Alfio Quarteroni

Complex Systems in Biomedicine

Numerical Methods for Delay Models in Biomathematics
by A. Bellen, N. Guglielmi, S. Maset

December 29, 2005

Springer
Berlin Heidelberg New York
Hong Kong London
Milan Paris Tokyo

1

Numerical Methods for Delay Models in Biomathematics

1.1 Introduction

Models involving retarded ordinary and partial differential equations with both discrete and distributed delays are quite frequent in mathematical biology. Introducing delays in the models have revealed to be a powerful tool for investigating qualitative behaviour of control systems and, in general, for simulating the evolution phenomena in many branches of medicine and biology. Introduction of delays allowed to improve models by taking into account important aspects previously neglected and to face more complicated phenomena based on feedback control. So, for instance, in a more realistic model for the spread of infections in large scale epidemics, a delay term may take into account the incubation period in the transmission of diseases via contacts among individuals. It was shown that arising of periodic hematological diseases can be caused by anormalities in the feedback mechanisme which regulate blood-cell numbers and, under appropriate conditions, this feedback mechanism can produce aperiodic irregular (chaotic) fluctuations. Recently, in the interaction analysis between cardiovascular and respiratory function, clever models have been considered that take into account the time necessary for tissue venous blood to reach the lungs and viceversa. In biomathematical literature there are plenty examples where the presence of delays makes the mathematical models much more reliable and consistent with the real phenomena and the laboratory observations. Actually the dynamics of equations including retarded arguments is much richer and this makes the models more realistic for simulation. At the same time, the equations with retarded arguments become more and more complicated to be analyzed and the existence and uniqueness of the solution as well as important features such as oscillation and asymptotic behaviour are still open problems in many cases. Finding accurate numerical solutions and sharp location of characteristic roots for establishing stability was also getting more and more difficult and, in some cases, still represent a real challenge for the numerical analysts. A careless and naif adaptation of standard numerical methods designed for ordinary and partial differential equations in the integration of equations with delays, behind revealing often useless, might lead and actually led, as claimed and proved in Banks and Mahaffy [2], to conjectures that turned out to

be erroneous and misleading for the understanding of the studied phenomenon. Algorithms for implementing delay models must be specifically designed according to the nature of the equations and the quality of the solution.

Here we confine to the case of ordinary derivatives, where the most general form of such models is given by the *retarded functional differential equation* (in short RFDE)

$$y'(t) = f(t, y_t), \quad t \geq t_0,$$

where the *state* $y_t(s) = y(t+s)$, $s \in [-r, 0]$, is a function belonging to the Banach space $C = C^0([-r,0], R^d)$ of continuous functions mapping the interval $[-r,0]$ into R^d, and $f : \Omega \longrightarrow R^d$ is a given function of the set $\Omega \subset R \times C$ into R^d. Contrary to the ordinary case, the Cauchy problem takes the form

$$\begin{cases} y'(t) = f(t, y_t), & t \geq t_0, \\ y_{t_0}(s) = y(t_0 + s) = \varphi(s), & s \in [-r, 0] \end{cases} \tag{1.1}$$

where φ represents, in the Banach space C, the *initial point* or the *initial state*. Equation 1.1, also called the *Volterra functional differential equation*, includes both *distributed delay differential equations*, where f operates on y computed on a continuum set of past values, and *discrete delay differential equations*, where only a finite number of past values of the variable y are involved.

Models with discrete delays are characterized by the presence of the function y computed at some deviated arguments $y(t-\tau)$, where the delay τ, which is always non negative, may be *constant* ($\tau = const$), *time dependent* ($\tau = \tau(t)$), and *state dependent* ($\tau = \tau(t, y(t))$ or even $\tau = \tau(t, y_t)$).

A real-life example of retarded functional differential equations with both discrete and distributed delays is given by the model of Banks and Mahaffy [2] for the regulation of protein synthesis:

$$\begin{aligned}
\tfrac{d}{dt}x^1(t) &= \frac{a_1}{1 + k_1 L_1^1(z_t^n)} - b_1 x^1(t) \\
\tfrac{d}{dt}y^1(t) &= \alpha_1 L_1^2(x_t^1) - \beta_1 y^1(t) \\
\tfrac{d}{dt}z^1(t) &= \gamma_1 L_1^3(y_t^1) - \delta_1 z^1(t) \\
\left.\begin{aligned}
\tfrac{d}{dt}x^i(t) &= \frac{a_i}{1 + k_i L_i^1(z_t^{i-1})} - b_i x^i(t) \\
\tfrac{d}{dt}y^i(t) &= \alpha_i L_i^2(x_t^i) - \beta_i y^i(t) \\
\tfrac{d}{dt}z^i(t) &= \gamma_i L_i^3(y_t^i) - \delta_i z^i(t)
\end{aligned}\right\} & i = 2, \cdots, n
\end{aligned}$$

where the operators L_i^j's are given by

$$L_i^j(z_t) = \sum_{l=0}^{\nu} c_{il}^j z(t - h_l) + \int_{-r}^{0} z(t+s)\zeta_i^j(s)ds,$$

x^i is the amount of mRNA (messenger Ribo-Nucleic Acid) by the transcription of gene i, y^i the amount of protein by the translation of x^i and z^i is the repressor produced by the protein y^i which shut down transcription in the gene $i+1$, etc.

The presence of an initial function φ, instead of an initial value y_0 in the Cauchy problem (1.1), entails some and often unexpected consequences in the solution $y(t)$ for $t > t_0$. In general, there is no longer injectivity between the set of initial data φ and the set of solutions $y(t)$. Moreover, the prolongation of the initial function φ over the initial point t_0 is not smooth whenever $\varphi'(0)^- \neq y'(t_0)^+ = f\left(t_0, y_{t_0}\right)$ and this lack of regularity, at t_0, propagates forward even if the ingredients f, τ and φ of the problem belong to C^∞.

For example, it is not difficult to see that, in general, the solution of the Cauchy problem

$$\begin{cases} y'(t) = f(t, y(t-\tau(t))), & t \geq t_0, \\ y(t) = \phi(t) := \varphi(t-t_0), & t \leq t_0 \end{cases} \tag{1.2}$$

does not possess the second derivative at any point $\xi_{1,i}$ such that

$$\xi_{1,i} - \tau(\xi_{1,i}) = t_0,$$

it does not possess the third derivative at any point $\xi_{2,j}$ such that

$$\xi_{2,j} - \tau(\xi_{2,j}) = \xi_{1,i}$$

for some i, and so on for higher order derivatives. This results in a sequence of points, called *breaking points* or *primary discontinuities*, where the solution possesses only a limited number of derivatives, the *order* of the breaking point, and remains piecewise regular between two consecutive of them. Locating the breaking points and including them into the mesh is a crucial issue in the numerical integration of RFDE in what any step-by-step method attains its own accuracy order provided that the sought solution is sufficiently smooth in the current integration interval. In principle, the breaking points can be computed by recursively solving the algebraic equations above, which are trivial for constant delays and may be solved "a priori" for time dependent delays. On the contrary, in the state dependent delay case where the algebraic equation is

$$\xi_{2,j} - \tau(\xi_{2,j}, y(\xi_{2,j})) = \xi_{1,i},$$

they depend on the solution and can not be computed in advance. In particular, their accurate computation depends on the accuracy of the approximation of y, which in turns depends on the accuracy in the computation of the breaking point itself. This makes accurate integration of RFDEs with state dependend delay a real challenge.

Other discontinuities, called *secondary discontinuities*, may propagate along the solution caused by discontinuities in the functions f, τ or ϕ. In particular, discontinuities in the initial function ϕ may be much more dreadful and cause the termination of the solution, that may reappear some time later and disappear again giving rise to *lacunary solutions*. It is not difficult to guess that introduction of delays will also upset the well-posedness of the problem and the others stability and asymptotic stability properties of the solution as well.

The general theory of RFDEs has been widely developed in the last four-five decades and results in a number of, now classic, books by Bellman and Cooke, Hale, Driver, El'sgol'ts and Norkin, Kolmanovskii and Myshkis, up to the more recent monographies by Diekmann, van Gils, Verduyn-Lunel and Walter [14] and Kuang [30], which also include many real-life examples of RFDEs and more general retarded functional differential equations.

From the numerical point of view, the presence of delays entails various additional difficulties which have been tackled by various different approaches. The choice of the approach specifically depends on the particular kind of delay one must handle and the particular aim is pursued in terms of accuracy, stability etc. These methods range from the classical method of steps, where the RFDE is seen as a sequence of ordinary differential equations, to the use of collocation and more general continuous Runge-Kutta methods leading to piecewise polynomial approximations, or to the transformation of the delay equation into a partial differential equations with appropriate initial/boundary conditions to be integrated by direct or transversed methods of lines. All these methods, along with some preliminaries and historical remarks on the numerics of delay differential equations, are described and developed in the recent book by Bellen and Zennaro [7] with particular emphasis to the class of continuous Runge-Kutta methods and their numerical stability.

Although models based on partial differential equations with delays for investigating complex biological phenomena started to be considered more than twenty years ago, they have received very little attention by numerical analysts and, to our knowledge, no public domain code is available for their integration yet. Developing algorithms for the numerical integration of partial differential equations with delays appears a very promising common ground of research for numerical ODE and PDE communities and reveals an attractive and challenging area of investigation that, in our opinion, will play a central role for the research in biomathematics during the next decade. For an overview of retarded partial differential equations, see Wu [38].

1.2 Solving RFDE by Continuous Runge-Kutta methods

Here we will present two different approaches for solving RFDEs (1.1) in the equivalent form

$$\begin{cases} y'(t) = f\big(t, y(t), y_t\big), & t \geq t_0, \\ y_{t_0} = y(t_0 + s) = \varphi(s), & s \in [-r, 0] \end{cases} \tag{1.3}$$

where we have outlined the possible dependence of f on the state function y_t and on the current value $y(t)$ as well. The methods are both based on the use of Continuous Runge-Kutta methods (CRK) as applied, step by step, to the local problems

$$\begin{cases} w'(t) = f\big(t, w(t), w_t\big), & t_n \leq t \leq t_{n+1}, \\ w(t) = \eta(t), & t_0 \leq t \leq t_n \\ w(t) = \phi(t), & t \leq t_0 \end{cases} \tag{1.4}$$

where η is the continuous approximate solution provided by the CRK method itself. The straightforward application of the CRK methods $(A, b(\theta), c)$ to the local

equation (1.4) takes the form

$$\begin{aligned} &\eta(t_n+\theta h_{n+1}) = y_n + h_{n+1}\textstyle\sum_{i=1}^{\nu} b_i(\theta) f\big(t_{n+1}^i, Y^i, Y^i_{t_{n+1}^i}\big), \quad 0\le\theta\le 1,\\ &Y^i = y_n + h_{n+1}\textstyle\sum_{j=1}^{\nu} a_{ij} f\big(t_{n+1}^j, Y^j, Y^j_{t_{n+1}^j}\big), \quad i=1,\dots,\nu, \end{aligned} \tag{1.5}$$

where $t_{n+1}^i = t_n + c_i h_{n+1}$ and the state functions $Y^i_{t_{n+1}^i}$ are suitable approximations of $w_{t_{n+1}^i}$. For sake of conciseness we omit the dependence of Y^i on n.

It is evident that, as long as all the state functions $Y^i_{t_{n+1}^i}(s) = Y^i(t_{n+1}^i + s)$, $s\in[-r,0]$, have to be computed, according to the action of the functional f, only at arguments s such that $t_{n+1}^i + s \le t_n$, we must set $Y^i(t_{n+1}^i+s) = \eta(t_{n+1}^i+s)$ and the equation (1.4) is essentially an ODE. On the contrary, when for some i and for some s, $t_{n+1}^i + s > t_n$, the unknown part of the state function $Y^i_{t_{n+1}^i}(s)$ has to be provided by suitable extensions of the CRK method itself. In this case, referred to as *overlapping*, the structure of the Runge-Kutta equations will change and the problem is intrinsically different from an ODE.

A central issue in the convergence analysis of the step-by-step method for RFDEs is how its discrete and uniform global errors depend on the local discrete error and the local uniform error of the method (1.5)(see Bellen and Zennaro [7] for definitions). The convergence of the CRK methods for RFDEs is governed by the following Theorem proved in [7].

Theorem 1.1. *Given the mesh $\mathscr{M}_h$, of maximum stepsize h, assume that all the breaking points of order $p+1$ are included in $\mathscr{M}_h$, so as the solution $y(t)$ is piecewise of class $C^{p+1}(t_0,t_f)$. If the method (1.5) has discrete order p (i.e. discrete local error of order $p+1$) and uniform order q (i.e. uniform local error of order $q+1$), then the resulting method for RFDEs has discrete and uniform global order $\min\{p,q+1\}$, i.e.*

$$\max_{t_i\in\mathscr{M}_h} \|y(t_i)-\eta(t_i)\| = \mathscr{O}(h^{\min\{p,q+1\}}).$$

$$\max_{t_0\le t\le t_f} \|y(t)-\eta(t)\| = \mathscr{O}(h^{\min\{p,q+1\}}).$$

In other words, the local uniform error does not propagate and, in particular, the local uniform order $q=p-1$ is sufficient for preserving the global order p of the overall method.

