τ_0 of order one and the recursive definition already introduced

$$\tau = [\tau_1^{n_1}, \tau_2^{n_2}, \ldots, \tau_M^{n_M}] := [\tau_1, \overset{n_1}{\ldots}, \tau_1, \tau_2, \overset{n_2}{\ldots}, \tau_2, \ldots, \tau_M, \overset{n_M}{\ldots}, \tau_M]$$

We finish with the asymptotic expansion in powers of the step size h of the local truncation error of a general RK method of s stages. If $\mathbf{f}$ is continuously differentiable with respect to all the orders until the order $M+1$ then when $(h \to 0)$

$$\mathbf{y}(x_n + h) - \mathbf{y}_{n+1} = \sum_{m=p+1}^{M} \left[\sum_{\tau\in T_m} \alpha(\tau)e(\tau)\mathbf{F}(\tau)(\mathbf{y}(x_n))\right] \frac{h^m}{m!} + O(h^{M+1}). \qquad (54)$$

τ	$\rho(\tau)$	$\mathbf{b}^T\Psi(\tau)$	$=$	$\sum$	$=$	$1/\gamma(\tau)$
τ_0	1	$\mathbf{b}^T\mathbf{e}$	$=$	$\sum_i b_i$	$=$	1
$[\tau_0]$	2	$\mathbf{b}^T\mathbf{c}$	$=$	$\sum_i b_i c_i$	$=$	$1/2$
$[\tau_0^2]$	3	$\mathbf{b}^T\mathbf{c}^2$	$=$	$\sum_i b_i c_i^2$	$=$	$1/3$
$[[\tau_0]]$		$\mathbf{b}^T A\mathbf{c}$	$=$	$\sum_{ij} b_i a_{ij} c_j$	$=$	$1/6$
$[\tau_0^3]$	4	$\mathbf{b}^T\mathbf{c}^3$	$=$	$\sum_i b_i c_i^3$	$=$	$1/4$
$[\tau_0[\tau_0]]$		$(\mathbf{b}.\mathbf{c})^T A\mathbf{c}$	$=$	$\sum_{ij} b_i c_i a_{ij} c_j$	$=$	$1/8$
$[[\tau_0^2]]$		$\mathbf{b}^T A\mathbf{c}^2$	$=$	$\sum_{ij} b_i a_{ij} c_j^2$	$=$	$1/12$
$[[[\tau_0]]]$		$\mathbf{b}^T A^2\mathbf{c}$	$=$	$\sum_{ijk} b_i a_{ij} a_{jk} c_k$	$=$	$1/24$

Table 1: Order conditions for $p \leq 4$

where

$$e(\tau) = 1 - \gamma(\tau) \sum_{i=1}^{s} b_i \Psi^i(\tau).$$

The factor of h^{p+1} in the right hand side of (54), as a function of x,

$$d_{p+1}(x) := \frac{1}{(p+1)!} \sum_{\tau \in T_{p+1}} \alpha(\tau) e(\tau) \mathbf{F}(\tau)(\mathbf{y}(x))$$

is called the principal function of the local truncation error of the method.

3.2 High order RK methods

It is easy to prove that the highest order for a explicit RK method of s stages is $p = s$. However, using his algebraic theory Butcher's proved the following barriers for the highest order p that an explicit RK method of s stages can attain.

1. For $p \geq 5$ no explicit RK method exists of order p with $s = p$ stages.
2. For $p \geq 7$ no explicit RK method exists of order p with $s = p + 1$ stages.
3. For $p \geq 8$ no explicit RK method exists of order p with $s = p + 2$ stages.

Implicit RK methods can attain a higher order that of the an explicit RK method with the same number of stages. For example, the method

$$\begin{array}{c|c} 1/2 & 1/2 \\ \hline & 1 \end{array}$$

has order 2 with only one stage. To find implicit RK methods of two stages, the order conditions for order three are

$$\begin{aligned} &b_1 + b_2 = 1, \\ &b_1 c_1 + b_2 c_2 = 1/2 \\ &b_1 c_1^2 + b_2 c_2^2 = 1/3 \\ &b_1(a_{11}c_1 + a_{12}c_2) + b_2(a_{21}c_1 + a_{22}c_2) = 1/6. \end{aligned}$$

Because the theory of Gaussian quadrature, the three first equations imply the orthogonality condition

$$\int_0^1 (x - c_1)(x - c_2)dx = 0,$$

from which we can derive the relation between the parameters c_1 and c_2,

$$c_2 = \frac{3c_1 - 2}{6c_1 - 3}.$$

Also, from the two first equations we get

$$b_1 = \frac{c_2 - 1/2}{c_2 - c_1}, \quad b_2 = \frac{c_1 - 1/2}{c_1 - c_2}.$$

Finally, if in the last equation we put $a_{21} = c_2 - a_{22}$ and $a_{11} = c_1 - a_{12}$, we obtain

$$a_{22} = \frac{1/6 - b_1 a_{12}(c_2 - c_1) - c_1/2}{b_2(c_2 - c_1)}.$$

Therefore we have obtained a two-parameter family of RK methods with two stages and order 3. In particular, with $a_{12} = 0$ we get a one-parameter family of semiimplicit RK method.

There is only one RK method of two stages and order 4, with Butcher array given by

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

It is called a Gaussian RK method because the quadrature rule with nodes given by the abscises of the vector $\mathbf{c}$ and weights given by the elements of the vector $\mathbf{b}$ is the Gaussian quadrature rule of order 4. This kind of Gaussian RK methods can be constructed for each s (the number of stages) with order $2s$. The same

idea can be used to define other families of methods. For example, the Radau methods of s stages have order $2s-1$ and they fixed or $c_1 = 0$ (Radau IA) or $c_s = 1$ (Radau IIA). The Lobatto methods of s stages have order $2s-2$ and they fixed $c_1 = 0$ and $c_s = 1$. There are three different families of methods depending on if the first row in the Butcher array of the method is null (Lobato IIIA), the last column in the Butcher array is null (Lobato IIIB) or if the last row of A in the Butcher array agree with the row of weights $\mathbf{b}$ (Lobato IIIC).

We finish introducing the important idea of collocation. We desire to approximate in the interval $[x_n, x_{n+1}]$ the local solution $\mathbf{u}$ defined by

$$\mathbf{u}' = \mathbf{f}(x, \mathbf{u}), \quad x_n \leq x \leq x_{n+1}, \quad \mathbf{u}(x_n) = \mathbf{y}_n \tag{55}$$

by a polynomial $\mathbf{p}(x)$ with coefficients in $I\!R^d$ and of degree $\leq s$. If s distinct $c_1, \ldots, c_s$ real numbers in $[0,1]$ are given the polynomial of collocation $\mathbf{p}(x)$ is the only polynomial of degree $\leq s$ such that satisfy the conditions

$$\mathbf{p}(x_n) = \mathbf{y}_n, \tag{56}$$

$$\mathbf{p}'(x_n + c_i h_n) = \mathbf{f}(x_n + c_i h_n, \mathbf{p}(x_n + c_i h)), \quad i = 1, \ldots, s \tag{57}$$

where $h_n = x_{n+1} - x_n$. The conditions (57) amount to impose that the polynomial $\mathbf{p}(x)$ holds the ordinary differential equation system at the abscises of collocation $x_n + c_i h_n, i = 1, \ldots, s$. The value

$$\mathbf{y}_{n+1} := \mathbf{p}(x_{n+1}) \tag{58}$$

can be taken as an approximation to $\mathbf{y}(x_{n+1})$. These methods are equivalent to implicit RK methods of order $p \geq s$.

4 Error control, variable step-size strategies and efficiency

4.1 How to change the step size?

An adaptive stepsize selection is central to an efficient numerical integration of ODEs. In principle, the stepsize should be selected with respect to prescribed accuracy requirements. Most of the codes for ODEs base their strategies for changing the stepsize in some way of control of the local error at each step of the numerical solution. In general, for explicit one-step methods of the form

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h_n \mathbf{\Phi}(x_n, \mathbf{y}_n, h),$$

the local error at $x = x_n$ is defined by

$$\mathbf{el}_{n+1} = \mathbf{u}(x_n + h_n) - \mathbf{y}_{n+1},$$

where the local solution $\mathbf{u}(x)$ is the solution of the problem $\mathbf{u}' = \mathbf{f}(x, \mathbf{u}), \mathbf{u}(x_n) = \mathbf{y}_n$. When $\mathbf{y}_n = \mathbf{y}(x_n)$, the local solution $\mathbf{u}(x)$ and the true solution $\mathbf{y}(x)$ are the same and then we get the local truncation error

$$\mathbf{t}_{n+1} = \mathbf{y}(x_n + h_n) - \mathbf{y}_{n+1}.$$

Generally, the local error and the local truncation error for a method of order p can be respectively expanded asymptotically as

$$\mathbf{el}_{n+1} = h_n^{p+1}\mathbf{\Psi}(x_n, \mathbf{y}_n) + O(h_n^{p+2}),$$

y

$$\mathbf{t}_{n+1} = h_n^{p+1}\mathbf{\Psi}(x_n, \mathbf{y}(x_n)) + O(h_n^{p+2}).$$

We assume the user have specified a norm $\|\cdot\|$ in which the error is to be mesasured and a tolerance $\tau > 0$. Two kinds of control on the local error (or local truncation error) are seen in modern codes. The criterion of error per step, EPS, accepts the step to x_{n+1} only when

$$\|\mathbf{el}_{n+1}\| \leq \tau.$$

The criterion of error per unit step, EPUS, accepts the step only when

$$\|\mathbf{el}_{n+1}\| \leq h_n\,\tau.$$

When a code takes a step from x_n to $x_n + h_n$, and estimate $\mathbf{est}_{n+1}$ is made of the local error incurred in the step:

$$\mathbf{est}_{n+1} \approx \mathbf{le}_{n+1} = \mathbf{u}(x_n + h_n) - \mathbf{y}_{n+1} = h_n^{p+1}\mathbf{\Psi}(x_n, \mathbf{y}_n) + O(h_n^{p+2}).$$

According to the expression for the local error, if we had taken the step from x_n with a step h^* to get a result $\mathbf{y}^*_{n+1}$, the local error would have been

$$\mathbf{u}(x_n + h^*) - \mathbf{y}^*_{n+1} = (h^*)^{p+1}\mathbf{\Psi}(x_n, \mathbf{y}_n) + O((h^*)^{p+2}).$$

If the estimated local error $\mathbf{le}_{n+1}$ exceeds the tolerance, we reject the step-size h_n and we estimate the optimal step size h^* that is predicted to pass the error test. When the control is EPS, we are interested in h^* such that

$$\tau = \|\mathbf{le}^*_{n+1}\| \approx (h^*/h_n)^{p+1}\|\mathbf{est}_{n+1}\|,$$

hence

$$h^* \approx h_n(\tau/\|\mathbf{est}_{n+1}\|)^{1/(p+1)}.$$

It is standard to use a fraction of the optimal step size, being the value of 0.9 representative of those seen in the codes. A scheme that is equivalent in codes of fixed order is to aim at a fraction β of the tolerance τ

$$h^* = h_n(\beta\tau/\|\mathbf{est}_{n+1}\|)^{1/(p+1)}.$$

If the step from x_n is a success, we want to predict what step size might be used for the next step. From

$$\mathbf{le}_{n+2} = h_{n+1}^{p+1}\mathbf{\Psi}(x_{n+1}, \mathbf{y}_{n+1}) + O(h_{n+1}^{p+2})$$

we can use that $\mathbf{\Psi}(x_{n+1}, \mathbf{y}_{n+1}) = \mathbf{\Psi}(x_n, \mathbf{y}_n) + O(h_n)$, to predict a suitable h_{n+1} as

$$h_{n+1} = h_n(\beta\tau/\|\mathbf{est}_{n+1}\|)^{1/(p+1)},$$

that agree with the formula used after an unsuccessful step.

4.2 Local extrapolation

Assume we have an error estimator of the local error at each step that is asumptotically correct, meaning that

$$\mathbf{est}_{n+1} = \mathbf{le}_{n+1} + O(h_n^{p+2}).$$

The idea of extrapolation is to define a new solution approximation by

$$\mathbf{y}^*_{n+1} = \mathbf{y}_{n+1} + \mathbf{est}_{n+1},$$

that represents a numerical result of order $p+1$, because

$$\mathbf{u}(x_n + h_n) - \mathbf{y}^*_{n+1} = O(h_n^{p+2}).$$

From another point of view, we take each step with two formulas, one of order p and the other or order $p+1$, and then estimate the error in the lower order result by comparison:

$$\mathbf{est}_{n+1} = \mathbf{y}^*_{n+1} - \mathbf{y}_{n+1}.$$

We talk then of a pair of formulas $(p, p+1)$. In this context, why not to advance the numerical solution with the more accurate result $\mathbf{y}^*_{n+1}$? Advancing the method with the higher order result is called **local extrapolation**, a procedure that it is adopted in quite a lot of codes based on explicit RK methods and in many of the Adams codes so.

4.3 Error estimation with RK methods: embedded RK pairs

A successful approach to the estimation of the local error in RK methods is to advance the numerical solution simultaneously with a pair of RK methods of orders p and $p+1$ respectively. The key idea is to construct this pair of RK methods in such a way that they have in commun the maximum number of stages. This idea was first introduced by Merson (1957), who considered the RK method with Butcher's array

$$\begin{array}{c|ccccc}
0 & & & & & \\
\frac{1}{3} & \frac{1}{3} & & & & \\
\frac{1}{3} & \frac{1}{6} & \frac{1}{6} & & & \\
\frac{1}{2} & \frac{1}{8} & 0 & \frac{3}{8} & & \\
1 & \frac{1}{2} & 0 & -\frac{3}{2} & 2 & \\
\hline
 & \frac{1}{6} & 0 & 0 & \frac{2}{3} & \frac{1}{6}
\end{array}$$

This is an explicit five stages RK method with order 4. Merson proposed that the principal term in the local error were estimated by

$$\mathbf{est} \approx h_n(-2\mathbf{k}_1 + 9\mathbf{k}_3 - 8\mathbf{k}_4 + \mathbf{k}_5)/30.$$

This is an asymptotically correct estimate of the local error of the RK method at x_n for linear systems of ordinary differential equations. When this estimate is added to the numerical approximation $\mathbf{y}_{n+1}$ we get an approximation of fifth order. However, for general systems the accuracy of this approximation degrades to order three.

Pairs of embedded RK methods are presented with the following modified Butcher's array

$\mathbf{c}$	A
	$\mathbf{b}^T$
	$\hat{\mathbf{b}}^T$
	$\mathbf{E}^T$

Here the RK method of order p is defined by the vector of abscises $\mathbf{c}$, the matrix $s \times s$, A, and the vector of weights $\mathbf{b}^T$. The RK method of order $p+1$ has the same vector of abscises $\mathbf{c}$ and the same matrix A but a different vector of weights $\hat{\mathbf{b}}^T$. The difference between the values $\mathbf{y}^{\star}_{n+1}$ y $\mathbf{y}_{n+1}$ computed respectively with the methods of order p and order $p+1$ is then an estimate of the local error en $\mathbf{y}^{*}_{n+1}$. The vector $\mathbf{E}^T$ is the vector $\hat{\mathbf{b}}^T - \mathbf{b}^T$. We associate the label RK p(p+1) to the pair of embedded RK methods if we use the result $\mathbf{y}^{*}_{n+1}$ of order p to advance the numerical solution and the label RK p+1(p) if local extrapolation it is used and the numerical solution is advanced with the value $\mathbf{y}_{n+1}$.

Sarafyan, Fehlberg, y England derived pairs of embedded RK methods of order $(4, 5)$ with the minimum number of stages, i.e. 6. Many higher order pairs have been constructed. In particular Fehlberg (1968,1969) has presented pairs with orders 4(5), 5(6), 7(8), 8(9), that were designed to be implemented in lower order mode and with optimized coefficients in the principal part of the local error (error-tuned methods). For example, the Runge-Kutta-Fehlberg 4(5) method is given by the modified Butcher's array

0						
$\frac{1}{4}$	$\frac{1}{4}$					
$\frac{3}{8}$	$\frac{3}{32}$	$\frac{9}{32}$				
$\frac{12}{13}$	$\frac{1932}{2197}$	$-\frac{7200}{2197}$	$\frac{7296}{2197}$			
1	$\frac{439}{216}$	-8	$\frac{3680}{513}$	$-\frac{845}{4104}$		
$\frac{1}{2}$	$-\frac{8}{27}$	2	$-\frac{3544}{2565}$	$\frac{1859}{4104}$	$-\frac{11}{40}$	
	$\frac{25}{216}$	0	$\frac{1408}{2565}$	$\frac{2197}{4104}$	$-\frac{1}{5}$	0
	$\frac{16}{135}$	0	$\frac{6656}{12825}$	$\frac{28561}{56430}$	$-\frac{9}{50}$	$\frac{2}{55}$
	$-\frac{1}{360}$	0	$-\frac{128}{4275}$	$-\frac{2197}{75240}$	$\frac{1}{50}$	$\frac{2}{55}$

The earliest RK pairs designed from the outset to operate in local extrapolation mode were presented by Dormand and Prince in 1980 and 1981. Of these, the

most efficient in their respective classes are the 7-stage RK 5(4) and the 13-stage RK 8(7), sometimes referenced as DOPRI5 and DOPRI8 respectively. A feature of DOPRI5 is the use of the FSAL (first same as last) technique, first introduced by Dormand y Prince, which in this case requires

$$\hat{b}_7 = 0, \quad a_{7j} = \hat{b}_j, \quad j = 1, \ldots, 6,$$

and means that the last row of the matrix A of the method agreed with the row of weights used to advance the numerical solution. This means that although the pair DOPRI5 has not a minimal number of stages, it operates at successful steps as it has six stages because the first stage in the next step agree with the last stage computed in the previous step. The modified Butcher's array of DOPRI5 is given by

$$\begin{array}{c|ccccccc}
0 & & & & & & & \\
\frac{1}{5} & \frac{1}{5} & & & & & & \\
\frac{3}{10} & \frac{3}{40} & \frac{9}{40} & & & & & \\
\frac{4}{5} & \frac{44}{45} & -\frac{56}{15} & \frac{32}{9} & & & & \\
\frac{8}{9} & \frac{19372}{6561} & -\frac{25360}{2187} & \frac{64448}{6561} & -\frac{212}{729} & & & \\
1 & \frac{9017}{3168} & -\frac{355}{33} & \frac{46732}{5247} & \frac{49}{176} & -\frac{5103}{18656} & & \\
1 & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & -\frac{2187}{6784} & \frac{11}{84} & \\
\hline
 & \frac{5179}{57600} & 0 & \frac{7571}{16695} & \frac{393}{640} & -\frac{92097}{339200} & \frac{187}{2100} & \frac{1}{40} \\
 & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & -\frac{2187}{6784} & \frac{11}{84} & 0 \\
\hline
 & \frac{71}{57600} & 0 & -\frac{71}{16695} & \frac{71}{1920} & -\frac{17253}{339200} & \frac{22}{525} & -\frac{1}{40}
\end{array}$$

5 The class of multistep methods

5.1 Introduction

In contrast to one-step methods, where each new step is defined solely in terms of the differential equation and the initial point given by the last numerical computed value, the multistep technique is based on a knowledge of a sequence of earlier numerical approximations of the solution at several previous steps. The most important multistep formulas considered by the codes for the numerical solution of (1) take the general form

$$\sum_{j=0}^{k} \alpha_j \mathbf{y}_{n+j} = h \sum_{j=0}^{k} \beta_j \mathbf{f}_{n+j}, \tag{59}$$

where the $\alpha_j, \beta_j, j = 0, 1, \ldots, k$ are real parameters defining the numerical method, h denotes the step size and

$$\mathbf{f}_{n+j} := \mathbf{f}(x_{n+j}, \mathbf{y}_{n+j}), \quad x_j = x_0 + jh \quad j = 0, 1, \ldots, k. \tag{60}$$

We shall also assume that

$$\alpha_k = 1, \quad |\alpha_0| + |\beta_0| > 0,$$

so the number of steps k in the formula and the coefficientes $\alpha_j, \beta_j, j = 0, \ldots, k$ could be defined unambiguously. The formula (59) gives a numerical approximation $\mathbf{y}_{n+k}$ a $\mathbf{y}(x_{n+k})$ once we know numerical approximations $\mathbf{y}_{n+k-1}, \ldots, \mathbf{y}_n$ to the solution at the k previous abscisas $x_{n+k-1}, \ldots, x_n$. If $\beta_k = 0$ the method is called explicit because the numerical solution $\{\mathbf{y}_n\}_{n=0}^N$ can be computed straightforwardly from the formula (59) if a starting procedure is provided to get the initial numerical values $\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_{k-1}$. Moreover, if a computer code implementing this method stores in two arrays the vectors

$$[\mathbf{y}_{n+k-1}, \ldots, \mathbf{y}_n]^T, \quad [\mathbf{f}_{n+k-1}, \ldots, \mathbf{f}_n]^T$$

and updates them after each new step, then to advance one step requires only one evaluation of the function $\mathbf{f}$.

When $\beta_k \neq 0$, the formula (59) defines $\mathbf{y}_{n+k}$ as the solution of a nonlinear system of equations that takes the form

$$\mathbf{y}_{n+k} = h\beta_k \mathbf{f}(x_{n+k}, \mathbf{y}_{n+k}) + \mathbf{g}, \tag{61}$$

where $\mathbf{g}$ denotes a known vector that depends on the numerical approximations to the solution previously computed. Then we call the method an implicit linear multistep method. Typically, an iterative method may be employed to solve the non-linear system for $\mathbf{y}_{n+k}$. In this situation, to advance one step may require several evaluations of the function $\mathbf{f}$ or its partial derivatives with respect the components of $\mathbf{y}$. This may seem to be a severe disadvantage, but in practice it is not so as will be seen later. For example, if the iterative method chosen is a fixed point iteration of the form

$$\mathbf{y}_{n+k}^{[\nu+1]} = h\beta_k \mathbf{f}(x_{n+k}, \mathbf{y}_{n+k}^{[\nu]}) + \mathbf{g}, \quad \nu = 0, 1, \ldots \quad \mathbf{y}_{n+k}^{[0]} \text{ arbitrary}$$

then the convergence of the process is guaranteed if the step size h is enough small to satisfy

$$h\,\beta_k\,L < 1, \tag{62}$$

where L is the Lipschitz constant of the function $\mathbf{f}$ with respect to $\mathbf{y}$. In general, when the Lipschitz constant L is of moderate size, the condition (62) is not more demanding that the restriction on the step size h to have an enough accurate approximation $\mathbf{y}_{n+k}$. However, for stiff problems, the Lipschitz constant L is of big magnitude and the restriction (62) on the time step can be unpractical. Then iterative methods based on the Newton's method or some variant of it may be considered.

A general theory of multistep methods was started by the work of G. Dahlquist (1956, 1959). After Dahlquist we associate to the linear multistep method (59) the polinomials

$$\rho(\zeta) = \sum_{j=0}^{k} \alpha_j \zeta^j, \quad \sigma(\zeta) = \sum_{j=0}^{k} \beta_j \zeta^j, \tag{63}$$

that are called respectively first and second characteristic polinomials of the method. If $\rho(\zeta)$ and $\sigma(\zeta)$ in (63) have a common factor $\phi(\zeta)$, then the polynomials

$$\tilde{\rho}(\zeta) = \frac{\rho(\zeta)}{\phi(\zeta)}, \quad \tilde{\sigma}(\zeta) = \frac{\sigma(\zeta)}{\phi(\zeta)},$$

are the characteristic polynomials of a new and simpler multistep method. This multistep method can be written in compact form as

$$\tilde{\rho}(E)\mathbf{y}_n = h\tilde{\sigma}(E)\mathbf{f}_n,$$

where E denotes the shift operator defined by $E\mathbf{y}_n = \mathbf{y}_{n+1}$. Multiplication by $\phi(E)$ shows that any numerical solution $\{\mathbf{y}_n\}$ of this method is also a solution of the original method $\rho(E)\mathbf{y}_n = h\sigma(E)\mathbf{f}_n$. The two methods are thus essentially equal. Therefore it is usually assumed that the characteristic polynomials $\rho(\zeta)$ and $\sigma(\zeta)$ have no common factor, and multistep methods satisfying this property are called irreducible.