1.2.1 Continuous Runge-Kutta (standard approach) and Functional Continuous Runge-Kutta Methods.

Despite the first method we are going to describe can be applied to the general problem (1.3), let us be more specific and consider, as a special case, the following RFDE with discrete state dependent delay

$$\begin{cases} y'(t) = f\Big(t, y(t), y\big(\alpha(t, y(t))\big)\Big), & t_0 \le t \le t_f, \\ y(t) = \phi(t), & t \le t_0, \end{cases}$$

where, for the sake of simplicity, we have set the deviated argument $t - \tau(t, y(t))$ in the form $\alpha(t, y(t))$. One possible option in the implementation of (1.5), referred to as *standard approach*, is characterized by the choice $Y^i(t^i_{n+1} + s) = \eta(t^i_{n+1} + s)$ for all i, which corresponds to setting, in (1.4),

$$w_t = \eta_t, \forall t \in [t_n, t_{n+1}]$$

The local problem is then

$$\begin{cases} w'(t) = f\Big(t, w(t), \eta\big(\alpha(t, w(t))\big)\Big), & t_n \le t \le t_{n+1}, \\ w(t) = \eta(t), & t_0 \le t \le t_n \\ w(t) = \phi(t), & t \le t_0 \end{cases}$$

and the CRK methods is

$$\begin{aligned} &\eta(t_n + \theta h_{n+1}) = y_n + h_{n+1} \textstyle\sum_{i=1}^{\nu} b_i(\theta) f(t^i_{n+1}, Y^i, \tilde{Y}^i), \quad 0 \le \theta \le 1, \\ &Y^i = y_n + h_{n+1} \textstyle\sum_{j=1}^{\nu} a_{ij} f(t^j_{n+1}, Y^j, \tilde{Y}^j), \quad i = 1, \dots, \nu. \\ &\tilde{Y}^i = \eta\left(\alpha(t^i_{n+1}, Y^i)\right). \end{aligned}$$

If overlapping occurs, that is if, for some index i,

$$t_n \le \alpha(t^i_{n+1}, Y^i) \le t_n + c_i h_{n+1},$$

then

$$\alpha(t^i_{n+1}, Y^i) = t_n + \theta^i_{n+1} h_{n+1},$$

with

$$(0 \le)\ \theta^i_{n+1} = \frac{\alpha(t^i_{n+1}, Y^i) - t_n}{h_{n+1}}\ (\le c_i),$$

and the *spurious stage values* $\tilde{Y}^i$ are still given by the continuous extension

$$\tilde{Y}^i = \eta(t_n + \theta^i_{n+1} h_{n+1}) = y_n + h_{n+1} \sum_{j=1}^{\nu} b_j(\theta^i_{n+1}) f(t^j_{n+1}, Y^j, \tilde{Y}^j).$$

Since the deviated function $y(\alpha(t, y(t))$ is approximated by the continuous extension $\eta(\alpha(t, y(t))$ in both current and past integration intervals, the standard approach is usually referred to as the Continuous Runge-Kutta method, simply.

It is worth remaking that if overlapping occurs, the current Runge-Kutta equation turns out to be implicit even if the underlying method is explicit. However, in spite of the appearance of possible spurious stages $\tilde{Y}^i$, the dimension of the algebraic Runge-Kutta system preserves the dimension s by using the alternative $K-$notation

$$\eta(t_n+\theta h_{n+1}) = y_n + h_{n+1}\sum_{i=1}^{\nu} b_i(\theta)K^i, \quad 0\le\theta\le 1,$$

$$K^i = f(t^i_{n+1}, y_n + h_{n+1}\sum_{j=1}^{\nu} a_{ij}K^j, y_n + h_{n+1}\sum_{j=1}^{\nu} b_j(\theta^i_{n+1})K^j), \quad i=1,\dots,\nu,$$

where

$$\theta^i_{n+1} = \frac{\alpha(t^i_{n+1}, y_n + h_{n+1}\sum_{j=1}^{\nu} a_{ij}K^j) - t_n}{h_{n+1}}.$$

Remark that overlapping takes place when the stepsize is larger than the delay as well as, independently of the stepsize, when we are integrating in a neighbourhood of points where the delay vanishes. Therefore the standard approach is suitable for stiff problems where an implicit CRK method is used. However, in presence of overlapping, the structure of the Jacobian of the Newton solver changes and this could lead to additional difficulties in the variable step size implementation of the overall method. The standard approach outlined above will be described in details in Section 1.3 as applied to a specific biological model leading to a system of stiff RFDEs with discrete vanishing state dependent delays that sums up, in its numerical integration, some of the theoretical and practical difficulties described above.

As counterpart to the standard approach let us consider a second method designed for the general equation (1.3), including RFDEs with distributed delays, where the local problem (1.4) is still approximated by the CRK method (1.5) but, for each i, the unknown part of the state function $Y^i_{t^i_{n+1}}$ is now given by

$$Y^i(t_n+\theta h_{n+1}) = y_n + h_{n+1}\sum_{j=1}^{\nu} a_{ij}(\theta) f(t^j_{n+1}, Y^j, Y^j_{t^j_{n+1}}), \ \ 0\le\theta\le c_i. \tag{1.6}$$

This approach, called *Functional Continuous Runge-Kutta* (FCRK) method, beside being still based on CRK methods, is quite different from the standard approach. In particular, as the RK method makes use of ν stage-values Y^i, $i=1,\dots\nu$, as approximations of $w(t^i_{n+1})$, ν different *state stage-functions* $Y^i_{t^i_{n+1}}$ are defined which approximate the state functions $w_{t^i_{n+1}}$.

Remark 1.2. Contrary to the standard approach, the resulting method preserves the implicit/explicit character of the undelying CRK method also in case of overlapping. This makes the FCRK method competitive for non stiff equation with small or vanishing delays, leading to possible overlapping, as well as for functional integral equations such as

$$y'(t) = F\left(t, y(t), \int_{t-\tau}^{t} k(t,s,y(s))ds\right),$$

where overlapping takes place at every step.

The method, described in Section 1.4 in its general form, was presented by Cryer and Tavernini [13], [35] as a particular predictor-corrector version of polynomial collocation. The method was recently reconsidered by Maset,Torelli and Vermiglio [32] who developed it in the general form (1.4) and derived the necessary and sufficient order conditions up to order four, along with some order barrier with respect to the number of stages.

1.3 A threshold model for antibody production: the Waltman model

We consider a mathematical model describing the mechanism by which an antibody is formed in response to an antigen challenge. This is one of the better understood parts of the human immune system and has been widely treated in the scientific literature. The earliest models were chemical kinetics or predator prey models (see Bell [3, 4, 5]); afterwards Hoffmann [25] and Richter [34] proposed network-based models including inhibiting and stimulating kind of signals. We consider here a more sophisticated and widely shared threshold model which makes use of an integral threshold to describe the onset of B-cell proliferation and to mark the signal which activates the antibody production. Such kind of approch has been extensively considered in the literature (see e.g. Gatica and Waltman [18, 19, 20], Waltman and Butz [37], Waltman [36], Cooke [12], Hoppensteadt and Waltman [29]).

These threshold models, which are still valid and topical (see e.g. [26, 31]) are complicate to treat mathematically and have the peculiarity to lead to functional differential equations rather than to ordinary differential equations.

The framework we consider describes a realistic although simplified situation describing the challenge of a chemical antigen which binds to receptor sites on the surface of lymphocytes. This determines the activation of a signal which gives rise to the lymphocyte production phase; the nature of this triggering mechanism is still not exactly known to immunologists and the model aims to represent it in a very general way.

In the first phase, the one preceding the onset of lymphocyte proliferation, the dynamics describing the interaction among free antigen molecules, free receptor sites and bound receptor sites is described by a *chemical reaction*-like system of *stiff* ordinary differential equations $y'(t) = f_a(y(t))$.

In the second phase, which is established through a first threshold effect modelled by an integral equation which depends on the concentration of bound receptor sites, the model changes form and is described by a system of delay differential equations of the type $y'(t) = f_b\Big(y(t), y\big(\alpha_1(t, y(t))\big)\Big)$ where y denotes the vector of unknown concentrations of antigen molecules and receptor sites and $\alpha_1(t, y(t)) \leq t$ is a deviating argument which allows to model the memory effect of the phenomenon. Such deviating argument depends on the solution y itself (the so-called state of the system) and its dynamics is also described by a suitable functional differential equations.

Finally, in the third phase, which is established through a second threshold effect, the model is described by a larger system of delay differential equations which includes the proliferation of antibodies. A further memory effect is described by a second deviating argument so that the whole system is described by a system of six delay differential equations; four equations describe the dynamics of the concentrations of antigen molecules, free and bound receptor sites on the surface of lymphocytes and antibodies. They have the form $y'(t) = f_c\Big(y(t), y\big(\alpha_1(t, y(t))\big), y\big(\alpha_2(t, y(t))\big)\Big)$, where $\alpha_2(t, y(t)) \leq t$ denotes the second deviating argument modelling the memory effect in the antibodies production process. The remaining two equations describe the dy-

namics of the deviating arguments α_1 and α_2. As time increases the memory effects tend to reduce or also to disappear in the model which means that $\alpha_1(t,y(t))$ and $\alpha_2(t,y(t))$ approach t.

1.3.1 The quantitative model

The whole model is an interesting system of stiff delay differential equations.

To describe it mathematically, we introduce the following quantities:

(1) $y_1(t)$ the concentration of unbound antigen molecules at time t;
(2) $y_2(t)$ the concentration of unbound receptor sites at time t;
(3) $y_3(t)$ the concentration of bound receptor sites at time t;
(4) $y_4(t)$ the concentration of unbound antibodies at time t.

Initial phase.

The antigen molecules and the receptor sites combine according to the mass action law,

$$\begin{aligned} y_1'(t) &= -r_1 y_1(t) y_2(t) + r_2 y_3(t) \\ y_2'(t) &= -r_1 y_1(t) y_2(t) + r_2 y_3(t) \\ y_3'(t) &= r_1 y_1(t) y_2(t) - r_2 y_3(t), \end{aligned}$$

with r_1, r_2 denoting suitable rate constants.

This model holds until the time t_0 when the trigger to initiate lymphocytes proliferation begins. The model assumes that t_0 is given by the integral equation

$$\int_0^{t_0} f_1\left(y_1(s), y_2(s), y_3(s)\right) ds = m_1,$$

being m_1 an appropriate biological threshold. The previous integral models the accumulation of signals depending on the concentration of free antigen molecules, free receptor sites and receptor-antigen complexes.

Intermediate phase.

In this phase new receptor sites are generated so that the system evolves according the following equations,

$$\begin{aligned} y_1'(t) &= -r_1 y_1(t) y_2(t) + r_2 y_3(t) \\ y_2'(t) &= -r_1 y_1(t) y_2(t) + r_2 y_3(t) + a\, r_1 y_1\left(\alpha_1(t)\right) y_2\left(\alpha_1(t)\right) \\ y_3'(t) &= r_1 y_1(t) y_2(t) - r_2 y_3(t), \end{aligned}$$

where a is an amplification factor and $\alpha_1(t) \leq t$ models a memory effect described by the integral equation

$$\int_{\alpha_1(t)}^{t} f_1\left(y_1(s),y_2(s),y_3(s)\right)\,ds = m_1, \qquad t \geq t_0. \tag{1.7}$$

This model holds until a certain time t_1 when the trigger to initiate antibodies production begins. The model assumes that t_1 is given by the integral equation

$$\int_{0}^{t_1} f_2\left(y_2(s),y_3(s)\right)\,ds = m_2,$$

being m_2 an appropriate biological threshold.

Final phase.

In this phase antibodies (y_4) are produced by the immune system according to the following equations,

$$\begin{aligned}
y_1'(t) &= -r_1 y_1(t) y_2(t) + r_2 y_3(t) - s y_1(t) y_4(t)\\
y_2'(t) &= -r_1 y_1(t) y_2(t) + r_2 y_3(t) + a r_1 y_1\left(\alpha_1(t)\right) y_2\left(\alpha_1(t)\right)\\
y_3'(t) &= r_1 y_1(t) y_2(t) - r_2 y_3(t),\\
y_4'(t) &= -s y_1(t) y_4(t) - \gamma y_4(t) + b r_1 y_1\left(\alpha_2(t)\right) y_2\left(\alpha_2(t)\right)
\end{aligned}$$

where s is a combination factor, b is an amplification factor related to antibody secretion capacity of plasma cells, γ is a catabolic factor and $\alpha_2(t) \leq t$ is a second memory effect described by the integral equation

$$\int_{\alpha_2(t)}^{t} f_2\left(y_2(s),y_3(s)\right)\,ds = m_2, \qquad t \geq t_1. \tag{1.8}$$

Summary

The whole problem consists of six equations; four to describe the interaction between the antigen and the immune system,

$$\begin{cases}
y_1'(t) = -r_1 y_1(t) y_2(t) + r_2 y_3(t) - s y_1(t) y_4(t)\\
y_2'(t) = -r_1 y_1(t) y_2(t) + r_2 y_3(t) + a r_1 y_1\big(\alpha_1(t)\big) y_2\big(\alpha_1(t)\big) H(t-t_0)\\
y_3'(t) = r_1 y_1(t) y_2(t) - r_2 y_3(t)\\
y_4'(t) = -s y_1(t) y_4(t) - \gamma y_4(t) + b r_1 y_1\big(\alpha_2(t)\big) y_2\big(\alpha_2(t)\big) H(t-t_1)
\end{cases} \tag{1.9}$$

and two to describe the dynamics of the deviating arguments α_1 and α_2, which are obtained by differentiating equations (1.7) and (1.8),

$$\begin{cases}
\alpha_1'(t) = H(t-t_0)\,\dfrac{f_1\left(y_1(t),y_2(t),y_3(t)\right)}{f_1\left(y_1\left(\alpha_1(t)\right),y_2\left(\alpha_1(t)\right),y_3\left(\alpha_1(t)\right)\right)}\\[2ex]
\alpha_2'(t) = H(t-t_1)\,\dfrac{f_2\left(y_2(t),y_3(t)\right)}{f_2\left(y_2\left(\alpha_2(t)\right),y_3\left(\alpha_2(t)\right)\right)}
\end{cases} \tag{1.10}$$

where $H(x)$ is the Heavyside function ($H(x) = 0$ if $x < 0$ and $H(x) = 1$ if $x \geq 0$).

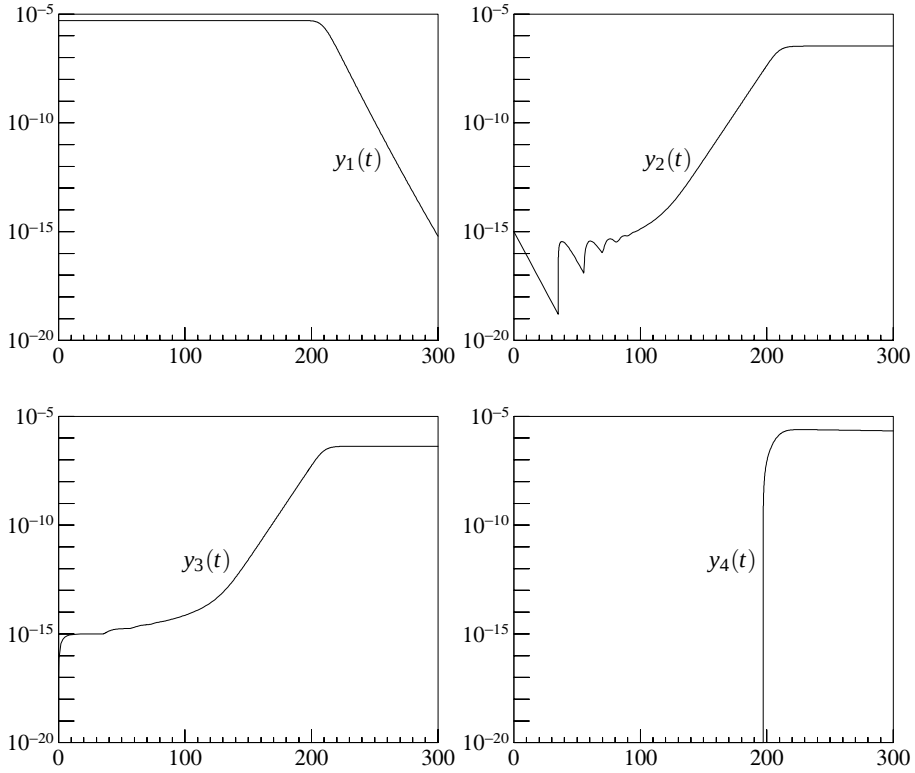

Fig. 1.1. Solution components of the problem (1.9) with the choice of parameters given in Sect 1.3.6.

Numerical integration

From the numerical point of view the model presents the following difficulties (see also Figures 1.1 and 1.2):

(i) the deviating arguments are state-dependent and hence are not known in advance;
(ii) the delays $t-\alpha_1(t,y(t))$ and $t-\alpha_2(t,y(t))$ become very small as time grows; this prevents the possibility of considering the problem step-by-step as a system of ordinary differential equations;
(iii) the solution components have very different magnitudes and have very steep variations in correspondence of the triggers initiating the proliferation first of lymphocytes and later of antibodies;
(iv) the presence of discontinuities in the right-hand side, due to the threshold mechanisms, determines a certain number of breaking points, which have to be treated carefully in order to avoid a loss of accuracy;
(v) the system is *stiff* and therefore needs to be integrated numerically by an implicit method.

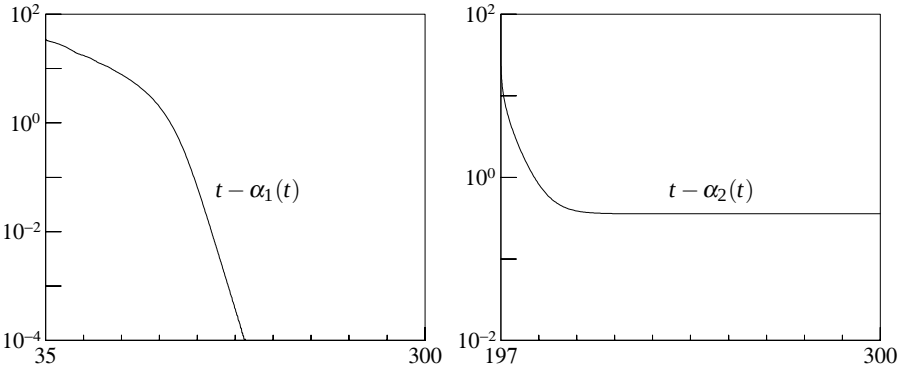

Fig. 1.2. Delays of the problem (1.9) with the choice of parameters given in Sect 1.3.6.

We have numerically integrated the problem by means of the code RADAR5, whose main features are described here. We shall discuss the application of stiffly accurate collocation methods based on Radau nodes to systems of delay differential equations of the form

$$y'(t) = f\Big(t, y(t), y\big(\alpha_1(t, y(t))\big), \ldots, y\big(\alpha_p(t, y(t))\big)\Big) \tag{1.11}$$

with initial data

$$y(t_0) = y_0, \qquad y(t) = g(t) \quad \text{for } t < t_0.$$

We assume that the deviating arguments are such that $\alpha_i(t, y(t)) \leq t$ for all $t \geq t_0$ and for all i.

As we have mentioned the integration of delay differential equations presents several additional difficulties with respect to ODEs.

In particular, discontinuities may occur in various orders of the derivative of the solution, independently of the regularity of the right-hand side; this could lead to a loss in the accuracy of the numerical approximation. Small delays complicate the use of explicit approximation methods and determine a structural change in the Runge–Kutta equations of implicit methods; on the other hand large delays force to store a large amount of information (the solution in the past). Furthermore error control strategies for ODEs may be inappropriate for delay differential equations since the continuous output has also to be controlled. For a comprehensive discussion on these issues we refer the reader to [7].

The remainder of this section is organized as follows. First we describe the integration process. Then we describe a technique to compute breaking points which is peculiar to implicit methods. Afterwards we direct our attention at the Newton process for the solution of the Runge–Kutta equations associated to each step. Finally we deal with an error control technique which is well-suited to stiff delay equations. The last section illustrates the application of the code RADAR5 to the considered Waltman model.

1.3.2 The integration process

We direct our attention here to stiff equations and more generally to problems where the use of an explicit method would lead to stepsize restrictions which are not due to accuracy requirements and can be overcome by using an implicit method.

The integration scheme we consider is based on the ν-stage Radau IIA collocation method (in particular the code RADAR5 uses $\nu = 3$). For a detailed description we address the reader to [21], [22] and [23]. We use the following notation:

- $f(t, y, z_1, \ldots, z_p)$ denotes the right hand side function;
- the nodes $\{c_i\}$, the weights $\{b_i\}$ and the coefficients $\{a_{ij}\}$ are those of the Radau IIA method; in the case $\nu = 3$ its Butcher tableau is given by

$$\begin{array}{c|ccc} \frac{4-\sqrt{6}}{10} & \frac{88-7\sqrt{6}}{360} & \frac{296-169\sqrt{6}}{1800} & \frac{-2+3\sqrt{6}}{225} \\ \frac{4+\sqrt{6}}{10} & \frac{296+169\sqrt{6}}{1800} & \frac{88+7\sqrt{6}}{360} & \frac{-2-3\sqrt{6}}{225} \\ 1 & \frac{16-\sqrt{6}}{36} & \frac{16+\sqrt{6}}{36} & \frac{1}{9} \\ \hline & \frac{16-\sqrt{6}}{36} & \frac{16+\sqrt{6}}{36} & \frac{1}{9} \end{array};$$

- Y^j denotes the j-th stage value computed at the current step;
- $h_{n+1} = t_{n+1} - t_n$ indicates the current stepsize;
- $\alpha_{\ell j} := \alpha_\ell(t_n + c_j h_{n+1}, Y^j)$ yields an approximation of $\alpha_\ell(t_n + c_j h_{n+1}, y(t_n + c_j h_{n+1}))$;
- $\eta(t)$ is the piecewise polynomial continuous approximation to the solution. In most cases it is given step-by-step by the collocation polynomial associated to the method.

The Runge–Kutta formula applies to (1.11) as

$$\begin{cases} 0 = F_i\left(Y^1, \cdots, Y^\nu, Z^{11}, \cdots, Z^{1\nu}, \ldots, Z^{p1}, \cdots, Z^{p\nu}\right),\ i = 1, \ldots, \nu \\ \\ y_{n+1} = Y^\nu, \end{cases} \tag{1.12}$$

with

$$F_i(\ldots) = Y^i - y_n - h_{n+1} \sum_{j=1}^{\nu} a_{ij} f\left(t_n + c_j h, Y^j, Z^{1j}, \cdots, Z^{pj}\right), \tag{1.13}$$

and

$$Z^{\ell j} = \begin{cases} g\left(\alpha_{\ell j}\right) & \text{if } \alpha_{\ell j} \le t_0 \\ \eta\left(\alpha_{\ell j}\right) & \text{if } \alpha_{\ell j} > t_0. \end{cases}$$

Also here we have omitted the dependence of F_i, $\alpha_{\ell j}$ and $Z_{\ell j}$ on n. The continuous approximation η to the solution at the m-th step, that is for $t_m \le t \le t_{m+1}$ ($m \le n$), is given by one of the following polynomials,

$$u_m(t_m+\vartheta h_{m+1}) = \mathscr{L}_0(\vartheta)y_m + \sum_{i=1}^{s} \mathscr{L}_i(\vartheta)Y^i, \qquad \vartheta \in [0,1], \tag{1.14}$$

$$v_m(t_m+\vartheta h_{m+1}) = \sum_{i=1}^{s} \hat{\mathscr{L}}_i(\vartheta)Y^i, \qquad \vartheta \in [0,1]. \tag{1.15}$$

Remark that in (1.14) and (1.15) the stage values Y^i are those relevant to the interval $[t_m, t_{m+1}]$. Here h_{m+1} is the stepsize used at the m-th step, $\mathscr{L}_i(\vartheta)$ is the polynomial of degree ν satisfying $\mathscr{L}_i(c_i) = 1$ and $\mathscr{L}_i(c_j) = 0$ for $j \neq i$ (where $c_0 = 0$ and $c_1, \ldots, c_\nu$ are the nodes of the method) and $\hat{\mathscr{L}}_i$ is the polynomial of degree $\nu - 1$ satisfying $\hat{\mathscr{L}}_i(c_i) = 1$ and $\hat{\mathscr{L}}_i(c_j) = 0$ for $j \neq i$ $(j, i = 1, \ldots, \nu)$.

In most cases the first polynomial is chosen, but in those cases where t_m is a jump discontinuity for the solution (see [21]) the second polynomial provides a more accurate uniform approximation.

If $\alpha_{\ell j} \leq t_n$ for all ℓ and j, then all arguments $Z^{\ell j}$ can be explicitly computed by the knowledge of the continuous approximation of the solution in the past, that is $\eta(t)$ for $t \leq t_n$. This situation corresponds to the so-called method of steps (see [8]). In such a case we have to deal locally with a system of ODEs. Nevertheless, if $\alpha_{\ell j} \in (t_n, t_{n+1}]$ for some pair (ℓ, j), it means that there are delays which are smaller than the used stepsize; hence $\eta\left(\alpha_{\ell j}\right)$ is not known explicitly. In fact for $m = n$, u_m (or v_m) identifies the continuous output to be computed in the current interval $[t_n, t_{n+1}]$ (which thus depends on the unknown current stage values). As a consequence the structure of the nonlinear equations (1.12) would be quite different with respect to the previous case.

For this reason, in order to solve the Runge–Kutta equations, we are driven to a more sophisticated scheme with respect to the one used for ODEs.

1.3.3 Tracking the breaking points

As we have seen in Section 1.1, a serious difficulty is the possible loss of regularity of the solution, due to breaking points, even in the presence of smooth functions $f(t,y,z)$, $g(t)$, and $\alpha_i(t,y)$ $(i = 1, \ldots, p)$ in the problem (1.11).

In most cases, only a few breaking points are significant for a numerical integration, because discontinuities in a sufficiently high derivative of the solution are not recognized by the numerical method.

In the case where α_i does not depend on $y(t)$ for all i, the breaking points can be computed in advance by solving first the scalar equations $\alpha_i(\zeta) = t_0$ for ζ, then for every solution ζ_k the scalar equation $\alpha_i(\xi) = \zeta_k$ and so on. For an efficient integration, computed breaking points can be inserted in advance into the mesh. But in the general (so-called state-dependent) case, where - for some i - α_i depends on y, such a computation is not possible a priori.

If the breaking points are not included in the mesh and a variable stepsize integration is used, the stepsizes may be severely restricted near the low order jump discontinuities. Thus it is important to design an algorithm that allows a code to compute automatically the disturbing breaking points and to include them in the mesh of

integration (for an extensive discussion we address the reader to [22]). In this way, not only step rejections will be avoided, but also the accuracy of the approximation will be significantly improved.

Detection of breaking points

To compute the set B of breaking points, at the beginning we set $B = \{t_0\}$ (and possibly include irregular points of the initial function $g(t)$ of problem (1.11)). The problem is to find the zeros of the functions

$$d_\ell(t;\zeta) = \alpha_\ell\big(t,\eta(t)\big) - \zeta, \qquad \ell = 1,\dots,p, \tag{1.16}$$

where $\zeta \in B$ is a previous breaking point and $\eta(t)$ is a suitable approximation to the solution.

A very simple approach would be to test, in every accepted step, whether at least one of the functions $d_\ell(t;\zeta)$ (see (1.16)) changes sign. The breaking point can then be localized, computed and added to the set B. Such a strategy is often expensive because of the many step rejections that usually occur when approaching a breaking point. We consider instead the following strategy (see [22]). Suppose that the problem is integrated successfully until t_n and a stepsize h_{n+1} is proposed for the next step. We expect a breaking point in $[t_n, t_n + h_{n+1}]$ if the following two conditions occur:

(a) the step is rejected, i.e., the iterative solver for the nonlinear system (1.12) fails to converge, or the local error estimate is not small enough,
(b) there exists a previous breaking point ζ such that $d_\ell(t;\zeta) = \alpha_\ell(t,u_{n-1}(t)) - \zeta$ changes sign on $[t_n, t_n + h_{n+1}]$, where $u_{n-1}(t)$ is the continuous output polynomial of the preceding step.