5.2 The Adams-Bashforth and Adams-Moulton formulae

The methods of Adams are the most important linear multistep methods both from the historical point of view and from the enormous practical value they have had for the numerical solution of (1). They are even more ancient that Runge-Kutta methods, dating back to at least 1.855 when F. Bashfort reported to the Royal Society some application of a method of treating differential equations devised by John Couch Adams. The Adams formulae are strongly related with Lagrange's interpolation processes as the following derivation in the case of an ordinary differential equation emphasizes. We start writting down the ordinary differential equation in the integral form

$$y(x_{n+1}) = y(x_n) + \int_{x_n}^{x_{n+1}} f(x, y(x))dx. \tag{64}$$

If we assume that $y_n, y_{n-1}, \ldots, y_{n-k+1}$ are approximations to the solution $y(x)$ at the abscisas $x_n, \ldots, x_{n-k+1}$ we can replace the integrand in the equation (64) with the only polynomial $P^*_{k-1}(x)$ of degree less than or equal to $k-1$ that interpolates the data

$$(x_n, f_n), (x_{n-1}, f_{n-1}), \ldots, (x_{n-k+1}, f_{n-k+1}). \tag{65}$$

We obtain in this way the k-step Adams-Bashforth formula,

$$y_{n+1} = y_n + \int_{x_n}^{x_{n+1}} P^\star_{k-1}(x)dx. \tag{66}$$

Backward differences of the values $f_n, \ldots, f_{n-k+1}$, defined recursively by the formulas

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

are very suitable for the implementation of the Adams methods. Using the Newton's form of the interpolatory polinomial,

$$P^\star_{k-1}(x) = \sum_{j=0}^{k-1} \frac{1}{j!h^j}(x - x_n)\cdots(x - x_{n+1-j})\nabla^j f_n,$$

and making in the integral in (66) the change to a new integration variable s, defined by the relation $x = x_n + sh$, we obtain the formula

$$y_{n+1} = y_n + h\sum_{j=0}^{k-1} \gamma^*_j \nabla^j f_n, \tag{67}$$

where the coefficientes $\gamma_j^*, j = 0, \ldots, k-1$ are given by

$$\gamma_j^* = (-1)^j \int_0^1 \left(\begin{array}{c} -s \\ j \end{array} \right) ds. \tag{68}$$

It is important to emphasize that the coefficients $\gamma_j^*, j = 0, \ldots, k-1$ in (67) do not depend on the number of steps k in the formula. Therefore to advance the numerical solution with a formula with more steps require only to retain aditional terms in the formula (67). It is useful to introduce the generating function defined by the formal series

$$G^\star(t) = \sum_{j=0}^{\infty} \gamma_j^\star t^j, \tag{69}$$

and it is easy to show that

$$G^\star(t) = \frac{-t}{(1-t)\ln(1-t)}.$$

In particular, the coefficients γ_j^* satisfy the recursive equations

$$\gamma_i^\star + \frac{\gamma_{i-1}^\star}{2} + \cdots + \frac{\gamma_0^\star}{i+1} = 1, \quad i = 0, 1, 2, \ldots$$

from we can derive the first Adams-Bashforth formulas after truncating the series after the first k terms on the right hand side of

$$y_{n+1} = y_n + h(f_n + \frac{1}{2}\nabla f_n + \frac{5}{12}\nabla^2 f_n + \frac{3}{8}\nabla^3 f_n + \frac{251}{720}\nabla^4 f_n + \cdots).$$

Expanding the backward differences in terms of function values, we obtain the standard k-step Adams-Bashforth methods for $k = 1, 2, 3, 4$ respectively

$$\begin{aligned}
y_{n+1} - y_n &= hf_n, \\
y_{n+1} - y_n &= \frac{h}{2}(3f_n - f_{n-1}), \\
y_{n+1} - y_n &= \frac{h}{12}(23f_n - 16f_{n-1} + 5f_{n-2}), \\
y_{n+1} - y_n &= \frac{h}{24}(55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3}).
\end{aligned}$$

The Adams-Moulton formulas of k steps are similarly based on replacing the integrand function in (64) with the only polinomial $P_k(x)$ of degree less than or equal to k that interpolates the data

$$(x_{n+1}, f_{n+1}), (x_n, f_n), \ldots, (x_{n-k+1}, f_{n-k+1}).$$

Because the use of the ordinate $f_{n+1} := f(x_{n+1}, y_{n+1})$, the Adams-Moulton formulas define implicit linear multistep methods. In terms of the new variable s, $-1 \leq s \leq 0$, defined by $x = x_{n+1} + sh$, the Newton's form for the interpolatory polinomial $P_k(x)$ is given by

$$P_k(x_{n+1} + sh) = \sum_{j=0}^{k} (-1)^j \left(\begin{array}{c} -s \\ j \end{array} \right) \nabla^j f_{n+1},$$

and therefore

$$y_{n+1} - y_n = \int_{-1}^{0} P_k(x_{n+1} + sh)hds = h\sum_{j=0}^{k} \gamma_j \nabla^j f_{n+1}$$

with

$$\gamma_j = (-1)^j \int_{-1}^{0} \left(\begin{array}{c} -s \\ j \end{array} \right) ds.$$

It is easy to determine the generating function $G(t) = \sum_{j=0}^{\infty} \gamma_j t^j$, that now is given by

$$G(t) = \frac{-t}{\ln(1-t)}$$

from what we can derive the following recursive relations for the coefficients γ_j

$$(\gamma_0 + \gamma_1 t + \gamma_2 t^2 + \cdots)(1 + \frac{t}{2} + \frac{t^2}{3} + \cdots) = 1.$$

The first Adams-Moulton formulas can be obtained after we truncate the series at the right hand side of

$$y_{n+1} = y_n + h(f_{n+1} - \frac{1}{2}\nabla f_{n+1} - \frac{1}{12}\nabla^2 f_{n+1} - \frac{1}{24}\nabla^3 f_{n+1} - \frac{19}{720}\nabla^4 f_{n+1} + \cdots)$$

and we expand the backward differences in terms of ordinates. These are for k=1,2,3,4 respectively

$$\begin{aligned}
y_{n+1} - y_n &= \frac{h}{2}(f_{n+1} + f_n), \\
y_{n+1} - y_n &= \frac{h}{12}(5f_{n+1} + 8f_n - f_{n-1}), \\
y_{n+1} - y_n &= \frac{h}{24}(9f_{n+1} + 19f_n - 5f_{n-1} + f_{n-2}), \\
y_{n+1} - y_n &= \frac{h}{720}(251f_{n+1} + 646f_n - 264f_{n-1} + 106f_{n-2} - 19f_{n-3}).
\end{aligned}$$

We have the following identities among the coefficients γ_j^* and γ_j, $j = 0, 1, 2, \ldots$,

1. $\gamma_j^\star = \sum_{i=0}^{j} \gamma_i, \quad j = 0, 1, 2, \ldots$.
2. $\gamma_j^\star - \gamma_{j-1}^\star = \gamma_j, \quad j = 1, 2, \ldots$
3. $$\sum_{j=0}^{k-1} (\gamma_j \nabla^j f_{n+1} - \gamma_j^\star \nabla^j f_n) = \gamma_{k-1}^\star \nabla^k f_{n+1}, \quad k \geq 1.$$

If $y(x)$ is $k+1$-times continuously differentiable then the approximation error in

$$\int_{x_n}^{x_{n+1}} y'(x)dx \approx \int_{x_n}^{x_{n+1}} P(x)dx, \tag{70}$$

where $P(x)$ is the interpolating polinomial based for example in the exact values

$$(x_n, y'(x_n)), \ldots, (x_{n-k+1}, y'(x_{n-k+1})),$$

is given by

$$y'(x) = \sum_{j=0}^{k-1} (-1)^j \begin{pmatrix} -s \\ j \end{pmatrix} \nabla^j y'(x_n) + (-1)^k \begin{pmatrix} -s \\ k \end{pmatrix} h^k y^{(k+1)}(\xi), \qquad (71)$$

where $\xi = \xi(x)$ is a point in the interval (x_{n-k+1}, x). Integrating both sides of (71) in (x_n, x_{n+1}) gives

$$y(x_{n+1}) - y(x_n) = h \sum_{j=0}^{k-1} \gamma_j^\star \nabla^j y'(x_n) + \mathcal{R}_k^{AB}$$

where

$$\mathcal{R}_k^{AB} = (-1)^k h^k \int_{x_n}^{x_{n+1}} \begin{pmatrix} -s \\ k \end{pmatrix} y^{(k+1)}(\xi(x)) dx = h^{k+1} \gamma_{k+1}^\star y^{(k+1)}(\xi').$$

Therefore, the residual that it is obtained when in the k steps Adams-Bashforth formula we replace the numerical values y_j and f_j by the corresponding exact values $y(x_j), j = n+1, \ldots, n-k+1$ and $y'(x_j) = f(x_j, y(x_j)), j = n, \ldots, n-k+1$ respectively it is equal to the first term in (67) that it is neglected after truncation of the series with the backward difference of order k subtituted by the derivative of the same order $y^{(k)}(x)$ evaluated in a intermediate point. With the same analysis for the Adams-Moulton formulas we obtain

$$y(x_{n+1}) - y(x_n) = h \sum_{j=0}^{k} \gamma_j \nabla^j y'(x_{n+1}) + \mathcal{R}_{k+1}^{AM}$$

with

$$\mathcal{R}_{k+1}^{AM} = h^{k+2} \gamma_{k+2}^\star y^{(k+2)}(\xi') \qquad (72)$$

and ξ' a point in (x_{n-k+1}, x_{n+1}).

5.3 The backward differentiation formulas

Another important class of linear multistep methods are the backward differentiation formulae (BDF), that are derived through the use of a formula of numerical differentiation. An interpolant $Q_k(x)$ of degree k is based on the y values, including (x_{n+1}, y_{n+1}), rather than f values,

$$(x_{n+1}, y_{n+1}), (x_n, y_n), \ldots, (x_{n-k+1}, y_{n-k+1}).$$

In backward difference form, with $x = x_n + sh$, the interpolant $Q_k(x)$ is written

$$Q_k(x_n + sh) = \sum_{j=0}^{k} (-1)^j \begin{pmatrix} -s+1 \\ j \end{pmatrix} \nabla^j y_{n+1}. \qquad (73)$$

Assuming that this interpolant satisfies the differential equation at $x = x_{n+1}$, we obtain

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

The local truncation error of the BDF formulae can be obtained directly from the error of the interpolant (73), as was done for the quadrature based Adams methods,

$$hy'(x_{n+1}) - \sum_{j=1}^{k} \frac{1}{j} \nabla^j y(x_{n+1}) = \mathcal{R}_k^{BDF}$$

with

$$\mathcal{R}_k^{BDF} = \frac{h}{(k+1)!} y^{(k+1)}(\xi) L'(x_{n+1}), \quad \xi \in (x_{n-k+1}, x_{n+1})$$

and $L'(x) = (x - x_{n+1})(x - x_n) \cdots (x - x_{n-k+1})$. Because we have

$$L(x) = (-1)^{k+1} h^{k+1} \begin{pmatrix} -s \\ k+1 \end{pmatrix}$$

then, we finally obtain

$$\mathcal{R}_k^{BDF} = h^{k+1} \frac{1}{k+1} y^{(k+1)}(\xi), \quad \xi \in (x_{n-k+1}, x_{n+1}).$$

Only the BDF formulae for $1 \leq k \leq 6$ are zero-stable formulae and define useful k steps linear multistep methods. However, their importance stems from the excelent properties of stability that made them apropriate for solving stiff problems. The typical implementation of a general-purpose BDF code is quasi-constant step size. This means that the formulas used are those for a constant step size ant the step size is held constant during an integration unless there is good reason to change it. General-purpose BDF codes also vary the order during an integration.