The extrapolated use of u_{n-1} in order to approximate the solution y in the interval $[t_n, t_n + h_{n+1}]$ is safe because we assume that the solution is regular in the previous accepted step. On the other hand the use of extrapolation may lead to an approximation of the breaking point which is not accurate. For this reason we do not use the polynomial u_{n-1} for its computation.

The search hence activates only in case of a stepsize rejection and proceeds through the following phases (we focus attention on the n-th time step, where we assume a stepsize rejection).

Algorithm 1.3 *Assume that the step $[t_n, t_n + h_{n+1}]$ is rejected;*

1. *Look for zeros of the functions*

$$d_\ell(t;\zeta) = \alpha_\ell\Big(t,u_{n-1}(t)\Big) - \zeta, \qquad \zeta \in B, \quad \ell \in \{1,\dots,p\},$$

   for $t \in [t_n, t_n + h_{n+1}]$;
2. **If** *$\hat\ell \in \{1,\dots,p\}$ and $\hat\zeta \in B$ are determined such that $d_{\hat\ell}(t_n;\hat\zeta)\cdot d_{\hat\ell}(t_n + h_{n+1};\hat\zeta) < 0$ pass to Algorithm 1.4;*

3. **Otherwise** *reduce the stepsize according to classical criteria.*

If Algorithm 1.3 actually detects a breaking point, the exact breaking point will be close to the zero of $d_{\hat{\ell}}(t;\hat{\zeta})$. We indicate it by $\hat{\xi}$, that is

$$\alpha_{\hat{\ell}}\Big(\hat{\xi}, u_{n-1}(\hat{\xi})\Big) - \hat{\zeta} = 0. \tag{1.17}$$

Computation of breaking points

Once a breaking point is detected the second phase of the procedure activates with the goal to compute it to the desired accuracy. Let us denote by $\mathbf{Y} = \left(Y^1, \cdots, Y^\nu\right)^{\mathrm{T}}$ the vectors of unknown stage values.

Algorithm 1.4 *Suppose that a breaking point has been detected by Algorithm 1.3.*

1. *We iteratively solve the augmented system*

$$Y^i = y_n + h\sum_{j=1}^{s} a_{ij} f\Big(t_n + c_j h, Y^j, Z^{1j}, \ldots, Z^{pj}\Big), \qquad i = 1, \ldots, \nu \tag{1.18}$$

$$\alpha_{\hat{\ell}}\Big(t_n + h, u_n(t_n + h)\Big) = \hat{\zeta} \tag{1.19}$$

   with respect to the unknowns $\mathbf{Y}$ *and* h.
2. **If** *the iterative process converges* **then** *set* $h_{n+1} = h$ *and* **go to** *4.*
3. **Otherwise** *reduce the stepsize according to classical criteria and* **exit***;*
4. **If** *the step is accepted (that is that the estimated local error is below the required error tolerance)* **then** *the new point* $\xi^* = t_n + h_{n+1}$ *is inserted into the set of computed breaking points;*
5. **Otherwise** *reduce the stepsize according to classical criteria and* **exit**.

Since we are interested in stiff problems we solve the Runge–Kutta equations by means of a suitable Newton process. In order to preserve the tensor structure of the Jacobian in the Newton process for solving (1.12)-(1.13) (see [21] and the next subsection), we alternatively solve (1.18) and (1.19) until convergence (for a convergence analysis see [22]). Experimental tests have shown that this strategy turns out to be very effective.

Remark 1.5. The need of an accurate computation of the breaking points is common to all methods for integration of equations with delays. However, for non-stiff problems, the procedure above turns out to be significantly simplified by using the class of explicit methods described in Section 1.4 because, in that case, the simultaneous solution of the RK equations and the breaking point equation is implicit in the sole unknown h.

Theoretical remarks

Other authors considered alternative techniques for approximating the breaking points (see e.g. [24] and [17]). Contrary to our approach, they do not use the continuous output of the current step, but some approximation whose error is difficult to control.

The main idea presented here is related to the fact that in the algorithm which computes the RK-step, the stepsize is not fixed but is variable; this allows for an accurate computation of the breaking point to the discrete order p of the method (the root $\hat{\xi}$ of (1.17) may instead be a quite inaccurate approximation of it and is related in any case to the uniform order of the method).

For the following discussion we assume that equations (1.18) and (1.19) are solved exactly. Under suitable smoothness and regularity assumptions the following results hold (for the proof see [22]).

Theorem 1.6. *Let $y(t)$ be the solution of (1.11), and let ζ and ξ be exact breaking points of the problem such that $\alpha_i(\xi, y(\xi)) = \zeta$ (for some i). Further, let ζ^* be an approximation of ζ obtained with sufficiently small stepsizes, and let $\xi \in (t_n, t_{n+1})$. If*

$$\frac{d}{dt}\Big(\alpha_i\big(t, y(t)\big)\Big)\Big|_{t=\xi} \neq 0, \tag{1.20}$$

then the breaking point ξ^* computed by the Algorithm 1.4 satisfies

$$|\xi^* - \xi| \leq C\big(\|y_{n+1} - y(t_{n+1})\| + |\zeta^* - \zeta|\big),$$

where C is a suitable constant.

As a consequence of this result we are able to extend Theorem 1.1 (see also Theorem 6.1.2 in [7]).

For problems (1.11) with state dependent delays it may happen that t_n is a numerically computed breaking point, and the corresponding exact breaking point is slightly different. If at this point the solution has a jump discontinuity, the global error cannot be bounded in terms of h. Nevertheless we have the following convergence result. Let H represent the maximal stepsize and $r = \max\{2\nu - 1, \nu + 1\}$, being $2\nu - 1$ is the classical order of the method.

Theorem 1.7. *Consider a smooth delay problem (1.11) on a bounded interval with well separated breaking points satisfying (1.20). If, instead of the exact breaking points, those obtained by Algorithm 1.4 are used, then we have*

$$\|\eta(t) - y(\vartheta)\| = \mathscr{O}(H^r), \tag{1.21}$$

where the function $\vartheta = \vartheta(t)$ satisfies $\vartheta = t + \mathscr{O}(H^r)$.

For the proof see [22].

This is equivalent to the property

$$\|\eta(t) - y(t)\| = \mathscr{O}(H^r) \qquad \text{for all } t \notin J,$$

with

$$J = \bigcup_{i\geq 0} [\xi_i, \xi_i^*],$$

where $\{\xi_i\}_{i\geq 0}$ and $\{\xi_i^*\}_{i\geq 0}$ denote the sets of the *exact* and of the corresponding *numerical* breaking points (respectively)and $[\xi_i, \xi_i^*]$ denotes the interval between them.

Continuous approximation after breaking points

Since the solution is in general not smooth in correspondence of a breaking point, also the continuous approximation may be inadequate. As an example, if the solution has a jump in correspondence of a breaking point, the use of the collocation polynomial should be avoided since it forces a global continuity and hence determines a loss in the uniform approximation accuracy. To obtain a more accurate approximation we prefer to consider in general the polynomial v_m (see (1.15)) of degree $\nu - 1$, which interpolates the values Y^i but not y_m (compare with (1.14)). This choice allows a globally discontinuous approximation to the solution and determines a local uniform order $q = \nu$. This choice might be better with respect to the use of the collocation polynomial also when the solution is theoretically continuous but in practise has a jump, that is it presents a large variation with respect to the stepsize h (see e.g. [21]).

1.3.4 Solving the Runge–Kutta equations

We solve the Runge–Kutta equations by means of a suitable Newton process. Then we make use of the further notation:

- $A := \{a_{ij}\}$ denotes the RK matrix;
- $U_{\ell j} := \begin{cases} u_m(\alpha_{\ell j}) & \text{if } t_m \leq \alpha_{\ell j} \leq t_{m+1} \\ 0 & \text{otherwise.} \end{cases}$

  In order to obtain an accurate computation of the derivatives of the function

$$F_i\left(Y^1, \cdots, Y^\nu, Z^{11}, \cdots, Z^{1\nu}, \ldots, Z^{p1}, \cdots, Z^{p\nu}\right)$$

we consider the following approximation

$$\frac{\partial F_i}{\partial Y^k} \approx \delta_{ik} I_d - h_{n+1} \sum_{\ell=1}^{p} \left(a_{ik} D_{\ell k} + \hat{D}_{\ell k}\right) \tag{1.22}$$

where I_d denotes the $d \times d$-identity matrix, δ_{ik} is the Kronecker delta symbol and

$$D_{\ell k} = \frac{\partial f}{\partial y} + \frac{\partial f}{\partial z_\ell} \eta'(\alpha_{\ell k}) \frac{\partial \alpha_\ell}{\partial y},$$
$$\hat{D}_{\ell k} = \sum_{j=1}^{s} a_{ij} \frac{\partial f}{\partial z_\ell} \frac{\partial U_{\ell j}}{\partial Y^k}.$$

with $\frac{\partial \alpha_\ell}{\partial y} = \frac{\partial \alpha_\ell}{\partial y}(t_n, y_n)$, $\frac{\partial f}{\partial y} = \frac{\partial f}{\partial y}\left(t_n, y_n, \sigma_1, \cdots, \sigma_p\right)$, and $\frac{\partial f}{\partial z_\ell} = \frac{\partial f}{\partial z_\ell}\left(t_n, y_n, \sigma_1, \cdots, \sigma_p\right)$, where $\sigma_\ell = \eta(\alpha_{\ell 0})$ and $\alpha_{\ell 0} = \alpha_\ell(t_n, y_n)$.

We note that the last term $\hat{D}_{\ell k}$ is always zero if the deviating argument falls on the left side of t_n ; more precisely, we get

$$\frac{\partial U_{\ell j}}{\partial Y^k} = \mathscr{U}^{\ell}_{jk} I_d,$$

where

$$\mathscr{U}^{\ell}_{jk} = \begin{cases} \mathscr{L}_k(\psi_{\ell j}) & \text{if} \quad \psi_{\ell j} > 0 \\ 0 & \text{otherwise,} \end{cases} \tag{1.23}$$

with

$$\psi_{\ell j} := \big(\alpha_\ell(t_n + c_j h_{n+1}, Y^j) - t_n\big)/h_{n+1}.$$

The considered approximations make the Newton process inexact and linearly convergent.

Structure of the Jacobian in the general case

In order to solve (1.12) we set $Y = \big((Y^1)^{\mathrm{T}}, (Y^2)^{\mathrm{T}}, \cdots, (Y^\nu)^{\mathrm{T}}\big)^{\mathrm{T}}$, the $\nu \cdot d$-dimensional vector of unknowns, and consider the Newton iteration process. In the general case the Jacobian of (1.12) is given by the following matrix,

$$J = I_\nu \otimes I_d - h_{n+1} A \otimes \left(\frac{\partial f}{\partial y} + \sum_{\ell=1}^{p} \frac{\partial f}{\partial z_\ell} \eta'(\alpha_{\ell 0}) \frac{\partial \alpha_\ell}{\partial y} \right) - h_{n+1} \sum_{\ell=1}^{p} A \cdot \mathscr{U}^{\ell} \otimes \frac{\partial f}{\partial z}, \tag{1.24}$$

where I_ν indicates the $\nu \times \nu$-identity matrix, $\partial f/\partial y$, $\partial f/\partial z$ denote the matrices of the partial derivatives of f w.r.t. the y and z variables respectively, $\partial \alpha_\ell/\partial y$ the row vector of the partial derivatives of α_ℓ w.r.t. y and $\mathscr{U}^{\ell} = \left\{ \mathscr{U}^{\ell}_{jk} \right\}_{j,k=1}^{\nu}$ (see (1.23)).

Although J is actually an approximation of the true Jacobian of (1.12), in order to distinguish it from further simplifications, we shall call the corresponding Newton iteration as *quasi-exact*.

The quasi-exact iteration is always the correct one and in particular it is substantially exact also in the cases when the delay vanishes or the stepsize is larger than the delay. It allows a more efficient solution of the Runge–Kutta equations although being quite expensive; in fact the Jacobian has a full structure (although often *sparse*), so that the cost of the LU factorization of J is $(1/3) \cdot (\nu n)^3$ ($9 \cdot n^3$ if $\nu = 3$). No general reduction to a special structure is possible in this general case. Nevertheless observe that $\mathscr{U}^{\ell}$ is the zero matrix if the corresponding ℓ-th deviating argument does not fall into the current step (see (1.23)). If this situation occurs for all ℓ the problem presents a so-called ODE-like structure.

The ODE-like iteration

Consider the case when $\mathscr{U}^{\ell} = \mathbf{0}$ for all ℓ, that is the case where delays are larger than the stepsize or equivalently that the deviating arguments fall on the left of t_n, i.e. $\alpha_\ell(t_n + c_j h, Y^j) < t_n$ for all $j = 1, \ldots, \nu$ and for all $\ell = 1, \ldots, p$. Then we have that the last term in (1.24) is identically zero; this means that

$$J = I_\nu \otimes I_d - h_{n+1} A \otimes \left(\frac{\partial f}{\partial y} + \sum_{\ell=1}^{p} \frac{\partial f}{\partial z_\ell} \eta'(\alpha_{\ell 0}) \frac{\partial \alpha_\ell}{\partial y} \right) := J_0. \tag{1.25}$$

Then, following the ideas of Bickart and Butcher, the matrix J_0 is pre-multiplied by $(h_{n+1}A)^{-1} \otimes I_d$. Successively, in order to exploit the structure of the system, the idea is to block-diagonalize A^{-1} (this is completely analogous to the ODE-case as shown e.g. in [28]). Indicating by T the transformation matrix, we have $T^{-1}A^{-1}T = D$ (where D is block-diagonal); then, introducing the transformed variables $W := (T^{-1} \otimes I_d)Y$, we obtain an equivalent Newton iteration with Jacobian

$$\hat{J_0} = h_{n+1}^{-1} D \otimes I_d - I_\nu \otimes \left(\frac{\partial f}{\partial y} + \sum_{\ell=1}^{p} \frac{\partial f}{\partial z_\ell} \eta'(\alpha_{\ell 0}) \frac{\partial \alpha_\ell}{\partial y} \right). \tag{1.26}$$

Since the obtained linear system has block-diagonal structure, the linear algebra is certainly more efficient with respect to the original iteration (based on J_0).