5.4 Order conditions for linear multistep methods

The local truncation error (LTE) of the linear multistep method (59) at x_{n+k} is the amount by which the true solution $y(x)$ fails to satisfy the numerical formula. It is defined by

$$\mathbf{t}_{n+k} := \sum_{j=0}^{k} \alpha_j \mathbf{y}(x_{n+j}) - h \sum_{j=0}^{k} \beta_j \mathbf{f}(x_{n+j}, \mathbf{y}(x_{n+j})). \tag{74}$$

It is highly convenient to introduce the linear difference operator

$$\mathcal{L}[\mathbf{w}(x), h] = \sum_{j=0}^{k} \alpha_j \mathbf{w}(x + jh) - h \sum_{j=0}^{k} \beta_j \mathbf{w}'(x + jh), \tag{75}$$

defined for all the functions $\mathbf{w} : [a, b] \to I\!R^d$ continuously differentiables in $[a, b]$. In particular, $\mathbf{t}_{n+k} = \mathcal{L}[\mathbf{y}(x_n), h]$. Assuming that the function $\mathbf{w}(x)$ is enough smooth for that $\mathbf{w}(x + jh)$ and $\mathbf{w}'(x + jh)$ could be expanded in a Taylor series in the step size h, we can derive

$$\mathcal{L}[\mathbf{w}(x), h] = C_0 \mathbf{w}(x) + C_1 h \mathbf{w}'(x) + \cdots + C_q h^q \mathbf{w}^{(q)}(x) + \cdots \tag{76}$$

where the constants $C_0, \ldots, C_q, \ldots$ do not depend on the step size h and are given by the formulas

$$C_0 = \sum_{j=0}^{k} \alpha_j \equiv \rho(1)$$

$$C_1 = \sum_{j=0}^{k} (j\alpha_j - \beta_j) \equiv \rho'(1) - \sigma(1)$$

$$C_q = \frac{1}{q!} \sum_{j=0}^{k} j^q \alpha_j - \frac{1}{(q-1)!} \sum_{j=0}^{k} j^{q-1} \beta_j, \quad q = 2, 3, \ldots \tag{77}$$

Definition 5.1 *The linear multistep method (59) is said to be of order p if $C_0 = C_1 = \cdots = C_p = 0$, and $C_{p+1} \neq 0$.*

The linear multistep method (59) is said to be consistent if their order p is greater than or equal to 1.

The consistency conditions for a linear multistep method can therefore be written in the form

$$\rho(1) = 0, \quad \rho'(1) = \sigma(1).$$

Different methods of the same order are distinguished by the error constant. A natural measure of the accuracy of the method is seen to be

$$C = \frac{C_{p+1}}{\sigma(1)},$$

that it is usually adopted as the error constant of a linear multistep method.

The local error of a linear multistep method (59) is not a concept so natural as for one-step methods. By definition, the local error of the linear multistep method (59) at $x = x_{n+k}$ is defined by

$$\mathbf{el}_{n+k} := \mathbf{y}(x_{n+k}) - \tilde{\mathbf{y}}_{n+k},$$

where $\mathbf{y}(x)$ is the exact solution and $\tilde{\mathbf{y}}_{n+k}$ is the numerical solution obtained by the method at $x = x_{n+k}$ using the exact values of the solution $\mathbf{y}_{n+j} = \mathbf{y}(x_{n+j}), j = 0, 1, \ldots, k-1$ at previous steps. It is easy to prove that

$$\mathbf{y}(x_{n+k}) - \tilde{\mathbf{y}}_{n+k} = \left(\alpha_k I - h\beta_k \overline{\frac{\partial \mathbf{f}}{\partial \mathbf{y}}}(x_{n+k}, \eta_{n+k}) \right)^{-1} \mathbf{t}_{n+k}.$$

Here I is the identity matrix and

$$\overline{\frac{\partial \mathbf{f}}{\partial \mathbf{y}}}(x_{n+k}, \eta_{n+k})$$

denotes the Jacobian matrix $\dfrac{\partial \mathbf{f}}{\partial \mathbf{y}}$ whose rows are evaluated at possibly different values lying on the segment joining $\mathbf{y}(x_{n+k})$ and $\tilde{\mathbf{y}}_{n+k}$. Therefore, for explicit linear multistep methods with $\alpha_k = 1$, the local error $\mathbf{el}_{n+k}$ is essentially the same that the local truncation error $\mathbf{t}_{n+k}$. If the method is implicit, however, the local error can be considered a $O(h)$ perturbation of $\alpha_k^{-1} \mathbf{t}_{n+k}$.

The formulas (77) define the constants $C_p, p = 0, 1, 2, \ldots$ as linear forms in the $2k + 2$ coefficients α_j, β_j, $j = 0, \ldots, k$ of the method. Because these linear forms are linearly independent and we have esentially only $2k+1$ free parameters defining a k-step linear multistep method, the highest order we can attain is $2k$. However, these methods are of no practical significance because a famous result that it is known as the first Dahlquist-barrier (Dahlquist, 1956).

Theorem 5.1 *The order p of a 0-stable linear k-step method satisfies*

$$\begin{array}{ll} p \leq k+2 & \textit{si } k \textit{ es par} \\ p \leq k+1 & \textit{si } k \textit{ es impar} \\ p \leq k & \textit{si } \beta_k/\alpha_k \leq 0. \end{array}$$

The proof of this theorem is out of the scope of these lecture notes and we refer for it to Hairer, Norsett and Wanner (2000).

6 Stability and convergency of linear multistep methods

We consider now a k step linear multistep method of the form (59) defining a numerical solution of an uniform grid of step size $h > 0$. We also assume that we dispose of an starting procedure that can be made dependent on the parameter h of the discretization

$$\mathbf{y}_0(h), \quad \mathbf{y}_1(h), \quad \ldots, \quad \mathbf{y}_{k-1}(h). \tag{78}$$

The concept of convergence is essentially the same as for one-step methods.

Definition 6.1 *The method (59), (78) is said convergent if for all initial value problem (1), we have*

$$\lim_{h \to 0^+} \max_{k \leq n \leq N_h} \|\mathbf{y}(x_n) - \mathbf{y}_n\| = 0, \tag{79}$$

for all starting values $\mathbf{y}_n(h), n = 0, 1, \ldots, k-1$ such that they satisfy

$$\lim_{h \to 0^+} \|\mathbf{y}(x_0) - \mathbf{y}_n\| = 0, \quad n = 0, 1, \ldots, k-1.$$

The method is convergent of order p if this is the greatest integer such that

$$\max_{0 \leq n \leq N_h} \|\mathbf{y}(x_n) - \mathbf{y}_n\| = O(h^p), \quad (h \to 0^+) \tag{80}$$

for all initial value problem (1) with $\mathbf{f} \in C^p$.

We reformulate the method as an one-step method. For that we consider that at each step the k-step linear multistep method advance from the vector

$$\mathbf{Y}_n := [\mathbf{y}_n^T, \mathbf{y}_{n-1}^T, \ldots, \mathbf{y}_{n-k+1}^T]^T \tag{81}$$

to the new vector

$$\mathbf{Y}_{n+1} := [\mathbf{y}_{n+1}^T, \mathbf{y}_n^T, \ldots, \mathbf{y}_{n-k+2}^T]^T.$$

We introduce the matrix

$$A = \begin{bmatrix} -\alpha'_{k-1} & -\alpha'_{k-2} & \cdots & -\alpha'_1 & -\alpha'_0 \\ 1 & 0 & \cdots & 0 & 0 \\ & \ddots & \ddots & & \\ & & & 0 & 0 \\ & & & 1 & 0 \end{bmatrix} \tag{82}$$

with the coefficients $\alpha'_j = \alpha_j/\alpha_k, j = 1, \ldots, k$, and we denote

$$\mathbf{s}(h) := [\mathbf{y}_{k-1}^T(h), \mathbf{y}_{k-2}^T(h), \ldots, \mathbf{y}_0^T(h)]^T,$$

the vector of starting values of the method. Finally, we write down the method in the form

$$\mathbf{Y}_{k-1} = \mathbf{s}(h), \tag{83}$$

$$\mathbf{Y}_{n+1} = (A \otimes I)\mathbf{Y}_n + h(\mathbf{e}_1 \otimes I)\Phi(x_n, \mathbf{Y}_{n+1}, \mathbf{Y}_n, h). \tag{84}$$

Here $A \otimes I$ denotes the Kronecker product of the matrix A and the identity $d \times d$ matrix I, $\mathbf{e}_1$ denotes the vector in $\mathbb{R}^k$ with the first component equal to 1 and all the rest components equal to zero, and

$$\Phi(x_n, \mathbf{Y}_{n+1}, \mathbf{Y}_n, h) = \sum_{j=0}^{k} \beta_{k-j}\mathbf{f}_{n-j}.$$

We simplify in the following the notation by restricting us to the case $d = 1$ in which case $A \otimes I = A$.

Definition 6.2 *The method (83), (84) is* 0*-stable if positive constants* S *and* h_0 *exist such that for all initial value problem (1) and for all* $h \in (0, h_0]$*, the numerical solutions of (83), (84) and of the perturbed difference equations*

$$\tilde{\mathbf{Y}}_{k-1} = \mathbf{s}(h) + \mathbf{d}_{k-1}, \tag{85}$$

$$\tilde{\mathbf{Y}}_{n+1} = (A \otimes I)\tilde{\mathbf{Y}}_n + h(\mathbf{e}_1 \otimes I)\Phi(x_n, \tilde{\mathbf{Y}}_{n+1}, \tilde{\mathbf{Y}}_n, h) + \mathbf{d}_{n+1}, \tag{86}$$

satisfy

$$\max_{k-1 \le n \le N_h} \|\tilde{\mathbf{Y}}_n - \mathbf{Y}_n\| \le S \sum_{j=k-1}^{n} \|\mathbf{d}_j\| \tag{87}$$

for all arbitrary perturbations $\mathbf{d}_j \in \mathbb{R}^{dk}, j = k-1, \ldots, N_h - 1$.

We have the following important theorem

Theorem 6.1 a) *The method (83),(84) is* 0*-stable for all* $\mathbf{f}$ *as in (1) if and only if the method is stable for the initial value problem with* $\mathbf{f} \equiv 0$.
b) *The method (83),(84) is* 0*-stable for the initial value problem with* $\mathbf{f} \equiv \mathbf{0}$ *if and only if there is a positive constant* K *such that*

$$\sup_{0 \le nh \le b-a} \|A^n\| \le K. \tag{88}$$

The condition (88) can be caracterized in terms of the spectra of the matrix A.

Theorem 6.2 *For a matrix A of order k the following conditions are equivalent*
a) *There is a norm in $\mathbb{R}^{kd}$ such that $\|(A \otimes I)\| \leq 1$.*
b) *There is a positive constant K satisfying the condition (88).*
c) *If λ is an eigenvalue of A then: or $|\lambda| < 1$ or if $|\lambda| = 1$ and the eigenvalue λ has a multiplicity of q then necessarily λ has associated q linearly independent eigenvectors.*

For a matrix A with the Frobenius form in (82), the characteristic polynomial is given by $p(\lambda) = (-1)^k \rho(\lambda)$. Then, if $\rho(\lambda) = 0$, the vector $(\lambda^{k-1}, \lambda^{k-2}, \ldots, 1)$ is the only eigenvector associated to λ and we have

Corolario 6.1 *The method (83), (84) with A given by (82) is 0-stable if and only if the characteristic polynomial of A*

$$\rho(\lambda) = \alpha_k \lambda^k + \alpha_{k-1} \lambda^{k-1} + \cdots + \alpha_1 \lambda + \alpha_0 \tag{89}$$

has all the roots of magnitude less than or equal to one, and the roots of magnitude 1 are necessarily simple.

This condition is called the root condition for the linear multistep method.

The concept of consistency it is a natural extension of the one for one-step methods. We introduce the local truncation error $\mathbf{t}_{n+1}$ of the method (59) at $x = x_{n+1}$ as the residual we obtain when we substitute in the difference equation of the method the exact values of the solution $\mathbf{y}(x_n)$.

$$\mathbf{t}_{n+1} := \sum_{j=0}^{k} \alpha_{k-j} \mathbf{y}(x_{n-j}) - h \sum_{j=0}^{k} \beta_{k-j} \mathbf{f}(x_{n-j}, \mathbf{y}(x_{n-j})).$$

Definition 6.3 *The method (59) is consistent if for all initial value problem (1)*

$$\lim_{h \to 0^+} \max_{k \leq n \leq N_h} \| \frac{\mathbf{t}_n}{h} \| = 0. \tag{90}$$

The method is consistent of order p if for all initial value problem (1) with $\mathbf{f} \in C^p$ we have

$$\max_{k \leq n \leq N_h} \| \frac{\mathbf{t}_n}{h} \| = O(h^p) \quad (h \to 0). \tag{91}$$

Finally we formulate the fundamental theorem

Theorem 6.3 *(Dahlquist, 1959) The method (59) is convergent if and only if is consistente and 0-stable.*

7 The concept of absolute stability

The concept of 0-stability concerns with the propagation properties by the numerical method of the errors introduced at each step of the numerical integration when the step size h tends to zero. Although this limit process is relevant for the

convergence of the numerical method, in real situations the numerical method is applied for a fixed sequence of successive step sizes

$$h_0, h_1, \ldots, h_{N-1}, \quad h_i := x_{i+1} - x_i, \quad i = 0, \ldots, N-1$$

and in particular, to consider a more academic and tractable situation, when the step size h is fixed and uniform along all the numerical integration. Therefore it is of great interest to analyze the behavior of the numerical solution obtained in this way under the effect of numerical perturbations of the method. A preliminary analysis requires to consider the restricted class of problems formed by the linear systems of ordinary differential equations of the form

$$\mathbf{y}' = J\mathbf{y} + \mathbf{g}(x), \tag{92}$$

with J a diagonalizable constant real matrix $d \times d$, and $\mathbf{g}(x)$ a smooth vector valued function. In this situation, the difference $\mathbf{e}(x)$ between two solutions $\mathbf{y}(x), \mathbf{z}(x)$ of (92) with initial conditions $\mathbf{y}(a) = \mathbf{A}$ and $\mathbf{z}(a) = \mathbf{A} + \delta_0$ respectively is a solution of the homogeneous linear system

$$\mathbf{e}' = J\mathbf{e}, \quad a \leq x \leq b, \quad \mathbf{e}(a) = \delta_0. \tag{93}$$

We denote $\lambda_1, \lambda_2, \ldots, \lambda_d$ the eigenvalues of J and $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d$ the corresponding eigenvectors, and we put

$$Q^{-1}JQ = \Lambda := diag(\lambda_1, \lambda_2, \ldots, \lambda_d), \quad Q := [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d].$$

Then the change of variables $\mathbf{e} = Q\eta$ allows to decouple the equations in (93) to obtain

$$(\eta^r)' = \lambda_r \eta^r, \quad r = 1, \ldots, d. \tag{94}$$

The component $\eta^r(x) = \exp(\lambda_r(x-a))\eta^r(a)$ grows exponentially when $x \to \infty$ if $\Re\lambda_r > 0$ and $\eta^r(a) \neq 0$. When $\Re\lambda_r \leq 0$, $\eta^r(x)$ is bounded by $\eta^r(a)$ for all $x \geq 0$ and if $\Re\lambda_r < 0$ then $\eta^r(x) \to 0$ exponentially when $x \to \infty$. The same conclusions can be derived for $\|\mathbf{e}(x)\|$.

7.1 Linear stability theory for linear multistep methods

We consider the application of a linear multistep method

$$\sum_{j=0}^{k} \alpha_j \mathbf{y}_{n+j} = h \sum_{j=0}^{k} \beta_j \mathbf{f}_{n+j}, \quad |\alpha_0| + |\beta_0| \neq 0 \tag{95}$$

to the linear system (92). Let $\rho(\zeta), \sigma(\zeta)$ denote the characteristic polynomials of the method. The difference $\{\mathbf{e}_n\}$ between two numerical solutions $\{\mathbf{y}_n\}$ y $\{\mathbf{z}_n\}$, with $\{\mathbf{z}_n\}$ given by the method when it is started with pertubed $\mathbf{z}_0 = \mathbf{y}_0 + \delta_0, \ldots, \mathbf{z}_{k-1} = \mathbf{y}_{k-1} + \delta_{k-1}$, satisfy the system of linear difference equations

$$\sum_{j=0}^{k} (\alpha_j I - h\beta_j J)\mathbf{e}_{n+j} = \mathbf{0}, \tag{96}$$

where I denotes the $d \times d$ identity matrix. It occurs that the matrix Q that decouple the equations (93) also decouple the system (96). Therefore, if we put $\mathbf{e}_n = Q\eta_n$, then

$$\sum_{j=0}^{k} (\alpha_j - h\beta_j\lambda_r)\eta_{n+j}^r = 0, \quad r = 1, \ldots, d. \tag{97}$$

Each equation in (97) is a linear difference equation with constant complex coefficients. The general solution of this kind of equation is given by

$$\eta_n^r = \sum_{m=1}^{k} \chi_{rm}(\zeta_m^r)^n, \quad r = 1, 2, \ldots, d$$

where χ_{rm} are arbitrary complex constants and for each r, $\zeta_1^r, \ldots, \zeta_k^r$ denote the roots of the characteristic polynomial of the difference equation (97)

$$\pi(\zeta, \hat{h}) = \rho(\zeta) - \hat{h}\sigma(\zeta) = 0, \quad \hat{h} := \lambda_r h$$

The polynomial $\pi(\zeta, \hat{h})$ is called the stability polynomial of the linear multistep method. Clearly, $\{\eta_n^r\}_{n=1}^{\infty}$, is uniform bounded if and only if the polynomial $\pi(\zeta, \lambda_r h)$ satisfy the root condition. We have

Definition 7.1 *The set*

$$\mathcal{R}_A := \{\hat{h} \in \mathbb{C} : \pi(\zeta, \hat{h}) \textit{satisface la condición de la raíz}\}$$

is called absolute stability domain of the method (95).

For example, the explicit Euler and the implicit Euler methods have respectively the stability polynomials

$$\pi_{Euler}(\zeta, \hat{h}) = \zeta - (1 + \hat{h}), \quad \pi_{EulerI}(\zeta, \hat{h}) = (1 - \hat{h})\zeta - 1$$

and the absolute stability domains for these methods are respectively the unit disk centered in -1 and the exterior of the open unit disk centered in 1. The stability domain for the trapezoidal rule is the left plane $\Re\hat{\lambda} \leq 0$ and for the midpoint rule reduces only to the segment $\hat{h} = i\mu$, $-1 \leq \mu \leq 1$. In general, the absolute stability region $\mathcal{R}_A$ of a method must be determined numerically. A practical graphical technique, called the root locus boundary method, plots the parametric curve in the complex plane given by

$$\gamma(\theta) = \rho(\exp(i\theta))/\sigma(\exp(i\theta)), \quad \theta \in [-\pi, \pi]. \tag{98}$$

The boundary $\partial\mathcal{R}_A$ of the absolute stability domain is a subset of the graph $\Gamma = \gamma([-\pi, \pi])$ of this curve because for all $\hat{h} \in \partial\mathcal{R}_A$, at least one of the roots of $\pi(\zeta, \hat{h}) = 0$ must be of module 1 and therefore can be represented as $\zeta = \exp(i\theta)$.

7.2 Linear stability theory for RK methods

We consider now the use of a general RK method of s stages

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\sum_{i=1}^{s} b_i\mathbf{f}(x_n + c_i h, \mathbf{Y}_i) \tag{99}$$

$$\mathbf{Y}_i = \mathbf{y}_n + h\sum_{j=1}^{s} a_{ij}\mathbf{f}(x_n + c_j h, \mathbf{Y}_j), \quad i = 1, \ldots, s,$$

with Butcher array

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s, \end{array} = \begin{array}{c|c} \mathbf{c} & A \\ \hline & \\ & \mathbf{b}^T \end{array}$$

to the linear test problem (92). Let $\mathbf{y}_n, n = 0, \ldots$, $\mathbf{z}_n, n = 0, \ldots$ be two numerical solutions given by the formulas (99) when applied to the test problem (92), with $\mathbf{z}_0 = \mathbf{y}_0 + \delta_0$. Let $\mathbf{Y}_i, i = 1, \ldots, s$ and $\mathbf{Z}_i, i = 1, \ldots, s$ denote also the inner stages when the method advance from $\mathbf{y}_n$ to $\mathbf{y}_{n+1}$ from $\mathbf{z}_n$ to $\mathbf{z}_{n+1}$ respectively. The difference between the two solutions $\mathbf{e}_n := \mathbf{z}_n - \mathbf{y}_n, n \geq 0$, satisfy the formulas

$$\begin{aligned} \mathbf{e}_{n+1} &= \mathbf{e}_n + h \sum_{i=1}^{s} b_i J \mathbf{E}_i \\ \mathbf{E}_i &= \mathbf{e}_n + h \sum_{j=1}^{s} a_{ij} J \mathbf{E}_j, \quad i = 1, \ldots, s, \end{aligned} \tag{100}$$

in which $\mathbf{E}_i := \mathbf{Z}_i - \mathbf{Y}_i, i = 1, \ldots, s$. Again the change of variables $\mathbf{e}_n = Q\eta_n, n \geq 0$ and $\mathbf{E}_i = Q\Xi_i, i = 1, \ldots, s$, decouple the equations in (100) and we obtain for $r = 1, \ldots, d$

$$\begin{aligned} \eta_{n+1}^r &= \eta_n^r + h \sum_{i=1}^{s} b_i \lambda_r \Xi_i^r \\ \Xi_i^r &= \eta_n + h \sum_{j=1}^{s} a_{ij} \lambda_r \Xi_j^r, \quad i = 1, \ldots, s. \end{aligned}$$

These are the same numerical approximations we have obtained if the method (99) were used to integrate the decoupled system (94). Therefore we have reduced the analysis to study the effect of perturbations on the numerical solution of the scalar test problem

$$y' = \lambda y, \quad \lambda \in \mathbb{C}, \quad \Re\lambda \leq 0. \tag{101}$$

In general, when the formulas (99) are applied to the problem (101) we obtain

$$y_{n+1} = R(\hat{h}) y_n, \quad \hat{h} := \lambda h \in \mathbb{C}.$$

The function $R(\hat{h})$ is called the stability function of the method and we have that $\{y_n\}_{n=0}^{\infty}$ is uniformly bounded if and only if

$$|R(\hat{h})| \leq 1.$$

Therefore, the perturbations $\{\mathbf{e}_n\}_{n=1}^{\infty}$ in (100), remain uniformly bounded for all initial perturbation δ_0, if and only if $|R(\lambda_r h)| \leq 1$ for all eigenvalue $\lambda_r, r = 1, \ldots, d$ of J.

Definition 7.2 *The set*

$$\mathcal{R}_A := \{\hat{h} \in \overline{\mathbb{C}} : |R(\hat{h})| \leq 1\}$$

is called the absolute stability domain of the method (99).