But when the stepsize is larger than one or more delays, which means that - for some ℓ and for some $j \in \{1, \ldots, \nu\}$ - $\alpha_\ell(t_n + c_j h_{n+1}, Y^j) > t_n$, the situation is completely different and the previous procedure cannot be fruitfully applied to (1.24). In order to maintain the advantage of the tensor structure given by (1.26) one could proceed by considering an inexact Newton process where the correct Jacobian (that is (1.24)) is only roughly approximated by (1.25). In this way - considering the transformation to (1.26) - the LU-factorization in the Newton process would be cheaper.

The risk lies in the fact that the Newton iteration may become significantly slower or also have problems to converge.

The stopping criteria implemented in the code RADAR5 are similar to those used for ODEs (see [27, 21]). Let us shortly review them. We set $\tilde{Y}^{[k]}$ the k-th iterate generated by the Newton process in order to approximate Y, the exact solution of (1.12), and define $\Delta^{[k]} = \tilde{Y}^{[k]} - \tilde{Y}^{[k-1]}$. If the method is linearly convergent, as we expect, we have $\|\Delta^{[k]}\| \leq \Theta \|\Delta^{[k-1]}\|$ with $|\Theta| < 1$. Then we get the estimate

$$\|\tilde{Y}^{[k]} - Y\| \leq \frac{\Theta}{1-\Theta} \|\Delta^{[k-1]}\|. \tag{1.27}$$

In order to estimate Θ the ratios

$$\Theta_k = \|\Delta^{[k]}\| / \|\Delta^{[k-1]}\|$$

are progressively computed. According to (1.27) and setting $\eta_k = \Theta_k/(1-\Theta_k)$, the iteration is then successfully stopped if

$$\eta_k \|\Delta^{[k]}\| \leq \rho \cdot Tol, \tag{1.28}$$

where Tol is the adopted error tolerance and ρ is a suitable coefficient. We denote by k_{max} the maximum number of allowed iterations. The iteration fails if one of the following situations occur (for some $k \leq k_{max}$):

$$\Theta_k \geq 1, \tag{1.29}$$

$$\frac{\Theta_k^{k_{max}-k}}{1-\Theta_k}\|\Delta^{[k]}\| > \rho \cdot Tol. \tag{1.30}$$

Condition (1.29) indicates that the iteration is diverging while condition (1.30) estimates that (1.28) seems not to be fulfilled within the remaining $k_{max} - k$ iterations.

Preserving the tensor structure of the Jacobian

As we have mentioned, using (1.25) as an inexact approximation of (1.24) may be not safe. An efficient simplification would consist in possibly approximating the matrices $\mathscr{U}^\ell$ in (1.24) as

$$\mathscr{U}^\ell \approx \gamma^\ell I_\nu, \qquad \text{for a suitable } \gamma^\ell \in \mathbb{R}.$$

This would determine for the corresponding Jacobian matrix the same special tensor structure of the Jacobian matrix J_0 (hence allowing a transformation which is analogous to (1.26)).

The ODE-like iteration corresponds to choosing $\gamma^\ell = 0$ for all ℓ. But when $\mathscr{U}^\ell \neq \mathbf{0}$ we have seen (and experimentally verified) that this might be quite critical for the convergence of the Newton process. Thus a suitable choice of the parameter γ^ℓ turns out to be important for the convergence of the Newton iteration.

A structure preserving approximation

A first possibility is that of setting $\mathscr{U}^\ell = I_\nu$ if $\alpha_\ell(t_n + c_j h, Y^j) > t_n$ for some j; this strategy is implemented in the first version of the code RADAR5.

A second possibility, which has been included in the second release of the code RADAR5 consists in finding the optimal coefficient γ^ℓ on the basis of a suitable optimization criterium. We propose the following choice:

$$\gamma^\ell \longrightarrow \min_{\gamma\in\mathbb{R}} \|\mathscr{U}^\ell - \gamma I_\nu\|^2 \tag{1.31}$$

where $\|\cdot\|$ is the Frobenius norm.

This choice has been motivated both by the simple determination of the optimal coefficient, which is be obtained by using such norm, and by the good results obtained in our numerical experience.

Since the argument function in the min in (1.31) is a quadratic function with respect to γ, the global minimizer is computed explicitly (see [23]). We remark that in the special case where $\alpha_\ell(t, y(t)) \equiv t$, we obtain - as we expect - $\gamma^\ell = 1$. This can be reasonably interpreted as the approximate situation where the stepsize is much larger than the corresponding delay.

With the previous procedure we obtain the following approximation of (1.24)

$$J_\gamma = I_\nu \otimes I_d - hA \otimes \left(\frac{\partial f}{\partial y} + \sum_{\ell=1}^{p} \left(\eta'(\alpha_{\ell 0}) \frac{\partial \alpha_\ell}{\partial y} + \gamma^\ell \right) \frac{\partial f}{\partial z_\ell} \right), \tag{1.32}$$

and consequently, by making use of the same transformation used in order to obtain $\hat{J}_0$ (see (1.26)), we get

$$\hat{J}_\gamma = (h_{n+1})^{-1} D \otimes I_d - I_\nu \otimes \left(\frac{\partial f}{\partial y} + \sum_{\ell=1}^{p} \left(\eta'(\alpha_{\ell 0}) \frac{\partial \alpha_\ell}{\partial y} + \gamma^\ell \right) \frac{\partial f}{\partial z_\ell} \right), \quad (1.33)$$

having the same block-diagonal structure of $\hat{J}_0$.

Implemented algorithm in RADAR5

The inexact iterations are computationally convenient. In fact the transformation of the approximated Jacobian to (1.33) is very convenient; for the 3-stage Radau method the cost for the linear systems would be $(5/3) \cdot n^3$ ops.

We allow two possible Newton iterations in the method:

- a cheap *inexact* iteration which consists of setting $\mathscr{U}^\ell = \gamma^\ell I_\nu$ (according to (1.31)) and leads to a block-diagonal structure Jacobian matrix;
- an expensive *quasi-exact* iteration, which consists of taking $\mathscr{U}^\ell \not\equiv \mathbf{0}$ according to (1.23), which leads to a full-structure Jacobian matrix.

The first iteration turns out to be equivalent to the second one if the stepsize is smaller than all delays, that is all deviating arguments fall on the left-hand side of the point t_n; furthermore it is very close to the second also when the delays are much larger than the stepsize, that is when $\alpha_\ell(s, y(s)) \approx s$ (for all ℓ) in the integration interval $[t_n, t_{n+1}]$. The strategy we have chosen is the following. We make use of an iteration indicator Iflag which drives the procedure.

Algorithm 1.8 *At the beginning of the step, if necessary, we compute the ODE-like Jacobian* J_0.

1. **If** *necessary we compute the optimal values* γ^ℓ $(\ell = 1, \ldots, p)$ *according to (1.31) and update the Jacobian* J_γ *according to (1.33). Then we set* Iflag $= 1$ *(inexact iteration).*
2. **Otherwise** *we set* Iflag $= 0$ *(pure ODE-like iteration).*
3. *Apply the inexact iteration.*
4. **If** *the iteration fails (see (1.29) and (1.30)) we stop it and* **go to** *6.*
5. **Otherwise** *we accept the step and* **exit**.
   Possible switch to a exact iteration.
6. **If** Iflag $= 0$ *we reduce the stepsize and* **restart** *a new step:* **go to** *3.*
7. **Otherwise** *we set* Iflag $= 2$ *(quasi-exact iteration).*
8. *Apply the quasi-exact iteration.*
9. **If** *the quasi-exact iteration fails (see (1.29) and (1.30)) we stop it and reduce the stepsize. Then we* **restart** *a new step:* **go to** *1.*
10. **Otherwise** *we accept the step and* **exit**.

1.3.5 Local Error Estimation and Step Size Control

Step size selection strategies for stiff ordinary differential equations are usually based on error estimations at grid points. For delay equations, where the accuracy of the dense output strongly influences the performance, such an approach is not sufficient. We shortly recall the technique used in RADAU5 [27, Sect. IV.8], and we discuss a modification suitable for delay equations. Standard error estimators for ordinary differential equations are based on embedded methods. This leads to

$$\Delta y_n = h_{n+1} f(y_n, z_n) + \sum_{i=1}^{\nu} e_i \big(Y^i - y_n\big), \tag{1.34}$$

where the coefficients e_i are chosen such that $\Delta y_n = \mathscr{O}(h_{n+1}^{\nu+1})$ whenever the problem and the solution are smooth. For very stiff problems, the expression Δy_n largely overestimates the true local error; thus it is pre-multiplied by the projection matrix

$$P = \big(I_d - h_{n+1}\lambda(f_y + \ldots)\big)^{-1}, \tag{1.35}$$

where λ is a real eigenvalue of the Runge–Kutta matrix A. Whenever the tensor product structure in (1.33) is exploited, an LU decomposition of the matrix $I_d - h_{n+1}\lambda(f_y + \ldots)$ is already available from the simplified Newton iterations..

We consider the following norm for an arbitrary (error) vector w_n,

$$\|w_n\|^2 = \frac{1}{d} \sum_{i=1}^{d} \left(\frac{w_{n,i}}{s_i} \right)^2,$$

where $s_i = 1 + \rho\,|y_{n,i}|$ and ρ is the ratio tol_r/tol_a between the relative (tol_r) and absolute (tol_a) input tolerances per step (which are used for the stepsize selection). Then we denote by ω_n the following measure of the error at grid points,

$$\omega_n = \|\Delta y_n\|.$$

We shall call ω_n as the *discrete* component of the local error. In the general case, the local order of the error-estimating method turns out to be $\nu + 1$, that is $\omega_n = \mathscr{O}(h_{n+1}^{\nu+1})$.

Estimation of the error in the dense output

As we have mentioned, for delay equations, where the uniform accuracy of the numerical solution has also influence on the local error, it is necessary to control the error uniformly in time. To do this we may consider in general also the polynomial v_m (see (1.15)) of degree $\nu - 1$, which interpolates the values Y^i but not y_m. It turns out that

$$\eta_n = \max_{\vartheta \in [0,1]} \|u_n(t_n + h_{n+1}\vartheta) - v_n(t_n + h_{n+1}\vartheta)\| = \|u_n(t_n) - v_n(t_n)\| = \mathscr{O}(h_{n+1}^{\nu}).$$

We use this quantity as an indicator for the uniform error and denote it as *continuous* component of the local error.

The estimate used for the stepsize control is finally given by

$$\mathrm{err}_n = \gamma_1 \omega_n + \gamma_2 (\eta_n)^{(\nu+1)/\nu} = \mathscr{O}(h_{n+1}^{\nu+1}), \tag{1.36}$$

with the parameters $\gamma_1, \gamma_2 \geq 0$ possibly tuned by the user. This choice is the fruit of both theoretical and empirical analysis. The order of the estimation is $\nu+1$ (that is 4 if $\nu = 3$) when the solution is smooth, and is obtained quite cheaply. After error estimation stepsize prediction is obtained by classical formulas (see [27]).

1.3.6 Numerical illustration for the Waltman problem

In this final paragraph we illustrate the behavior of the algorithms presented in this section, which have been implemented in the code RADAR5 (version 2). The code applied to problem (1.9) behaved very well.

In particular, let us focus our attention on the breaking point computation technique, on the devised Newton process, and on the error control strategy.

We consider the following choices for the model problem (1.9)–(1.10): $f_1(x,y,w) = xy + w$ and $f_2(y,w) = y + w$ are the functions modelling the accumulation effects in (1.7) and (1.8), $a = 1.8$ and $b = 20$ are the amplification factors, $\gamma = 0.002$ is a catabolic factor, $r_1 = 5 \cdot 10^4, r_2 = 0$ and $s = 10^5$ are combination factors. Finally, in order to simplify the problem, we fix $t_0 = 35$, $t_1 = 197$ as the activation instants.

The initial values and initial functions are given by $y_1(t) = 5 \cdot 10^{-6}$, $y_2(t) = 10^{-15}$, and $y_3(t) = y_4(t) = \alpha_1(t) = \alpha_2(t) = 0$ for $t \leq 0$.

breaking point	ancestor	argument
$\xi_1 = 55.21325176$	t_0	α_1
$\xi_2 = 69.26718167$	ξ_1	α_1
$\xi_3 = 79.63960593$	ξ_2	α_1
$\xi_4 = 197.0000071$	t_0	α_2
$\xi_5 = 197.0000115$	ξ_1	α_2
$\xi_6 = 197.0000125$	ξ_5	α_1
$\xi_7 = 197.0000147$	ξ_2	α_2
$\xi_8 = 197.0000173$	ξ_3	α_2

Table 1.1. Some of the computed breaking points.

The right-hand side of the differential equation has jump discontinuities at $t_0 = 35$ and $t_1 = 197$, but the solution is continuous and has jumps only in its derivatives. There are two state dependent delays $\alpha_1(t,y(t)) = y_5(t)$ and $\alpha_2(t,y(t)) = y_6(t)$. After

activation, the first delay monotonically approaches a constant value (nearly vanishing delay); the second has an extremely steep slope and rapidly approaches 0 (see Fig. 1.2).