8 Predictor-Corrector Methods

To motivate this important class of methods we start with an implicit linear multistep method

$$\mathbf{y}_{n+k} + \sum_{j=0}^{k-1} \alpha_j \mathbf{y}_{n+j} = h\beta_k \mathbf{f}(x_{n+k}, \mathbf{y}_{n+k}) + h \sum_{j=0}^{k-1} \beta_j \mathbf{f}_{n+j}, \tag{102}$$

with $|\alpha_0| + |\beta_0| \neq 0, \beta_k \neq 0$, and $\alpha_k = 1$. To step from x_{n+k-1} to x_{n+k} with this method it is necessary to solve for $\mathbf{y}_{n+k}$ the nonlinear system defined by the equations (102) and this can be accomplish, if the step size h is enough small, through the fixed point iteration defined by

$$\begin{aligned} &\mathbf{y}_{n+k}^{[0]} \quad \text{arbitrary} \qquad (103)\\ &\mathbf{y}_{n+k}^{[\nu+1]} + \sum_{j=0}^{k-1} \alpha_j \mathbf{y}_{n+j} = h\beta_k \mathbf{f}(x_{n+k}, \mathbf{y}_{n+k}^{[\nu]}) + h \sum_{j=0}^{k-1} \beta_j \mathbf{f}_{n+j}, \quad \nu = 0, 1, \ldots \end{aligned}$$

A reasonable initial approximation $\mathbf{y}_{n+k}^{[0]}$ for this process can be computed with a different explicit linear multistep method. The explicit linear multistep method is called the Predictor, the implicit linear multistep method is called the Corrector and both methods define a pair Predictor-Corrector pair. We collect the formulas for both methods in respectively

$$\sum_{j=0}^{k} \alpha_j^* \mathbf{y}_{n+j} = h \sum_{j=0}^{k-1} \beta_j^* \mathbf{f}_{n+j}, \quad \textit{predictor} \tag{104}$$

$$\sum_{j=0}^{k} \alpha_j \mathbf{y}_{n+j} = h \sum_{j=0}^{k} \beta_j \mathbf{f}_{n+j}, \quad \textit{corrector} \tag{105}$$

and we assume $|\alpha_0^*| + |\beta_0^*| \neq 0$, $\alpha_k^* \neq 0$ and $\alpha_k \neq 0$.

In a predictor-corrector pair we can iterate with the corrector (103) until the convergence, i.e. until two successive iterations satisfy some criteria for convergence as for example $\|\mathbf{y}_{n+k}^{[\nu+1]} - \mathbf{y}_{n+k}^{[\nu]}\| < \epsilon$ with ϵ, for some $\epsilon > 0$ of the size of the machine number. . The main inconvenient of this approach is that we can not know *a priori* the number of iterations that the method will make at each step. The standard approach it is to iterate with the corrector only a fixed number of iterations. For example, (104) can be used to compute a quantity $\mathbf{y}_{n+k}^{[0]}$ (the predicted value), then we modify the term $\mathbf{f}_{n+k}$ in the corrector formula (105) by replacing the argument $\mathbf{y}_{n+k}$ of $\mathbf{f}$ by $\mathbf{y}_{n+k}^{[0]}$ to get $\mathbf{f}_{n+k}^{[0]} := \mathbf{f}(x_{n+k}, \mathbf{y}_{n+k}^{[0]}$ (first evaluation of $\mathbf{f}$), the second formula in (103) with $\nu = 0$ is used to obtain a first correction $\mathbf{y}_{n+k}^{[1]}$, and again we evaluate $\mathbf{f}_{n+k}^{[1]} := \mathbf{f}(x_{n+k}, \mathbf{y}_{n+k}^{[1]})$ to be used in succeeding steps. We said in this case that the pair of methods (104), (105) it is said implemented in the $PECE$ mode. By contrast the predictor-corrector method can be used in PEC mode if the last evaluation of $\mathbf{f}$ were omitted. Also we have $P(EC)^m E$ modes and $P(EC)^m$ modes in which the corrector formula in (103) it is used a fixed number m of times before completing the step. We collect all the formulas of the mode $P(EC)^m E^{1-t}$,

with $t = 0$ or $t = 1$,

$$
\begin{aligned}
P: \quad & \mathbf{y}_{n+k}^{[0]} + \sum_{j=1}^{k-1} \alpha_j^* \mathbf{y}_{n+j}^{[m]} = h \sum_{j=0}^{k-1} \beta_j^* \mathbf{f}_{n+j}^{[m-t]}, \\
(EC)^m: \quad & \mathbf{f}_{n+k}^{[\nu]} = f(x_{n+k}, \mathbf{y}_{n+k}^{[\nu]}) \\
& \mathbf{y}_{n+k}^{[\nu+1]} + \sum_{j=0}^{k-1} \alpha_j \mathbf{y}_{n+j}^{[m]} = h\beta_k \mathbf{f}_{n+k}^{[\nu]} + h \sum_{j=0}^{k-1} \beta_j \mathbf{f}_{n+j}^{[m-t]} \\
& \qquad \nu = 1, \ldots, m-1 \\
E^{(1-t)}: \quad & \mathbf{f}_{n+k}^{[m]} = f(x_{n+k}, \mathbf{y}_{n+k}^{[m]}), \quad \text{si } t = 0.
\end{aligned}
$$

At this point we make only a brief account of the order theory for predictor-corrector schemes. Let $\mathbf{y}_{n+k}^{[0]}, \mathbf{y}_{n+k}^{[1]}, \ldots, \mathbf{y}_{n+k}^{[m]}$ denote the sequence of the predictor value and the m corrected values for a predictor-corrector pair in $P(EC)^m E^{1-t}$ mode ($t = 0, 1$). We assume that the predictor linear multistep method has order p^* and constant error $C_{p^*+1}^*$ and that the corrector linear multistep method has order p with constant error C_{p+1}. We will denote $\tilde{\mathbf{y}}_{n+k}^{[\nu]}$, $\nu = 0, 1, \ldots, m$, the numerical approximations we would obtain with the predictor-corrector method under the assumption that $\mathbf{y}_{n+j}^{[\nu]} = \mathbf{y}(x_{n+j}), j = 0, 1, \ldots, k-1$ (localization assumption). Then we have

$$
\mathbf{y}(x_{n+k}) - \tilde{\mathbf{y}}_{n+k}^{[0]} = C_{p^*+1}^* h^{p^*+1} \mathbf{y}^{(p^*+1)}(x) + O(h^{p^*+2}), \tag{106}
$$

for the predicted value, and for the corrected values $\mathbf{y}_{n+k}^{[\nu+1]}$, $\nu = 0, 1, \ldots, m-1$,

$$
\begin{aligned}
& \mathbf{y}(x_{n+k}) - \tilde{\mathbf{y}}_{n+k}^{[\nu+1]} = h\beta_k [\mathbf{f}(x_{n+k}, \mathbf{y}(x_{n+k})) - \mathbf{f}(x_{n+k}, \tilde{\mathbf{y}}_{n+k}^{[\nu]})] \\
& + C_{p+1} h^{p+1} \mathbf{y}^{(p+1)}(x) + O(h^{p+2}).
\end{aligned} \tag{107}
$$

A recursive argument can then be followed to prove that, if $p^* \geq p$, then

$$
\mathbf{y}(x_{n+k}) - \tilde{\mathbf{y}}_{n+k}^{[m]} = C_{p+1} h^{p+1} \mathbf{y}^{(p+1)}(x_n) + O(h^{p+2}). \tag{108}
$$

In general, if $p^* < p$, the predictor linear multistep method gives an approximation $\tilde{\mathbf{y}}_{n+k}^{[0]}$ of order p^*, and the successive corrections $\tilde{\mathbf{y}}_{n+k}^{[1]}, \tilde{\mathbf{y}}_{n+k}^{[2]}, \ldots$ are approximations of respectively order $p^*+1, p^*+2, \ldots$, until we attain the order p of the corrector. After then, additional corrections does not arise the order of the new approximations but the principal term of the local truncation error of these approximations became the same of the principal term of the local truncation error for the corrector linear multistep method.

With the two estimates in (106) and (108) the estimation of the error in a predictor-corrector pair is quite straightforward when $p = p^*$. For example, the formula

$$
C_{p+1} h^{p+1} \mathbf{y}^{(p+1)}(x_n) = \frac{C_{p+1}}{C_{p+1}^* - C_{p+1}} (\mathbf{y}_{n+k}^{[m]} - \mathbf{y}_{n+k}^{[0]}), \tag{109}
$$

in which we have suppressed the tildes in the values $\overset{[\nu]}{n+k}$ that were remembering us that the localization hypothesis was in effect, gives an asymptotically correct estimate of the local truncation error in the corrected value $\mathbf{y}_{n+k}^{[m]}$. The process

of adding on the error estimate to this value is sometimes called Milne's device but amounts simply to local extrapolation. The most important use of the estimate (109) is for the determination of an appropriate step-size h_n when the method is implemented with a variable step size strategy.

8.1 Predictor-Corrector methods based in Adams formulas

Good stability characteristics make Adams Predictor-Corrector pairs the most commonly used formulae in the codes for non stiff problems. If we use the k-step Adams-Bashforth method as predictor and the $k-1$ step Adams-Moulton method as corrector we can use the Milne's device because both methods have the same order. Most of the codes base on the formulas in differences

$$\begin{aligned}
\mathbf{y}_{n+1} - \mathbf{y}_n &= h\sum_{j=0}^{k-1} \gamma_j^* \nabla^j \mathbf{f}_n, \quad p^* = k, \quad C_{k+1}^* = \gamma_k^* \\
\mathbf{y}_{n+1} - \mathbf{y}_n &= h\sum_{j=0}^{k-1} \gamma_j \nabla^j \mathbf{f}_{n+1}, \quad p = k, \quad C_{k+1} = \gamma_k.
\end{aligned} \tag{110}$$

We introduce the notation $\nabla_\nu^j \mathbf{f}_{n+1}$ to denote that the difference operator acts on the ordinate values $\mathbf{f}_{n+1}^{[\nu]}, \mathbf{f}_n^{[m-t]}, \ldots, \mathbf{f}_{n-k+1}^{[m-t]}$. Then, the predictor-corrector scheme in $P(EC)^m E^{(1-t)}$ is defined by

$$P: \qquad \mathbf{y}_{n+1}^{[0]} = \mathbf{y}_n^{[m]} + h\sum_{j=0}^{k-1} \gamma_j^* \nabla^j \mathbf{f}_n^{[m-t]}, \tag{111}$$

$$(EC)^m: \qquad \mathbf{f}_{n+1}^{[\nu]} = f(x_{n+1}, \mathbf{y}_{n+1}^{[\nu]}) \tag{112}$$

$$\mathbf{y}_{n+1}^{[\nu+1]} = \mathbf{y}_n^{[\mu]} + h\sum_{j=0}^{k-1} \gamma_j \nabla_\nu^j \mathbf{f}_{n+1}^{[m-t]} \tag{113}$$

$$\nu = 0, 1, \ldots, m-1 \tag{114}$$

$$E^{(1-t)}: \qquad \mathbf{f}_{n+1}^{[m]} = f(x_{n+1}, \mathbf{y}_{n+1}^{[m]}), \quad \text{if } t = 0. \tag{115}$$

We can avoid to store the differences $\nabla_\nu^j \mathbf{f}_{n+1}^{[m-t]}$, $j = 1, \ldots, k-1$, by using the equivalent formulation given by

$$\mathbf{y}_{n+1}^{[1]} = \mathbf{y}_{n+1}^{[0]} + h\gamma_{k-1}^* \nabla_0^k \mathbf{f}_{n+1}^{[m-t]}, \tag{116}$$

and

$$\mathbf{y}_{n+1}^{[\nu+1]} = \mathbf{y}_n^{[\nu]} + h\gamma_{k-1}^* (\mathbf{f}_{n+1}^{[\nu]} - \mathbf{f}_{n+1}^{[\nu-1]}), \quad \nu = 1, 2, \ldots, m-1. \tag{117}$$

Finally, the estimate of the local truncation error is obtained by summing all the formulas in (117) for $\nu = 1, 2, \ldots, m-1$, to obtain

$$\mathbf{y}_{n+1}^{[m]} - \mathbf{y}_{n+1}^{[0]} = h\gamma_{k-1}^* \nabla_{m-1}^k \mathbf{f}_{n+1}^{[m-t]},$$

and

$$\mathbf{t}_{n+1} = h\gamma_k \nabla_{m-1}^k \mathbf{f}_{n+1}^{[m-t]}. \tag{118}$$

At the beginning of the step we dispose of the differences $\nabla^j \mathbf{f}_n^{[m-t]}, j = 0, \ldots, k-1$. Then we compute $\nabla_0^k \mathbf{f}_{n+1}^{[m-t]}$ to be used in the first correction using the recurrence

$$\nabla_0^{j+1} \mathbf{f}_{n+1}^{[m-t]} = \nabla_0^j \mathbf{f}_{n+1}^{[m-t]} - \nabla^j \mathbf{f}_n^{[m-t]},$$

and then the difference

$$\nabla_{m-1}^k \mathbf{f}_{n+1}^{[m-t]} = \nabla_0^k \mathbf{f}_{n+1}^{[m-t]} + \mathbf{f}_{n+1}^{[m-1]} - \mathbf{f}_{n+1}^{[0]}$$

that it is used in (118). At the end of the step we updated the differences by summing $\mathbf{f}_{n+1}^{[m-t]} - \mathbf{f}_{n+1}^{[0]}$ to $\nabla_0^j \mathbf{f}_{n+1}^{[m-t]}, j = 1, 2, \ldots, k-1$.