The code successfully solves this problem on the interval $[0,300]$ for all tolerances. Breaking points for the solution are obviously t_0 and t_1 (included in the mesh). The code further computes several breaking points, some of which are indicated in Table 1.1. Due to the nearly vanishing delays in the problem, there is a huge amount of breaking points beyond $t = 197$, and our computation shows that only a few of them are important and have to be included in the mesh. Any code that tries to compute all breaking points will be inefficient for this problem, because it has to take too small steps. With relative and absolute tolerances $tol_r = 10^{-6}$, $tol_a = 10^{-6} tol_r$ at the endpoint $t = 300$ the numerically computed values of the deviating arguments are

$$\alpha_1(300) = 299.9999, \qquad \alpha_2(300) = 299.6649,$$

while the stepsize $h = 16.5045$, which means that the stepsize is much larger than the delays.

	version 2		version 1	
e	feval	error	feval	error
3	2227	0.218	2800	0.778
6	3409	6.85e-4	4244	1.05e-2
9	7939	3.32e-6	8537	2.48e-4
12	22694	3.66e-8	—-	——

Table 1.2. Error behavior: comparison between versions 1 and 2 of RADAR5.

Table 1.2 illustrates the behavior of the code for various relative error tolerances (per step), which we denote by tol_r. We indicate by $e = -\log(tol_r)$. We denote by feval the number of function evaluations and by error the computed relative error on the solution. For $tol_r = 10^{-12}$ version 1 stops soon after t_1.

Finally look at the effects of the approximation of the Jacobian described at the end of Section 1.3.4, and implemented in version 2, of the code RADAR5 (nstep is the total number of steps).

	version 2		version 1	
e	nstep	feval	nstep	feval
4	210	1575	393	2798
6	316	2307	443	3091
8	631	4554	798	5621
10	1183	8544	1447	10340
12	2311	16644	2799	20021

Table 1.3. Behavior of Newton iteration: comparison between versions 1 and 2 of RADAR5.

Table 1.3 shows that the new version of the code is more efficient than the previous one. This confirms that a better approximation of the Jacobian achieved.

Observe that for smaller tolerances, the advantage of the novel approach is less evident since the average stepsize decreases and overlapping occurs less frequently.

1.3.7 Software

Release 2.1 of the code RADAR5 is presently being distributed at the web-sites
http://univaq.it/∼guglielm
http://www.unige.ch/∼hairer/software.html
with several examples, including the considered Waltman model.

1.4 The Functional Continuous Runge-Kutta Method

In this section we construct the Functional Continuous Runge-Kutta Method introduced in Section 1.2 for the general Retarded Functional Differential Equation in the form

$$\begin{cases} y'(t) = f(t, y_t),\ t \geq t_0 \\ y(t) = \phi(t),\ t \leq t_0. \end{cases} \tag{1.37}$$

According to (1.5) with the option (1.6) for the stages-functions $Y^{i}_{t^i_{n+1}}$, the FCRK methods takes the form

$$\eta(t_n + \theta h_{n+1}) = \eta(t_n) + h_{n+1} \sum_{i=1}^{\nu} b_i(\theta) K^i,\ \theta \in [0,1] \tag{1.38}$$

where the derivatives K^i are given by

$$K^i = f\left(t^i_{n+1}, Y^i_{t^i_{n+1}}\right) \tag{1.39}$$

and $Y^i_{t^i_{n+1}}$ is a stage-function given by

$$\begin{aligned} &Y^i(t_n + \theta h_{n+1}) = \eta(t_n) + h_{n+1} \sum_{j=1}^{\nu} a_{ij}(\theta) K^j,\ \theta \in [0, c_i] \\ &Y(t) = \eta(t),\ t \leq t_n. \end{aligned} \tag{1.40}$$

Note that the coefficients a_{ij}, $i, j = 1, ..., \nu$, are polynomial functions of the parameter $\theta \in [0,1]$ and this feature renders these schemes different from Continuous Runge-Kutta (CRK) methods where only the weights b_i, $i = 1, ..., \nu$, are polynomial functions of θ.

The method is called explicit if $a_{ij}(\cdot)$ is the zero function for $j \geq i$. In case of explicit methods the derivatives K^i, $i = 1, ..., \nu$, can be successively computed as:

- $K^1 = f\left(t^1_{n+1}, Y^1_{t^1_{n+1}}\right)$ where

$$\begin{aligned} &Y^1(t_n + \theta h_{n+1}) = \eta(t_n),\ \theta \in [0, c_1] \\ &Y^1(t) = \eta(t),\ t \le t_n \end{aligned}$$

(note that $K^1 = f(t_n, \eta_{t_n})$ when $c_1 = 0$) and

- for $i = 2, ..., \nu$, $K^i = f\left(t^i_{n+1}, Y^i_{t^i_{n+1}}\right)$ where

$$\begin{aligned} &Y^i(t_n + \theta h) = \eta(t_n) + h_{n+1} \sum_{j=1}^{i-1} a_{ij}(\theta) K^j,\ \theta \in [0, c_i] \\ &Y^i(t) = \eta(t),\ t \le t_n. \end{aligned}$$

On the contrary in the general implicit case described by (1.39) and (1.40) the derivatives vector $K = \left(K^1, ..., K^\nu\right) \in \mathbb{R}^{\nu d}$ is the solution of an $\nu d-$dimensional algebraic system.

Remark 1.9. Let us note that if $a_{ij}(\cdot) = b_j(\cdot)$, then $Y^i = \eta$ for $i = 1, ..., \nu$ and the FCRK method coincides with the standard approach. On the contrary a CRK method $(A, b(\cdot), c)$ coincides with a FCRK method when it is natural, i.e. $a_{ij} = b_j(c_i)$, $i, j = 1, ..., \nu$, and the coefficients are given by $a_{ij}(\cdot) = b_j(\cdot)$.

When the FCRK method is applied to ODEs, only the value $\eta(t_n + h_{n+1})$ is needed and (1.38), (1.39) and (1.40) become

$$\eta(t_n + h_{n+1}) = \eta(t_n) + h_{n+1} \sum_{i=1}^{\nu} b_i(1) K^i,$$

$$K^i = f\left(t^i_{n+1}, Y^i\left(t^i_{n+1}\right)\right)$$

and

$$Y^i\left(t^i_{n+1}\right) = \eta(t_n) + h_{n+1} \sum_{j=1}^{\nu} a_{ij}(c_i) K^j \tag{1.41}$$

respectively. So it is the same as the $\nu-$stage RK method for ODEs (A, b, c) where A is $\nu \times \nu-$matrix of elements

$$a_{ij} = a_{ij}(c_i),\ i, j = 1, ..\nu \tag{1.42}$$

and b is the $\nu-$vector of components

$$b_i = b_i(1),\ i = 1, ..., \nu.$$

We denote the FCRK method by the usual Butcher tableau

$$\begin{array}{c|ccc} c_1 & a_{11}(\theta) & \dots & a_{1\nu}(\theta) \\ \vdots & \vdots & & \vdots \\ c_\nu & a_{\nu 1}(\theta) & \dots & a_{\nu\nu}(\theta) \\ \hline & b_1(\theta) & \dots & b_\nu(\theta) \end{array}$$

Here are some elementary examples of FCRK methods.

- One-stage FCRK methods:

$$\begin{array}{c|c} c & \theta \\ \hline & \theta \end{array} \tag{1.43}$$

  i.e.

$$\eta(t_n+\theta h_{n+1}) = \eta(t_n) + h_{n+1}\theta K^1, \quad \theta \in [0,1]$$

  where $K^1 = f\left(t_n + ch_{n+1}, Y^1_{t_n+ch_{n+1}}\right)$ and

$$\begin{array}{l} Y^1(t_n+\theta h_{n+1}) = \eta(t_n) + h_{n+1}\theta K^1, \quad \theta \in [0,c] \\ Y^1(t) = \eta(t),\ t \le t_n. \end{array}$$

  In particular, for $c = 0, 1/2, 1$ we get the explicit Euler, Midpoint and implicit Euler FCRK methods respectively.
- Trapezoidal FCRK method:

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & \frac{1}{2}\theta & \frac{1}{2}\theta \\ \hline & \frac{1}{2}\theta & \frac{1}{2}\theta \end{array} \tag{1.44}$$

  i.e.

$$\eta(t_n+\theta h_{n+1}) = \eta(t_n) + h_{n+1}\frac{\theta}{2}\left(K^1+K^2\right),\ \theta \in [0,1].$$

  where $K^1 = f(t_n, \eta_{t_n})$, $K^2 = f\left(t_{n+1}, Y^2_{t_{n+1}}\right)$ and

$$\begin{array}{l} Y^2(t_n+\theta h_{n+1}) = \eta(t_n) + h_{n+1}\frac{\theta}{2}\left(K^1+K^2\right),\ \theta \in [0,1]. \\ Y^2(t) = \eta(t),\ t \le t_n. \end{array}$$

According to the Remark 1.9, the foregoing examples turn out to be standard approach. Next two are not.

- Another version of the trapeziodal FCRK method:

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & \frac{1}{2}\theta & \frac{1}{2}\theta \\ \hline & \theta - \frac{1}{2}\theta^2 & \frac{1}{2}\theta^2 \end{array} \tag{1.45}$$

i.e.

$$\eta(t_n+\theta h_{n+1}) = \eta(t_n) + h_{n+1}\left[\left(\theta - \frac{\theta^2}{2}\right)K^1 + \frac{\theta^2}{2}K^2\right],\ \theta\in[0,1]$$

where $K^1 = f(t_n,\eta_{t_n})$, $K^2 = f\left(t_{n+1}, Y^2_{t_{n+1}}\right)$ and

$$\begin{array}{l} Y^2(t_n+\theta h_{n+1}) = \eta(t_n) + h_{n+1}\frac{\theta}{2}\left(K^1+K^2\right),\ \theta\in[0,1] \\ Y^2(t) = \eta(t),\ t\le t_n. \end{array}$$

This version differs from the previous one in what it has uniform order two instead of one.

- Heun method:

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & \theta & 0 \\ \hline & \theta - \frac{1}{2}\theta^2 & \frac{1}{2}\theta^2 \end{array} \tag{1.46}$$

i.e.

$$\eta(t_n+\theta h_{n+1}) = \eta(t_n) + h_{n+1}\left[\left(\theta - \frac{\theta^2}{2}\right)K^1 + \frac{\theta^2}{2}K^2\right],\ \theta\in[0,1]$$

where $K^1 = f(t_n,\eta_{t_n})$, $K^2 = f\left(t_{n+1}, Y_{t_{n+1}}\right)$ and

$$\begin{array}{l} Y(t_n+\theta h_{n+1}) = \eta(t_n) + h_{n+1}\theta K^2,\ \theta\in[0,1] \\ Y(t) = \eta(t),\ t\le t_n. \end{array}$$

The explicit Euler and Heun methods for the general RFDE (1.37) were first presented by Cryer and Tavernini in [13]. In the subsequent paper [35], Tavernini considered particular implicit FCRK methods derived from collocation and particular explicit FCRK methods derived from predictor-corrector versions of the formers. In particular, he obtained the four-stage explicit method

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ 1 & \theta & 0 & 0 & 0 \\ \frac{1}{2} & \theta - \frac{\theta^2}{2} & \frac{\theta^2}{2} & 0 & 0 \\ 1 & \theta - \frac{\theta^2}{2} & \frac{\theta^2}{2} & 0 & 0 \\ \hline & \theta - \frac{3\theta^2}{2} + \frac{2\theta^3}{3} & 0 & 2\theta^2 - \frac{4\theta^3}{3} & -\frac{\theta^2}{2} + \frac{2\theta^3}{3} \end{array}$$

and the seven-stage explicit method

$$\begin{array}{c|ccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & \theta & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \theta - \frac{\theta^2}{2} & \frac{\theta^2}{2} & 0 & 0 & 0 & 0 & 0 \\ 1 & \theta - \frac{\theta^2}{2} & \frac{\theta^2}{2} & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & \theta - \frac{3\theta^2}{2} + \frac{2\theta^3}{3} & 0 & 2\theta^2 - \frac{4\theta^3}{3} & -\frac{\theta^2}{2} + \frac{2\theta^3}{3} & 0 & 0 & 0 \\ \frac{2}{3} & \theta - \frac{3\theta^2}{2} + \frac{2\theta^3}{3} & 0 & 2\theta^2 - \frac{4\theta^3}{3} & -\frac{\theta^2}{2} + \frac{2\theta^3}{3} & 0 & 0 & 0 \\ 1 & \theta - \frac{3\theta^2}{2} + \frac{2\theta^3}{3} & 0 & 2\theta^2 - \frac{4\theta^3}{3} & -\frac{\theta^2}{2} + \frac{2\theta^3}{3} & 0 & 0 & 0 \\ \hline & b_1(\theta) & 0 & 0 & 0 & b_5(\theta) & b_6(\theta) & b_7(\theta) \end{array} \tag{1.47}$$

where

$$
\begin{aligned}
b_1(\theta) &= \theta - \tfrac{11\theta^2}{4} + 3\theta^3 - \tfrac{9\theta^4}{8} \\
b_5(\theta) &= \tfrac{9\theta^2}{2} - \tfrac{15\theta^3}{2} + \tfrac{27\theta^4}{8} \\
b_6(\theta) &= -\tfrac{9\theta^2}{4} + 6\theta^3 - \tfrac{27\theta^4}{8} \\
b_7(\theta) &= \tfrac{\theta^2}{2} - \tfrac{3\theta^3}{2} + \tfrac{9\theta^4}{8}.
\end{aligned}
$$

More recently Maset, Torelli and Vermiglio in [32] provided, for the FCRK method, uniform and discrete order conditions up to the order four and found the minimum number of stages necessary in the explicit case.

In the remainder of this section we provide the order conditions for FCRK methods and construct explicit methods of uniform global order two, three and four. Finally, we analyze the effect of perturbations due to approximations in the evaluation of the right-hand side function f in (1.37).

1.4.1 Order conditions

Henceforward we assume the following simplifying conditions for the $\nu-$stage FCRK method $(A(\cdot), b(\cdot), c)$

$$
\begin{aligned}
&\sum_{i=1}^{\nu} b_i(\theta) = \theta,\ \theta \in [0,1] \\
&\sum_{j=1}^{\nu} a_{ij}(\theta) = \theta,\ \theta \in [0,c_i],\ i = 1,...,\nu.
\end{aligned}
\tag{1.48}
$$

which guarantee the uniform order one. Moreover, we set $b_i = b_i(1),\ \ i = 1,...,\nu$, and denote by $c_1^*,...,c_{\nu^*}^*$ the distinct c_i's.

The (necessary and sufficient) condition for the uniform order two is

$$
\sum_{i=1}^{\nu} b_i(\theta)\, c_i = \frac{\theta^2}{2},\ \theta \in [0,1] \tag{1.49}
$$

whereas the condition for the discrete order two is the previous one with $\theta = 1$.

Therefore the one-stage methods (1.43), which have uniform order one, have discrete order two iff $c = \frac{1}{2}$. On the basis of Theorem 1.1, the method (1.43) with $c = \frac{1}{2}$ has global order two.

Moreover the two versions (1.44) and (1.45) of the trapezoidal rule have discrete order two and uniform order one and two, respectively. Thus, both of them have the global order two.

In Table 1.4 we show the (necessary and sufficient) additional conditions for the uniform orders three and four.
The conditions for the discrete orders three and four are obtained by setting $\theta = 1$.

Note that the conditions are quite different from the order conditions for the RK methods. The most striking difference are the sums

Order	Conditions
3	$\sum_{i=1}^{v} b_i(\theta) c_i^2 = \frac{\theta^3}{3},\ \theta \in [0,1]$
	for $m = 1,...,v^*$: $\sum_{\substack{i=1 \\ c_i = c_m^*}}^{v} b_i(\theta) \left(\sum_{j=1}^{v} a_{ij}(\beta) c_j - \frac{\beta^2}{2} \right) = 0,\ \theta \in [0,1],\ \beta \in [0, c_m^*]$
4	$\sum_{i=1}^{v} b_i(\theta) c_i^3 = \frac{\theta^4}{4},\ \theta \in [0,1]$
	for $m = 1,...,v^*$: $\sum_{\substack{i=1 \\ c_i = c_m^*}}^{v} b_i(\theta) \left(\sum_{j=1}^{v} a_{ij}(\beta) c_j^2 - \frac{\beta^3}{3} \right) = 0,\ \theta \in [0,1],\ \beta \in [0, c_m^*]$
	for $l,m = 1,...,v^*$: $\sum_{\substack{i=1 \\ c_i = c_l^*}}^{v} \sum_{\substack{j=1 \\ c_j = c_m^*}}^{v} b_i(\theta) a_{ij}(\beta) \left(\sum_{k=1}^{v} a_{jk}(\gamma) c_k - \frac{\gamma^2}{2} \right) = 0,\ \theta \in [0,1],\ \beta \in [0, c_l^*],\ \gamma \in [0, c_m^*]$

Table 1.4. Uniform order conditions for FCRK methods.

$$\sum_{\substack{i=1 \\ c_i = c_m^*}}^{v}$$

where we do not sum over all nodes but only over those that are equal to c_m^*.

1.4.2 Explicit methods

In this section we specialize the previous order conditions to explicit methods by setting $a_{ij}(\cdot)$ equal to the zero function for $j \geq i$ and then construct methods up to global order four. Explicit methods satisfying (1.48) must have $c_1 = 0$. Moreover we assume, without loss of generality, $c_i \neq 0$ for $i = 2,...,v$ and set $c_1^* = 0$.

Two-stage explicit methods satisfying (1.48) take the form

$$\begin{array}{c|cc} 0 & 0 & 0 \\ c_2 & \theta & 0 \\ \hline & \theta - b_2(\theta) & b_2(\theta) \end{array} \tag{1.50}$$

with $c_2 \neq 0$. By (1.49), we see that the methods of uniform order two are given by

$$b_2(\theta) = \frac{\theta^2}{2c_2},\ \theta \in [0,1].$$

For example, for $c_2 = 1$ we obtain the Heun method (1.46) and for $c_2 = \frac{1}{2}$ we obtain the method

$$\begin{array}{c|cc} 0 & 0 & 0 \\ \frac{1}{2} & \theta & 0 \\ \hline & \theta - \theta^2 & \theta^2 \end{array}$$

that we could call Runge method since it reduces, in the ODE case, to the classical Runge method

$$\begin{array}{c|cc} 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \hline & 0 & 1 \end{array}.$$

Since the methods (1.50) have uniform order one, the discrete order two suffices for the global order two and, by (1.49) with $\theta = 1$, this is obtained when

$$b_2 = \frac{1}{2c_2}.$$

Order three and four

Consider three-stage explicit methods satisfying the simplifying assumptions (1.48)

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ c_2 & \theta & 0 & 0 \\ c_3 & \theta - a_{32}(\theta) & a_{32}(\theta) & 0 \\ \hline & \theta - b_2(\theta) - b_3(\theta) & b_2(\theta) & b_3(\theta) \end{array} \tag{1.51}$$

where $c_2, c_3 \neq 0$. By (1.49) the condition for uniform order two is

$$b_2(\theta)\, c_2 + b_3(\theta)\, c_3 = \frac{\theta^2}{2},\ \theta \in [0,1]. \tag{1.52}$$

The first condition in Table 1.4 for uniform order three is

$$b_2(\theta)\, c_2^2 + b_3(\theta)\, c_3^2 = \frac{\theta^3}{3},\ \theta \in [0,1]. \tag{1.53}$$

Thus there exist polynomials b_2, b_3 satisfying (1.52) and (1.53) only if $c_2 \neq c_3$. In this case one of the two remaining conditions for the uniform order three reads

$$b_2(\theta)\left(-\frac{\beta^2}{2}\right) = 0,\ \theta \in [0,1],\ \beta \in [0, c_2].$$

Since $b_2(\cdot) \neq 0$ (this follows by (1.52) and (1.53)) explicit FCRK methods (1.51) of uniform order three do not exist. Note that the same order barrier holds for explicit CRK methods where four stages are necessary (and sufficient) for uniform order three (see [7]).

Now we look for three-stage explicit methods (1.51) of uniform order two and discrete order three, and then of global order three. First, let us consider the case $c_2 \neq c_3$. By (1.49) and Table 1.4 with $\theta = 1$, the necessary and sufficient conditions for uniform order two and discrete order three are

$$\begin{cases} b_2(\theta)\, c_2 + b_3(\theta)\, c_3 = \frac{\theta^2}{2},\ \theta \in [0,1] \\ b_2 c_2^2 + b_3 c_3^2 = \frac{1}{3} \\ b_2\left(-\frac{\beta^2}{2}\right) = 0,\ \beta \in [0, c_2] \\ b_3\left(a_{32}(\beta)\, c_2 - \frac{\beta^2}{2}\right) = 0,\ \beta \in [0, c_3]. \end{cases}$$

The third condition yields $b_2 = 0$ and then the first two conditions are satisfied when $c_3 = \frac{2}{3}$ and $b_3 = \frac{3}{4}$. Thus a method (1.51) is of uniform order two and discrete order three if and only if

$$\begin{aligned}
&c_3 = \tfrac{2}{3}\\
&b_2 = 0\\
&b_3(\theta) = \tfrac{3\theta^2}{4} - \tfrac{3}{2}b_2(\theta)c_2,\ \theta \in [0,1]\\
&a_{32}(\theta) = \tfrac{\theta^2}{2c_2},\ \theta \in [0,c_3].
\end{aligned}$$

An example of such a method (obtained with $b_2(\cdot) = 0$ and $c_3 = \frac{1}{3}$) is

$$\begin{array}{c|ccc}
0 & 0 & 0 & 0\\
\frac{1}{3} & \theta & 0 & 0\\
\frac{2}{3} & \theta - \frac{3}{2}\theta^2 & \frac{3}{2}\theta^2 & 0\\
\hline
 & \theta - \frac{3}{4}\theta^2 & 0 & \frac{3}{4}\theta^2
\end{array}$$

that reduces to the three-stage Heun method

$$\begin{array}{c|ccc}
0 & 0 & 0 & 0\\
\frac{1}{3} & \frac{1}{3} & 0 & 0\\
\frac{2}{3} & 0 & \frac{2}{3} & 0\\
\hline
 & \frac{1}{4} & 0 & \frac{3}{4}
\end{array}$$

in the ODEs case.

Other methods (1.51) of uniform order two and discrete order three can be obtained when $c_2 = c_3$. In this case (1.51) is of uniform order two and discrete order three if and only if

$$\begin{aligned}
&c_2 = c_3 = \tfrac{2}{3}\\
&b_3 \neq 0\\
&b_2(\theta) = \tfrac{3\theta^2}{4} - b_3(\theta),\ \theta \in [0,1]\\
&a_{32}(\theta) = \tfrac{9\theta^2}{16b_3},\ \theta \in [0,c_3].
\end{aligned}$$

An example (obtained with $b_3(\theta) = \frac{1}{2}\theta$) is given by

$$\begin{array}{c|ccc}
0 & 0 & 0 & 0\\
\frac{2}{3} & \theta & 0 & 0\\
\frac{2}{3} & \theta - \frac{9}{8}\theta^2 & \frac{9}{8}\theta^2 & 0\\
\hline
 & -\frac{3}{4}\theta^2 + \theta & \frac{3}{4}\theta^2 - \frac{1}{2}\theta & \frac{1}{2}\theta
\end{array}$$

that reduces to the

$$\begin{array}{c|ccc}
0 & 0 & 0 & 0\\
\frac{2}{3} & \frac{2}{3} & 0 & 0\\
\frac{2}{3} & \frac{1}{6} & \frac{1}{2} & 0\\
\hline
 & \frac{1}{4} & \frac{1}{4} & \frac{1}{2}
\end{array}$$

in the ODEs case.

We can conclude that three-stage FCRK methods of global order three actually exist as in the ordinary case.

When we pass to consider FCRK methods of global order four, i.e. uniform order three and discrete order four, six stages turns out to be necessary and sufficient. Note that for explicit CRK methods, the uniform order three and discrete order four is achieved with four stages only (see [7]).

Consider then a six-stage explicit method satisfying (1.48)

$$\begin{array}{c|cccccc}
0 & & & & & & \\
c_2 & \theta & & & & & \\
c_3 & \theta - a_{32}(\theta) & a_{32}(\theta) & & & & \\
c_4 & \theta - \sum_{j=2}^{3} a_{4j}(\theta) & a_{42}(\theta) & a_{43}(\theta) & & & \\
c_5 & \theta - \sum_{j=2}^{4} a_{5j}(\theta) & a_{52}(\theta) & a_{53}(\theta) & a_{54}(\theta) & & \\
c_6 & \theta - \sum_{j=2}^{5} a_{6j}(\theta) & a_{62}(\theta) & a_{63}(\theta) & a_{64}(\theta) & a_{65}(\theta) & \\
\hline
 & \theta - \sum_{i=2}^{6} b_i(\theta) & b_2(\theta) & b_3(\theta) & b_4(\theta) & b_5(\theta) & b_6(\theta)
\end{array} \tag{1.54}$$

where $c_2, c_3, c_4, c_5, c_6 \neq 0$.