8.2 Variable step size in linear multistep methods

8.2.1 Interpolatory tehcniques

Changing the step-size in linear multistep methods and in predictor-corrector schemes is not so straightforward as for one-step methods. Any change in step-size requires the computation of new starter information if it is wanted to use the same linear multistep formula. For a step length change from h to αh these new values will be approximations to the solution at the abscises $x_n - \alpha h, x_n - 2\alpha h, \ldots, x_n - (k-1)\alpha h$ if a k step formula is used at $x = x_n$. If accuracy is to be maintained following a revision of steplength, the interpolating mechanism adopted for the new starter information must be of a sufficiently high algebraic order. To yield the highest order of accuracy, the Hermite interpolating polynomial, employing the data $(x_{n-j}, \mathbf{y}_{n-j}, \mathbf{f}_{n-j} = \mathbf{y}'_{n-j})$, $j = 0, 1, \ldots, k-1$, could be chosen.

For a k step Adams formula the stepping from $x = x_n$ to $x = x_{n+1}$ amounts to the transformation between the data

$$(\mathbf{y}_n, \mathbf{y}'_n, \ldots, \mathbf{y}'_{n-k+1}) \to (\mathbf{y}_{n+1}, \mathbf{y}'_{n+1}, \ldots, \mathbf{y}'_{n-k+2}).$$

and for doing this it is used the polynomial $I(x)$ of degree $\leq k$ that interpolate the data $(x_n, \mathbf{y}'_n), \ldots, (x_{n-k+1}, \mathbf{y}'_{n-k+1})$. Similarly, for a k step BDF formula, the stepping from $x = x_n$ to $x = x_{n+1}$ is equivalent to the transformation

$$(\mathbf{y}_n, \mathbf{y}_{n-1}, \ldots, \mathbf{y}_{n-k+1}, \mathbf{y}'_{n-k}) \to (\mathbf{y}_{n+1}, \mathbf{y}_n, \ldots, \mathbf{y}_{n-k+2}, \mathbf{y}'_{n+1}),$$

and it is represented by the polynomial $I(x)$ of degree $\leq k$ that interpolates the data $(x_n, \mathbf{y}_n), \ldots, (x_{n-k+1}, \mathbf{y}_{n-k+1}), (x_n, \mathbf{y}'_n)$.

A change of the step size in a k step formula at $x = x_n$ requires to apply the interpolation process in at least all the following $k-1$ steps. This is a conceptual simple process but very expensive and researchers have designed other representations of the interpolatory polynomials $I(x)$ in terms of appropriate linear functionals of the original data that facilitates the change of the step size. For example, a very important strategy used for first time by Gear in his code DIFSUB was the introduction of the Nordsieck vector

$$[I(x_n), h_n I'(x_n), \frac{h_n^2}{2!} I''(x_n), \ldots, \frac{h^{k-1}}{(k-1)!} I^{(k-1)}(x_n)].$$

to store the necessary information at $x = x_n$ of the interpolating polynomial $I(x)$. The interest of this vector is that when the step size is changed from h_n to $h_{n+1} = \alpha_n h_n$ the vector it is changed by multiplying it by the diagonal matrix $D(\alpha_n) =$ diag $(1, \alpha_n, \alpha_n^2, \ldots, \alpha_n^{k-1})$.

8.2.2 Adams formulas with variable coefficients

The alternative approach is to consider the construction of linear multistep formulas based on non uniform grids. In general, and restricting the attention only to formulas of Adams type, these have the structure

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h_n \sum_{j=0}^{k} \beta_j(h_n, \ldots, h_{n-k+1}) \mathbf{f}_{n+1-j} \tag{119}$$

in which the coefficients $\beta_j, j = 0, \ldots, k$, depend on the step lengths of the k previous steps through the quotients $\dfrac{h_{n-1}}{h_n}, \ldots, \dfrac{h_{n-k+1}}{h_{n-k+2}}$.

The Adams formula based on the data $(x_{n-k+1}, \mathbf{f}_{n-k+1}), \ldots, (x_n, \mathbf{f}_n)$ can be derived similarly to the case of step size h constant by using the Newton divided difference formula for the interpolating polynomial. We have

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h_n \sum_{j=0}^{k-1} \alpha_j(n) \mathbf{f}[x_n, \ldots, x_{n-j}], \tag{120}$$

where

$$\alpha_j(n) = \frac{1}{h_n} \int_{x_n}^{x_{n+1}} \prod_{i=0}^{j-1} (x - x_{n-i}) dt, \tag{121}$$

and $\mathbf{f}[x_n, \ldots, x_{n-j}]$ denotes the j-th divided difference, which can be computed with the following recursive formulas

$$\begin{aligned} &\mathbf{f}[x_n] = \mathbf{f}_n \\ &\mathbf{f}[x_n, \ldots, x_{n-j}] = \frac{\mathbf{f}[x_n, \ldots, x_{n-j+1}] - \mathbf{f}[x_{n-1}, \ldots, x_{n-j}]}{x_n - x_{n-j}}, \quad j = 1, \ldots \end{aligned}$$

For computational purposes it is suitable to introduce the generalized backward differences given by

$$\mathbf{\Phi}_j(n) := \left(\prod_{i=1}^{j} (x_n - x_{n-i}) \right) \mathbf{f}[x_n, \ldots, x_{n-j}], \quad j = 0, \ldots; \tag{122}$$

that can be also computed recursively by the formulas

$$\begin{aligned} &\mathbf{\Phi}_0(n) = \mathbf{f}_n, \\ &\mathbf{\Phi}_j(n) = \mathbf{\Phi}_{j-1}(n) - \beta_{j-1}(n) \mathbf{\Phi}_{j-1}(n-1), \end{aligned}$$

where $\beta_0(n) = 1$ and

$$\beta_j(n) = \prod_{i=1}^{j} \frac{x_n - x_{n-i}}{x_{n-1} - x_{n-1-i}}, \quad j = 1, 2, \ldots. \tag{123}$$

With this notations the explicit k step Adams method is given by

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h_n \sum_{j=0}^{k-1} g_j(n) \beta_j(n+1) \mathbf{\Phi}_j(n), \tag{124}$$

where

$$g_j(n) = \frac{1}{h_n}\int_{x_n}^{x_{n+1}} \prod_{i=0}^{j-1} \frac{x - x_{n-i}}{x_{n+1} - x_{n-i}} dt. \tag{125}$$

If we add to the interpolating polynomial of the explicit method the term

$$\left(\prod_{i=0}^{k-1}(x - x_{n-i})\right) \mathbf{f}[x_{n+1}, \ldots, x_{n-k+1}]$$

we get the polynomial of degree k that interpolate the $k+1$ data

$$(x_{n-k+1}, \mathbf{f}_{n-k+1}), \ldots, (x_n, \mathbf{f}_n), (x_{n+1}, \mathbf{f}_{n+1}).$$

Therefore, the k step implicit Adams method is given by the formula

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h_n \sum_{j=0}^{k-1} g_j(n)\beta_j(n+1)\mathbf{\Phi}_j(n) + h_n g_k(n)\mathbf{\Phi}_k(n+1). \tag{126}$$

The coefficients $g_j(n) := g_{j,1}$ are also computed recursively with the formulas

$$g_{j,q} = \begin{cases} \dfrac{1}{q}, & \text{if } j = 1 \\ \dfrac{1}{q(q+1)}, & \text{if } j = 2 \\ g_{j-1,q} - \dfrac{h_n}{x_{n+1} - x_{n+1-j}} g_{j-1,q+1}, & \text{if } j \geq 3. \end{cases}$$

The predictor-corrector scheme based in these formulas are then defined by the formulas

predicted value

$$\mathbf{y}_{n+1}^{[0]} = \mathbf{y}_n^{[m]} + h_n \sum_{j=0}^{k-1} g_j(n)\beta_j(n+1)\mathbf{\Phi}_j(n)^{[m-t]};$$

first corrected value

$$\mathbf{y}_{n+1}^{[1]} = \mathbf{y}_n^{[0]} + h_n g_{k-1}(n)\mathbf{\Phi}_k(n+1)^{[0]};$$

next corrected values

$$\mathbf{\Phi}_k(n+1)^{[\nu]} = \mathbf{\Phi}_k(n+1)^{[\nu-1]} + (\mathbf{f}_{n+1}^{[\nu]} - \mathbf{f}_{n+1}^{[\nu-1]}),$$

$$\mathbf{y}_{n+1}^{[\nu+1]} = \mathbf{y}_{n+1}^{[\nu]} + h_n g_{k-1}(n)(\mathbf{f}_{n+1}^{[\nu]} - \mathbf{f}_{n+1}^{[\nu-1]}), \quad \nu = 1, \ldots, m;$$

estimate of the error

$$\mathbf{est}_{n+1}(k) = h_n(g_k(n) - g_{k-1}(n))\mathbf{\Phi}_k(n+1)^{[m-1]}.$$

8.2.3 BDF formulas with variable coefficients

This important family of methods is generalized to nonuniform grids imposing that the polynomial $P_{kn}(x)$ of degree k that interpolates the values $\mathbf{y}_j, j = n+1, \ldots, n-k+1$ at the abscises $x_{n-k+1}, \ldots, x_{n+1}$ satisfied the ordinary differential equation at $x = x_{n+1}$. The corresponding formula is

$$\sum_{j=1}^{k} \left(\prod_{i=1}^{j-1} (x_{n+1} - x_{n+1-i}) \right) \mathbf{y}[x_{n+1}, \ldots, x_{n-j+1}] = \mathbf{f}_{n+1}. \tag{127}$$

Defining generalized divided differences $\psi_j(n+1)$ of the values $\mathbf{y}_\ell$,

$$\psi_j(n+1) := \left(\prod_{i=1}^{j} (x_{n+1} - x_{n+1-i}) \right) \mathbf{y}[x_{n+1}, \ldots, x_{n-j+1}], \quad j = 0, \ldots, k \tag{128}$$

the formula it is written as

$$\sum_{j=1}^{k} \frac{h_n}{h_n + \cdots + h_{n-j+1}} \psi_j(n+1) = h_n \mathbf{f}_{n+1}. \tag{129}$$

Better than to use the Adams-Bashforth formula as a predictor is to extrapolate the initial iteration $\mathbf{y}_{n+1}^{[0]}$ as the value at $x = x_{n+1}$ of the polynomial of degree k that interpolates the data $\mathbf{y}_{n-k}, \ldots, \mathbf{y}_n$. In this way we have

$$\mathbf{y}_{n+1}^{[0]} = \mathbf{y}_n + \sum_{j=1}^{k} \beta_j(n+1) \psi_j(n), \tag{130}$$

with the coefficients $\beta_j(n+1)$ as in the Adams formulas above.