By using the conditions for uniform order three and the conditions for discrete order four ($\theta = 1$) in Table 1.4, we can show that a six-stage explicit FCRK method (1.54) is of uniform order three and discrete order four if

$$\begin{array}{l}
c_3 \neq c_4 \\
\frac{c_5+c_6}{3} - \frac{c_5 c_6}{2} = \frac{1}{4} \\
b_2(\cdot) = 0 \\
b_3, b_4 = 0 \\
b_3(\theta)c_3 + b_4(\theta)c_4 + b_5(\theta)c_5 + b_6(\theta)c_6 = \frac{\theta^2}{2},\ \theta \in [0,1] \\
b_3(\theta)c_3^2 + b_4(\theta)c_4^2 + b_5(\theta)c_5^2 + b_6(\theta)c_6^2 = \frac{\theta^3}{3},\ \theta \in [0,1] \\
a_{32}(\theta) = \frac{\theta^2}{2c_2},\ \theta \in [0, c_2] \\
a_{42}(\theta)c_2 + a_{43}(\theta)c_3 = \frac{\theta^2}{2},\ \theta \in [0, c_4] \\
a_{52}(\cdot) = 0 \\
a_{53}(\theta) = \frac{\theta^2 c_4}{2c_3(c_4-c_3)} - \frac{\theta^3}{3c_3(c_4-c_3)},\ \theta \in [0, c_5] \\
a_{54}(\theta) = -\frac{\theta^2 c_3}{2c_4(c_4-c_3)} + \frac{\theta^3}{3c_4(c_4-c_3)},\ \theta \in [0, c_5] \\
a_{62}(\cdot) = 0 \\
a_{63}(\theta)c_3 + a_{64}(\theta)c_4 + a_{65}(\theta)c_5 = \frac{\theta^2}{2},\ \theta \in [0, c_6] \\
a_{63}(\theta)c_3^2 + a_{64}(\theta)c_4^2 + a_{65}(\theta)c_5^2 = \frac{\theta^3}{3},\ \theta \in [0, c_6].
\end{array} \tag{1.55}$$

So, by taking the abscissae c_3, c_4, c_5, c_6 such that $c_3 \neq c_4,\ c_4 \neq c_5$ and

$$\frac{c_5 + c_6}{3} - \frac{c_5 c_6}{2} = \frac{1}{4}, \tag{1.56}$$

and weights and coefficients such that

$$\begin{aligned} &b_2(\cdot) = b_3(\cdot) = b_4(\cdot) = 0 \\ &a_{42}(\cdot) = 0 \\ &a_{52}(\cdot) = 0 \\ &a_{62}(\cdot) = a_{63}(\cdot) = 0 \end{aligned}$$

we obtain the following tableau

$$\begin{array}{c|cccccc} 0 & & & & & & \\ c_2 & \theta & & & & & \\ c_3 & \theta - \frac{\theta^2}{2c_2} & \frac{\theta^2}{2c_2} & & & & \\ c_4 & \theta - \frac{\theta^2}{2c_3} & 0 & \frac{\theta^2}{2c_3} & & & \\ c_5 & \theta - a_{53}(\theta) - a_{54}(\theta) & 0 & a_{53}(\theta) & a_{54}(\theta) & & \\ c_6 & \theta - a_{64}(\theta) - a_{65}(\theta) & 0 & 0 & a_{64}(\theta) & a_{65}(\theta) & \\ \hline & \theta - b_5(\theta) - b_6(\theta) & 0 & 0 & 0 & b_5(\theta) & b_6(\theta) \end{array}$$

where

$$\begin{aligned} a_{53}(\theta) &= \frac{\theta^2 c_4}{2c_3(c_4-c_3)} - \frac{\theta^3}{3c_3(c_4-c_3)}, \theta \in [0, c_5] \\ a_{54}(\theta) &= -\frac{\theta^2 c_3}{2c_4(c_4-c_3)} + \frac{\theta^3}{3c_4(c_4-c_3)},\ \theta \in [0, c_5] \\ a_{64}(\theta) &= \frac{\theta^2 c_5}{2c_4(c_5-c_4)} - \frac{\theta^3}{3c_4(c_5-c_4)},\ \theta \in [0, c_6] \\ a_{65}(\theta) &= -\frac{\theta^2 c_4}{2c_5(c_5-c_4)} + \frac{\theta^3}{3c_5(c_5-c_4)},\ \theta \in [0, c_6] \\ b_5(\theta) &= \frac{\theta^2 c_6}{2c_5(c_6-c_5)} - \frac{\theta^3}{3c_5(c_6-c_5)},\ \theta \in [0, 1] \\ b_6(\theta) &= -\frac{\theta^2 c_5}{2c_6(c_6-c_5)} + \frac{\theta^3}{3c_6(c_6-c_5)},\ \theta \in [0, 1]. \end{aligned}$$

Figure 1.3 displayes the couples (c_5, c_6) satisfying the relation (1.56). Among them we quote $c_5 = 1/2$ and $c_6 = 1$ and, symmetrically, $c_5 = 1$ and $c_6 = 1/2$.

It is worth remarking that conditions (1.55), which are sufficient for uniform order three and discrete order four, are also necessary when the abscissae c_2, c_3, c_4, c_5, c_6 are distinct.

The construction of higher order FCRK methods is in progress. So far it was proven that seven stages are sufficient for uniform order four and are necessary in case of distinct abscissae. An example of a seven-stage method of uniform order four is given by (1.47).

1.4.3 The quadrature problem

For RFDEs with distributed delay

$$\begin{cases} y'(t) = F\left(t, y(t), \int\limits_{t-\tau}^{t} k(t, s, y(t+s))\, ds\right),\ t \geq t_0 \\ y(t) = \phi(t),\ t \leq t_0 \end{cases} \tag{1.57}$$

the function f in (1.37) is given by

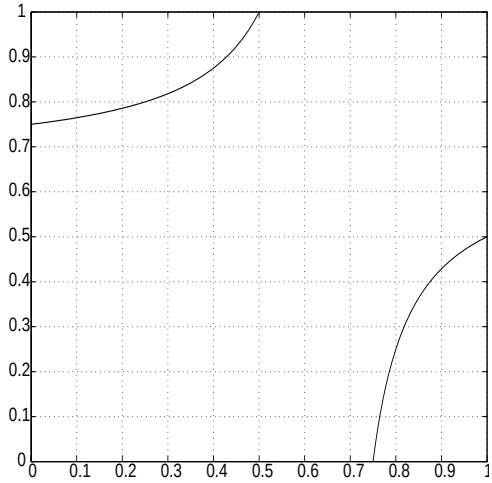

Fig. 1.3. The curves are the set of couples $(c_5, c_6) \in [0,1]^2$ satisfying the condition (1.56)

$$f(t,\varphi) = F\left(t, \varphi(0), \int_{-\tau}^{0} k(t,s,\varphi(s))\,ds\right)$$

and involves an integral. So, in general, we can provide only approximated values of f by a quadrature rule. In other terms, we shall use an approximation $\widetilde{f}$.

Another situation where an approximation of f is required is the RFDE

$$\begin{cases} y'(t) = F\left(t, y(t), \sum\limits_{m=0}^{\infty} k(t,m,y(t-\tau_m))\right), \ t \geq t_0 \\ y(t) = \phi(t) \ \ t \leq t_0 \end{cases} \tag{1.58}$$

where the function f is given by

$$f(t,\varphi) = F\left(t, \varphi(0), \sum_{m=0}^{\infty} k(t,m,\varphi(-\tau_m))\right)$$

In this subsection the effect of using an approximation $\widetilde{f}$ instead of f in a FCRK method is considered. We report only the main result whereas the details can be found in [32].

We denote by $\widetilde{f}(t,\varphi;\lambda)$ the approximation of $f(t,\varphi)$, where the parameter λ takes into account the adopted approximation procedure in the computation of $f(t,\varphi)$ such as, for example, the quadrature rule selected for the integral in (1.57).

Let us introduce the errors

$$\varepsilon_{n+1}^i = \widetilde{f}\left(t_{n+1}^i, y_{t_{n+1}^i}; \lambda_{n+1}^i\right) - f\left(t_{n+1}^i, y_{t_{n+1}^i}\right), \ i = 1, \ldots, \nu$$

where λ_{n+1}^i is the parameter relevant to the procedure used for the approximation of $f\left(t_{n+1}^i, Y_{t_{n+1}^i}^i\right)$ in (1.39). It can be easily proved that if

$$\varepsilon_{n+1}^{i} = O\left(\left(h_{n+1}\right)^{\min\{q+1,p\}}\right)$$

for all n and i, then the global order remains $\min\{q+1,p\}$ even if the values $f\left(t_{n+1}^{i}, Y_{t_{n+1}^{i}}^{i}\right)$ are replaced by their approximations $\widetilde{f}\left(t_{n+1}^{i}, Y_{t_{n+1}^{i}}^{i}; \lambda_{n+1}^{i}\right)$.

For instance, in case of (1.57), replacing the integrals in

$$F\left(t_{n+1}^{i}, Y^{i}\left(t_{n+1}^{i}\right), \int_{t_{n+1}^{i}-\tau}^{t_{n+1}^{i}} k\left(t_{n+1}^{i}, s, Y^{i}\left(t_{n+1}^{i}+s\right)\right) ds\right), i = 1, \ldots, \nu$$

by a composite l-point Gaussian quadrature rule, with

$$l = \left\lceil \frac{\min\{q+1,p\}}{2} \right\rceil,$$

across the intervals $\left[t_{n+1}^{i}-\tau, t_m\right] \subset [t_{m-1}, t_m]$, $[t_k, t_{k+1}]$, $k = m, \ldots, n-1$, and $\left[t_n, t_{n+1}^{i}\right]$, the global order $\min\{q+1,p\}$ is preserved.

We end this section by remarking that (1.57) can be numerically integrated also by the numerical methods, specifically designed for integro-differential equations, described in [10], [9] or [11] where the quadrature rule for the integrals is a part of the method.

References

1. C.T.H. Baker, C.A.H. Paul, and D.R. Willè, Issues in the numerical solution of evolutionary delay differential equations, *Adv. Comp. Math.* **3** (1995) 171–196.
2. H.T. Banks, J.M. Mahaffy, Stability of cyclic gene models for systems involving repression, *J. Theor. Biol.* **74** (1978) 323–334.
3. G.I. Bell, Mathematical model of clonal selection and antibody production, I, *J. Theor. Biol.*, **29** (1970) 191–232.
4. G.I. Bell, Mathematical model of clonal selection and antibody production, II, *J. Theor. Biol.*, **33** (1971) 339–378.
5. G.I. Bell, Mathematical model of clonal selection and antibody production, III, *J. Theor. Biol.*, **33** (1971) 378–398.
6. G.I. Bell, Predator-prey equations simulating an immune response, *Math. Biosc.*, **16** (1973) 291–314.
7. A. Bellen and M. Zennaro, *Numerical Methods for Delay Differential Equations*, Oxford University Press, Oxford, 2003.
8. R. Bellman and K.L. Cooke, *Differential-Difference Equations* Academic Press, 1963.
9. H. Brunner, The numerical analysis of functional and integro-differential equations of Volterra type. Acta Numerica (2004), 55-145.
10. H. Brunner and P. J. van der Houwen, *The numerical solution of Volterra equations*. CWI monographs 3, 1986.
11. H. Brunner, *Collocation methods for Volterra Integral and Related Functional Differential Equations*. Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, 2004.
12. K.L. Cooke, Functional-differential equations: some models and perturbation problems. *In Differential equations and dynamical systems*, (J.K. Hale and J.P. LaSalle eds), Academic Press, New York.
13. C. Cryer and L. Tavernini, The numerical solution of Volterra functional differential equations by Euler's method. SIAM J. Numer. Anal. **9** (1972), 105-129.
14. O. Diekmann, S.A. van Gils, S.M. Verduyn Lunel and H.O. Walter, Delay Equations: Functional-, Complex-, and Nonlinear Analysis, *AMS series* **110**, Springer-Verlag, Berlin, 1995.
15. L.E. El'sgol'ts and S.B. Norkin, *Introduction to the theory and application of differential equations with deviating arguments*, Academic Press, New York, 1973.
16. W.H. Enright and H. Hayashi, A delay differential equation solver based on a continuous Runge-Kutta method with defect control, *Numer. Algorithms* **16** (1998) 349–364.

17. A. Feldstein and K.W. Neves, High order methods forstate-dependent delay differential equations with nonsmooth solutions, *SIAM J. Numer. Anal.*, **21** (1984) 844–863.
18. J.A. Gatica and P. Waltman, A threshold model of antigen antibody dynamics with fading memory. In Lakshmikantham (Ed.) *Nonlinear phenomena in mathematical sciences Arlingtonn, Tex., 1980)*, Academic Press, New York, 1982, 425–439.
19. J.A. Gatica and P. Waltman, Existence and uniqueness of solutions of a functional-differential equation modeling thresholds, *Nonlinear Anal*, **8** (1984) 1215–1222.
20. J.A. Gatica and P. Waltman, A system of functional-differential equations modeling threshold phenomena, In Lakshmikantham (Ed) *Nonlinear analysis and applications (Arlington, Tex., 1986), Lecture Notes in Pure and Appl. Math.*, Dekker, New York, 1987, 185–188.
21. N. Guglielmi and E. Hairer, Implementing Radau IIA methods for stiff delay differential equations, *Computing* **67** (2001) 1–12.
22. N. Guglielmi and E. Hairer, Automatic computation of breaking points in implicit delay differential equations, submitted.
23. N. Guglielmi, On the Newton iteration in the application of collocation methods to implicit delay equations, *Applied Numer. Math.* **53** (2005) 281–297.
24. R. Hauber, Numerical treatment of retarded differential-algebraic equations by collocation methods, *Adv. Comput. Math.*, **7** (1997) 573–592.
25. G.W. Hoffmann, A theory of regulation and self-nonself discrimination in an immune network, *Europ. J. Immunol.*, **5** (1975) 638–647.
26. F. Hoppensteadt and P. Waltman, Did something change? Thresholds in population models, *Trends in nonlinear analysis*, Springer-Verlag (2003) 341–374.
27. E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*, Springer-Verlag, 1996.
28. E. Hairer and G. Wanner, Stiff differential equations solved by Radau methods, *J. Comput. Appl. Math.* **111** (1999) 93–111.
29. F. Hoppenstadt and P. Waltman, A problem in the theory of epidemics, *Math. Biosc.*, **9** (1970) 71–91.
30. Y. Kuang, *Delay Differential Equations with Application in Population Dynamics* (Academic Press, Boston, 1993).
31. J.M. Mahaffy, J. Bélair and M.C. Mackey, Hematopoietic Model with Moving Boundary Condition and State Dependent Delay: Applications in Erythropoiesis, *J. Theor. Biol.*, **190**, (1998) 135–146.
32. S. Maset, L. Torelli and R. Vermiglio, Runge-Kutta methods for Retarded Functional Differential Equations. Mathematical Models and Methods in Applied Sciences, 2005.
33. C.A.H. Paul, A test set of functional differential equations, *Technical Report 243, Univ. Manchester* (1992).
34. P.H. Richter, A network theory of the immune system, *Europ. J. Immunol.*, **5**, (1975) 350–354.
35. L. Tavernini, One-step method for the numerical solution of Volterra functional differential equations. SIAM J. Numer. Anal. **8** (1971), 786-795.
36. P. Waltman, A threshold model of antigen-stimulated antibody production, *Theoretical Immunology (Immunology Ser. 8) Dekker* (1978) 437–453.
37. P. Waltman and E. Butz, A threshold model of antigen-antibody dynamics, *J. Theor. Biol.* **65**, (1977) 499–512.
38. J. Wu, *Theory and Applications of Partial Functional Differential Equations*. Applied Mathematical Sciences 119, Springer, 1996.