9 Numerical Methods for Delay Differential Equations

9.1 Delay Differential Equations

Delay differential equations (DDEs) arise in many areas of mathematical modelling: for example, population dynamics (taking in account the gestation times), infectious diseases (accounting for incubation periods), chemical kinetics (such as mixing reactans), etc... The major difference between DDEs and ODEs is the presence in the model of memory terms that refer to delayed solution values. There is inherent qualitative differences between solutions of DDEs and ODEs (oscillatory solutions, onset of chaos, stability of solutions) that made the presence of lags or delays in the system of fundamental interest.

We first introduce the scalar DDEs

$$y'(x) = f(x, y(x), y(x-\tau)), \quad x \geq x_0, \tag{131}$$

where $\tau > 0$ is a constant which we call the lag (or delay). Once we can evaluate the right hand side of equation (131, we can reduce the problem of existence of solutions for (131) to the case or ordinary differential equations. Therefore, if

$$y(x) = \phi(x), \quad x_0 - \tau \leq x \leq x_0, \tag{132}$$

is given, the function $y(x-\tau)$ is known in $[x_0, x_0+\tau]$ and equation (131) becomes an ordinary differential equation in $[x_0, x_0+\tau]$ that can be solved for $y(x)$. Then, we can evaluate the right hand side of (131) and to solve the delay differential equations for $y(x)$ in $[x_0+\tau, x_0+2\tau]$, and so on. This is called the method of steps. Alternatively, we can write down the system of ordinary differential equations

$$\begin{aligned} y_1'(x) &= f(x, y_1(x), \phi(x-\tau)), \quad \sigma_0 \le x \le \sigma_1, \\ y_2'(x) &= f(x, y_2(x), y_1(x-\tau)), \quad \sigma_1 \le x \le \sigma_2, \\ &\vdots \\ y_m'(x) &= f(x, y_m(x), y_{m-1}(x-\tau)), \quad \sigma_{m-1} \le x \le \sigma_m, \end{aligned}$$

where $\sigma_l = x_0 + l\tau$, l integer.

The method of steps can be extended to treat more general DDEs with a non-vanishing variable lag

$$y'(x) = f(x, y(x), y(x - \tau(x, y(x)))), \quad x \ge x_0. \tag{133}$$

The lag function $\tau(x, y(x)) \ge 0$ is called state-dependent if it depends on values of the solution $y(x)$. Now the solution $y(x)$ is required to satisfy an initial condition

$$y(x) = \phi(x), \quad x \in [\min_{s \ge x_0}(s - \tau(s, y(s))), x_0]. \tag{134}$$

A general form for a system of DDEs is

$$\mathbf{y}'(x) = \mathbf{F}(x, \mathbf{y}(x), \mathbf{y}(\alpha_1(x, \mathbf{y}(x))), \ldots, \mathbf{y}(\alpha_k(x, \mathbf{y}(x)), \quad x \ge x_0, \tag{135}$$

where the delay functions $\alpha_\ell(x, \mathbf{y}(x)) equiv x - \tau_\ell(x, \mathbf{y}(x)) leq x$, for $1 \le \ell \le k$ the nonnegative lags $\tau(x, \mathbf{y}(x))$ being possibly state-dependent lag functions.

In the method of steps for this problem we must solve for the ordered set of points $\{\sigma_i\}$, which satisfy

$$\sigma_j - \alpha(\sigma_k, y(\sigma_k)) = 0, \quad \text{for some } \sigma_k > \sigma_j \text{ with } \sigma_0 = x_0. \tag{136}$$

The method of steps suffers a limitation in the case that $\tau(x) \to 0$ as $x \to x^*$, a situation that it is usually described with the term of vanishing lag. In this situation $x = x^*$ is a natural barrier beyonds which the method of steps cannot proceed.

9.2 Derivative discontinuities

Delay differential equations frequently display simple properties not present in ordinary differential equations. If the solution $y(x)$ of (131) is to be continuous in all its derivatives at the initial point $x = x_0$, then it is clear that the left hand derivatives $\phi^{(r)}(x_0-)$ -defined by the initial function- and the right hand derivatives $y^{(r)}(x_0+)$ -define by the differential equation- must all agree. If this condition is not satisfied, then the solution will have a jump discontinuity in some derivative at the initial point that will be passed on to subsequent times $x \ge x_0$, through the effect of delayed terms. This is ilustrated by the following equation

$$y'(x) = y(x-1), \quad x \ge 0, \quad y(x) = 1, \quad x \in [-1, 0). \tag{137}$$

Clearly $y'(x) = 0$ on $[-1, 0)$ but is equal to 1 on $[0, 1)$. Thus $y'(x)$ has a jump discontinuity at $x = 0$. The effect of this discontinuity is then propagated on to the points $x = 1, 2, 3, \ldots$ by the delayed term $y(x-1)$. Consider for example the point $x = k$ where k is a positive integer. Differentiating k-times we obtain

$$y^{(k+1)}(x) = y^{(k)}(x-1),$$

that by induction implies $y^{(k+1)}(x) = y'(x-k)$, thus showing that $y^{(k+1)}$ has a discontinuity at $x = k$. We say that y has a $k+1$ th-order derivative discontinuity at the point $t = k$. Derivative discontinuities of this type turn out to be a common feature in delay-differential equations. For problem (137) it is possible the following representation of the solution

$$y(x) = \sum_{n=0}^{[x/\tau]} \frac{(x-(n-1)\tau)^n}{n!},$$

where $[z]$ denotes the integer part of z. The form of the solution indicates also that the solution grows smoother as x increases. This smoothing of derivatives occurs for all DDEs where $x \to \tau(x)$ is monotonic increasing.

For a scalar equation with a variable lag as in (133) it is not difficult to show how discontinuities propagate. If $y'(x)$ undergoes a jump discontinuity at $\sigma_0 = x_0$ and the delay function $\alpha(x, y(x)) = x - \tau(x, y(x))$ traverses the point σ_0, then the jump in the derivative $y'(x)$ at σ_0 is transmitted through the DDE to produce a jump discontinuity in one of the derivatives of $y(x)$ at $x = \sigma_1$. In general, the set of ordered points $\{\sigma_i\}$ to which discontinuities can be propagated are those points $\sigma = \sigma_k$ such that

$$\sigma_j - \alpha(\sigma, y(\sigma)) \quad \sigma > \sigma_j,$$

changes sign for some σ_j. The extension of these ideas to general systems as in (135) was developped by Willé and Baker (1992). In a system of DDEs discontinuities tracking can be complicated by discontinuities being propagated between solution components. This fact gives rise to the concept of strong and weak coupling and network dependency graphs. Strong coupling describes the propagation of discontinuities between different solution components by an ODE term, that is to say $\tau(x) \equiv 0$. Weak coupling describes the propagation of discontinuities within the same solution component and between different solution components by a DDE term. The importace of tracking become apparent when constructing effcient, high-order numerical methods for solving DDEs. As an example, we consider the DDE system

$$y_1'(x) = y_1(x)y_2(x-1), \quad y_2'(x) = y_2(x^2) - y_3(y_1(x)), \quad y_3'(x) = y_2(x-3).$$

There is a strong coupling from component y_1 to component y_2, and from component y_3 to component y_2 and weak couplings from component y_2 to componen y_1, from component y_2 to component y_3 and finally from component $y2$ to itself.

10 Numerical Issues for solving DDEs

Most of the codes for solving DDEs systems base on a robust ODE solver with an additional dense-output algorithm (for evaluating delayed solution values).

ODE solvers are not only convenient when treating a DDEs system as an ODE system using the method of steps, but also allow to take advantage of all the numerical knowledge adquired over years of research on the numerical solution of ordinary differential equations.

10.1 Dense output

Dense output has interest *per se* also in the numerical solution of ODES:

1. To provide to the codes of ODE the capabilities of dense output without degrading the performance of the method (for example, for plotting). After forcing a very small step size to hit a node where output it is desired, codes might need several additional steps until they rise again the step size until the optimal for advancing the integration of the problem.

2. To make posible to the codes to find the zeroes of some component of the solution, as for example in continuation algorithms.

One of the earliest codes for DDEs was based upon an RK formula combined with cubic Hermite interpolation. When advancing from x_n to x_{n+1} with an explicit RK method we have to our disposal numerical approximations to the value and to the slope of the local solution $\mathbf{u}(x)$ at $x = x_n$. Also, after finishing the step and if this is accepted, we find an approximation to the value $\mathbf{u}(x_{n+1})$. Interpolation of these data appear as a natural approach but as the value of the slope $\mathbf{f}(x_{n+1}, \mathbf{y}_{n+1})$ will be needed in the next step we can consider his computation in the actual step. Therefore we can construct a C^1-piecewise cubic Hermite interpolatory polynomial that it is of order 3.

A natural approach is also provided by collocation formulae (and hence of RK collocation formulae), that provide the collocation polynomial itself for generating approximations inside the step length.

10.1.1 Continuous embedded extensions of RK methods

RK formulae have now been supplemented by continuous RK formulae that correspond to Burcher's array of the form

$$\begin{array}{c|cccc} c_1 & a_{11} & & & a_{1s} \\ c_2 & a_{21} & 0 & & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline \theta & b_1(\theta) & b_2(\theta) & \cdots & b_s(\theta) \end{array} \tag{138}$$

The most interesting property of the Butcher's array for these RK formulae is that the coefficients a_{ij} of the method does not depend on θ. Therefore, with this formula it is possible to generate numerical approximations at the intermediate abscises $x = x_n + \theta h_n, 0 < \theta \leq 1$ of the interval $[x_n, x_{n+1}]$ without any additional evaluation of $\mathbf{f}$. The Euler method

$$\mathbf{y}(x_n + \theta h) \approx \mathbf{y}_n + \theta h \mathbf{f}(x_n, \mathbf{y}_n)$$

provides a first example of an embedded continuous RK method. Another example of a third order RK method with an embedded continuous extension is provided by

0			
$\frac{1}{2}$	$\frac{1}{2}$		
1	-1	2	
	$\theta(1+\theta(-3/2+2/3\theta))$	$\theta^2(2-2/3\theta)$	$\theta^2(2/3-\theta/2)$

Embedded continuous extensions have been developed by Horn (1983) and Enright, Jackson, Norsett y Thomsen (1985). For example, Horn found for the embedded pair Runge-Kutta-Fehlberg 4(5) an embedded continuous extension or four order at the cost of one additional evaluation of $\mathbf{f}$. For the embedded pair of Dormand and Prince is possible a continuous extension of fourth order without any additional evaluation of $\mathbf{f}$. It is given by

$$\begin{aligned}
b_1(\theta) &= \theta(1+\theta(-1337/480+\theta(1039/360+\theta(-1163/1152)))),\\
b_2(\theta) &= 0,\\
b_3(\theta) &= 100\theta^2(1054/9275+\theta(-4682/27825+\theta(379/5565)))/3,\\
b_4(\theta) &= -5\theta^2(27/40+\theta(-9/5+\theta(83/96)))/2,\\
b_5(\theta) &= 18225\theta^2(-3/250+\theta(22/375+\theta(-37/600)))/848,\\
b_6(\theta) &= -22\theta^2(-3/10+\theta(29/30+\theta(-17/24)))/7,
\end{aligned}$$

and

$$\mathbf{y}(x_n+\theta h) \approx \mathbf{y}_n + h\sum_{i=1}^{6} b_i(\theta)\mathbf{k}_i$$

It is important to stress that the order, the asymptotic form of the error and the stability of the DDEs formula depend not only on the underlying ODE formulae but also on the dense-output formulae.

Theorem 10.1 *(Order of Convergence) Given an ODE method of order p combined with an interpolant of order q, if discontinuities not exceeding order r only occur at meshpoints then the order of the resulting DDE method is* $\min(p,q,r)$.